



KONGSBERG

Group Members	Supervisors	Sensors
Jon Skjelsbæk	External Supervisor	External Sensor
Ming Kit Wong	Eirik Demmo Normann	Aage Vejlgaard Sørensen
Rita Hogstad		
Erica Fegri	Internal Supervisor	Internal Sensor
Thomas Mundal	José M. M. Ferreira	Karoline Moholth Mcclenaghan
Håvar Østrem		

Abstract

A rapidly expanding market for Cubesats encourages companies to explore business opportunities by innovating new solutions for use on small satellites. Cubesat is aimed to be modular and easily accessible for any who want to launch satellites into space. Cubesats larger than 8U has never been launched before, and even though these are still small satellites, they may require a more reliable power source than normally accessible on a conventional Cubesat. Engineering a solution for power collection and transfer has for larger satellites traditionally been tailored for each different satellite. Since Cubesat is aiming to standardize components, there may be a demand for a power transfer system for Cubesats larger than 8U when these become the norm. This report discusses a solution to a Solar Array Drive Mechanism for Cubesats as a plug-and-play solution, including power transfer and test bench.

Contents

- 1 Introduction** **10**

- 2 Background** **11**
 - 2.1 What is CubeSat? 11
 - 2.2 Costs and Risks of CubeSats 12
 - 2.3 Standards for CubeSat 13
 - 2.4 State of the Art 15

- 3 Project** **16**
 - 3.1 The Team and Employers 16
 - 3.1.1 Team Members 16
 - 3.1.2 University of South-Eastern Norway 17
 - 3.1.3 Division Space and Surveillance 17
 - 3.2 Process model 18
 - 3.2.1 Requirement analysis 19
 - 3.2.2 User Stories 19
 - 3.2.3 User Story Workload Estimation 19
 - 3.2.4 User Story Workflow 20
 - 3.2.5 Epics 20
 - 3.2.6 Sprints 20
 - 3.2.7 Stand-ups 20
 - 3.2.8 Scheduled Stakeholder Meetings 21
 - 3.2.9 Retrospective 21
 - 3.2.10 Process tools 21
 - 3.2.11 Roles and Responsibilities 22
 - 3.2.11.1 Product Owner 22
 - 3.2.11.2 Process Supervisor 23
 - 3.2.11.3 Requirement Manager 23
 - 3.2.11.4 Verification Manager 23
 - 3.2.11.5 Finance Manager 23
 - 3.2.11.6 Document Manager 24
 - 3.2.11.7 Risk Manager 24
 - 3.2.11.8 Weekly Responsibility 24
 - 3.3 Project Plan 25
 - 3.3.1 Gantt 25
 - 3.3.2 Presentations and Submissions 27
 - 3.3.3 Working Hours 27
 - 3.4 Handling and Understanding the Requirements 28
 - 3.4.1 Project Requirements 28
 - 3.4.2 Product Requirements 28
 - 3.5 The Process of Validation and Verification 31
 - 3.5.1 Validation 31

3.5.2	Verification	32
3.6	Project Risk	34
3.6.1	Risk Matrix	34
3.6.2	Project Risks	35
3.6.3	Technical Risks	36
3.7	Additional tools	36
4	Technical Work	39
4.1	Prototypes	39
4.2	Concept Development	39
4.3	Concept Model	41
4.4	Mark1	42
4.5	Mark2	44
4.5.1	The Structure	46
4.5.2	Twist Capsule	47
4.5.3	Flex	48
4.5.3.1	Flex Length and Twist Capsule Geometry	48
4.5.3.2	Flex Width	52
4.5.3.3	Rigid prototype	53
4.5.3.4	Prototypes of Flex Design	55
4.6	Mark3	56
4.6.1	Drivetrain	56
4.6.2	Motor	56
4.6.3	Gears	60
4.6.4	Bearings	60
4.6.5	Twist Capsule	61
4.6.5.1	Flex Shaft and Main Shafts	61
4.6.6	Power Transfers	63
4.6.7	Flex	65
4.6.7.1	Scemactical Drawing	65
4.6.7.2	Padstacks and footprints	66
4.6.7.3	PCB Layout	66
4.7	Space Environment Effects	66
4.7.1	Vacuum	67
4.7.2	Atomic Oxygen	67
4.7.3	Ultraviolet Radiation	67
4.7.4	Particulate or Ionizing Radiation & Hydrogen	67
4.7.5	Plasma	68
4.8	Materials	68
4.9	Finite Element method	69
4.9.1	Launch	70
4.9.2	Radial, Axial and Torsional Forces	71
4.9.3	Vibration analysis	73
4.9.4	Thermal analysis	74

4.10	Diagnostics System	75
4.10.1	Software Layer	76
4.10.1.1	Alpha 0.1	76
4.10.1.2	Alpha 1	77
4.10.1.3	Alpha 2	79
4.10.1.4	Alpha 3	81
4.10.1.5	Beta 1	82
4.10.1.6	Beta 2	86
4.10.2	Hardware Layer	88
4.10.2.1	Hardware	88
4.10.2.1.1	Control Cabinet	89
4.10.2.1.2	Feedback and Sensors	90
4.10.2.2	Firmware	92
4.10.2.2.1	Motor Driver and Parameters	93
4.10.2.2.2	Alpha 0.1	95
4.10.2.2.3	Alpha 1	95
4.10.2.2.4	Alpha 2	97
4.10.2.2.5	Alpha 3	98
4.10.2.2.6	Beta 1	98
4.10.2.2.7	Beta 2	99
4.10.3	Virtual Prototype	100
5	The Result/Final Prototype	102
5.1	Physical Prototype	102
5.1.1	Mechanical assembly	102
5.1.2	Power Transfers	104
5.2	Diagnostic System	105
5.2.1	Hardware layer	105
5.2.1.1	Control Cabinet	105
5.2.1.2	Firmware	107
5.2.2	Software layer	107
5.2.3	The finance	108
5.3	Project Challenges	108
6	Conclusion	111
6.1	Summary	111
6.2	Reflections	112
6.3	Future Work	112
7	Bibliography	114
A	Appendices	121
A.1	Expenses and invoices	
A.2	Calculation of Inertia	
A.3	Models and sketches of configuration	
A.4	Design Description	

A.5 User Manual - Mark3

A.6 Bonding test

A.7 Electrical Schematics - DS HWL

A.8 Mail correspondence - OEM's sales office

A.9 Flex Comparison Tables

A.10 DS SWL Design documents requirements

A.11 DS SWL Alpha 1 requirements

A.12 DS SWL Alpha 2 requirements

A.13 DS SWL Alpha 3 requirements

A.14 DS SWL Beta 1 requirements

A.15 DS SWL Beta 2 requirements

A.16 Verification report: VR-SWL-A1

A.17 Verification report: VR-SWL-A2

A.18 Verification report: VR-SWL-A3

A.19 Verification report: VR-SWL-B1

A.20 Verification report: VR-SWL-B1-2

A.21 Verification report: VR-SWL-B2

A.22 DS communication protocol

A.23 SatStat user manual

A.24 HWL Decision document

A.25 Requirements HWL Alpha 1

A.26 Class diagram HWL Alpha 1

A.27 Verification report HWL Alpha 1

A.28 Requirements HWL Alpha 2

A.29 Class diagram HWL Alpha 2 & 3

A.30 Verification report HWL Alpha 2

A.31 Requirements HWL Alpha 3

A.32 Verification report HWL Alpha 3

A.33 Requirements HWL Beta 1

A.34 Class diagram HWL Beta 1

A.35 Verification report HWL Beta 1

A.36 Requirements HWL Beta 2

A.37 Class diagram HWL Beta 2

A.38 Verification report HWL Beta 2

A.39 LinkedList class diagram

A.40 SSTL class diagram

A.41 LinkedList API documentation

A.42 SSTL API documentation

A.43 HWL Alpha 1 API documentation

A.44 HWL Alpha 2 & 3 API documentation

A.45 HWL Beta 1 API documentation

A.46 HWL Beta 2 API documentation

A.47 Control Cabinet: List of hardware

A.48 SADM drivetrain requirements

A.49 SADM housing requirements

A.50 SADM motor requirements

A.51 SADM power and signal transfer

A.52 Project Planning

A.53 Areas of responsibilities

A.54 Timesheet and time summary

A.55 Group contract

A.56 LaTeX Bibliography Guide

A.57 V-Model inspired workflow

A.58 Procedure for agenda and minutes

A.59 Team Naming and Logo Design

A.60 Requirement sheet description

A.61 Requirements

A.62 VMD Description

A.63 Verification Management Document

A.64 Risk Tables

A.65 SWOT Analysis

A.66 Meeting requests

A.67 Epic Reports

A.68 Version Reports

A.69 Sprint Reports

A.70 Meeting notes

A.71 Agreement of electronic distribution

List of Figures

1.1	Illustration of 1U CubeSat in space. Source: NASA	10
2.1	The different sizes of CubeSats. Source: ECM-space	11
2.2	Specification drawing of a 2U, according to the Cal Poly-standard. Standardiza- tion creates off-the-shelf products that reduces the price. Source: Cal Poly	12
2.3	A dummy of a 16U CubeSat and the associated deployer. Deployers are setting most of the requirements for CubeSats. Source: ISISpace	13
2.4	fifty-five 16U's are planned to be launched over the next six years. Facts as of January 19,2019. [1]	14
2.5	The 6U CubeSats MarCO A and B are the first CubeSats in deep space. Source: JPL NASA	15
3.1	The product requirements hirerachy.	29
3.2	The intended flow for each requirement	30
3.3	The updated requirement analysis.	31
3.4	Six by four risk Matrix	34
3.5	Selected project risks	35
3.6	A few selected technical risks	36
4.1	Lego components	40
4.2	Motor perpendicular with shaft	40
4.3	Drawing of the inside ratio 1:1	41
4.4	Concept model of 8U (left) and 16U (right)	42
4.5	Mark1 exploded view	43
4.6	The inside of the physical prototype Mark1	43
4.7	Mark1	44
4.8	Mark2	45
4.9	Mark2 without structure	45
4.10	The structure of M2 in the orientation it would have been printed.	46
4.11	Twist Capsule components for Mark2	47
4.12	Relationship between minimum flex length, shaft radius and capsule radius	50
4.13	Flexculator	52
4.14	The interface between flex and wires must fit inside the flex shaft.	53
4.15	Copper tape flex.	54
4.16	Flexible PCB with no mechanical connectors	55
4.17	Flexible PCB, widened connectors	55
4.18	Circuit model of one phase in a stepper motor. [2, Chapter 5.2]	59
4.19	Gear Assembly	60
4.20	Versions of Flex Shaft	61
4.21	Main Shaft Versions	62
4.22	Configuration with one, large lead each direction for each solar array.	64
4.23	Several alternatives are achievable by having more leads to each solar array. . .	64
4.24	Flexible PCB, second iteration	65



4.25 Flexible PCB, second iteration	66
4.26 Aluminum 7075-T6 Mechanical Properties [3]	68
4.27 Offer from WayKen	69
4.28 Launch 10G	70
4.29 Requirement R5-06	71
4.30 Maximum Moment	71
4.31 Maximum Radial and Axial force	72
4.32 Max Radial and Axial scale 150	73
4.33 Simplified drivetrain 2000Hz	73
4.34 Thermal analysis with SolidWorks	74
4.35 The left lid after modification.	75
4.36 Use case diagram DS SWL Alpha 1	78
4.37 Class diagram DS SWL Alpha 1	78
4.38 Diagnostics system software layer alpha 2 class diagram	80
4.39 Alpha 3 use-case diagram	82
4.40 Class diagram for the ObservableNumericValue system	84
4.41 Class diagram for the ObservableNumericValueCollection system	85
4.42 Class diagram concerning test configuration	87
4.43 Beta 2 use case diagram	88
4.44 The control cabinet's role in the system	89
4.45 The sensors send data to the firmware for interpretation before it is sent to the software layer	90
4.46 HWL interconnection diagram	96
4.47 Diagnostics System HWL architecture diagram.	97
4.48 Mark 3 virtual prototype for Hololens, exploded and expanded view	100
4.49 Virtual SADM prototype on 16U cubesat	101
5.1 Mark3 - Space Final CAD Design	102
5.2 The Mass properties of Mark3 CAD	103
5.3 Mark3 Prototype	103
5.4 Photography of the Flex, without leads	104
5.5 Photography of the Flex, with leads	104
5.6 Photography of the Flex, with leads	104
5.7 The control cabinet and the test station.	106
5.8 Mark3 and control cabinet	106
5.9 Diagnostics system software layer user interface	107
5.10 Requirement R3-01	109

Abbreviations

ABS	Acrylonitrile Butadiene Styrene
AO	Atomic oxygen
CAD	Computer-aided design
Cal Poly	California Polytechnic State University
CCW	Counter clockwise
CDC	Cubesat design specification
CW	Clockwise
DS	Diagnostics system
DSS	Division space and surveillance
ECSS	European cooperation for space standardization
EEE	Electrical and electronics engineering
ESA	The European space agency
FEM	Finite element method
Flex	Flexible printed circuit board
FPGA	Field-programmable gate array
HMI	Human machine interface
HWL	Hardware layer
IDE	Integrated Development Environment
JPL	Jet propulsion laboratory
JSON	JavaScript Object Notation
KDA	Kongsberg defence and aerospace
LEO	Low Earth orbit
LV	Launch vehicle
M1, M2, M3	Mark1, Mark2, Mark3
MarCO	Mars cube one
MCU	Microcontroller unit
MMOD	Micrometeoroids and orbital debris
MR	Mixed Reality
NASA	National aeronautics and space administration
NEMA	National electrical manufacturers association
O	Oxygen
O ₂	Allotrope of oxygen
O ₃	Ozone
P-POD	Poly pico satellite orbital deployer
PCB	Printed circuit board
PE	Protective Earth
SADM	Solar Array Drive Mechanism
SSO	Sun-synchronous orbit
SSTL	Subscriber System Template Library
SWL	Software layer
TE	Tronrud Engineering
U	Unit. Used to describe size of a CubeSat. $1U = 10m^3$
USN	University of South-Eastern Norway
VMD	Verification management document

Chapter 1

Introduction

The scope of this project were separated into two subsystems, a physical prototype of a Solar Array Drive Mechanism (SADM), and a Diagnostics System (DS). The SADM were developed for a 16u Nanosat. Its main responsibility is to rotate the solar panels attached to the satellite so that they always face the sun. The mechanism is also responsible for transferring power from the solar arrays to the satellite power system. The Diagnostics System is a monitoring solution complementary to the SADM. It consists of a Human Machine Interface (HMI) and a physical controller. The combination of the HMI and the controller allow the user to define and execute different tests on the SADM. During execution, various feedback data is recorded and visualized in the HMI. This data is used to determine if a given test is considered successful or failed.



Figure 1.1: Illustration of 1U CubeSat in space. Source: NASA

Chapter 2

Background

When professors Jordi Puig-Suari of California Polytechnic State University (Cal Poly) and Bob Twiggs of Stanford University brought to life the idea of creating a compact satellite in 1999, their goal was to encourage students to increase interest for space crafts by letting them develop a satellite within 10x10x10 cm for experimenting and to try out new technology. [4]

The concept of CubeSat was originally intended to be a student project, and is now used by government and companies world wide. A combination of size, light weight and a simple structure revolutionized the satellite industry by making it more affordable to build and launch a satellite for new and emerging players. This chapter is describing CubeSat in general.

2.1 What is CubeSat?

A CubeSat is a satellite made up of modular cubes of approximately 10cm on each sides. The size of one cube is known as 1U (one unit), where the mass is limited to 1.33 kg according to the CubeSat Design Specification (CDS), also known as the Cal Poly-standard. When a CubeSat is scaled up to the double of 1U, it is called 2U and so on. Since the limitation of mass is depending on the deployer that releases the payload from the rocket, the mass can vary from 0.8 kg to 2 kg per unit. [1]



Figure 2.1: The different sizes of CubeSats. Source: ECM-space

The general classification for satellites are defined by their mass. A Nanosatellite (Nanosat) is 1 to 10 kg, and a microsatellite is 10 to 100 kg. The smallest existing CubeSat design is 0.5U and the biggest 27U, making the mass limitation of CubeSats approximately 0.5 kg to 40 kg. A satellite between 0.1 kg and 1 kg is referred to as a picosatellite. A CubeSat is therefore sometimes referred to as a picosat, Nanosat or microsat depending on the weight. Smaller satellites exist, called PocketQube and SunQube, but this report will not discuss these in further detail. [1]

CubeSats are used in low earth orbit (LEO) for earth observations, signal monitoring, communication, scientific application and more. Nanosats are launched into Polar orbit of constellation and make it possible to cover big areas of the Earth with a lower cost than a bigger satellite. The lifetime of a CubeSat varies, but it can be up to twenty years or more. It will eventually reenter the atmosphere and disintegrate with the help of gravity or thrusters after operational time. [5]

2.2 Costs and Risks of CubeSats

The cost of building a CubeSat strongly depends on production time and the components selected for the satellite. This is because components like structures and solar cells have become off-the-shelf products as there are specifications that define the shape and volume of a CubeSat. Launching a Nanosat is also much cheaper than conventional satellites as Nanosats are so small that they can share launch vehicles (LV) with other satellites. Some companies also offer smaller rockets designed specifically for Nanosats, and NASA even provides free launches for Nanosats with scientific applications. Therefore most of the costs for building a CubeSat comes from materials, labor, testing and travel. [6]

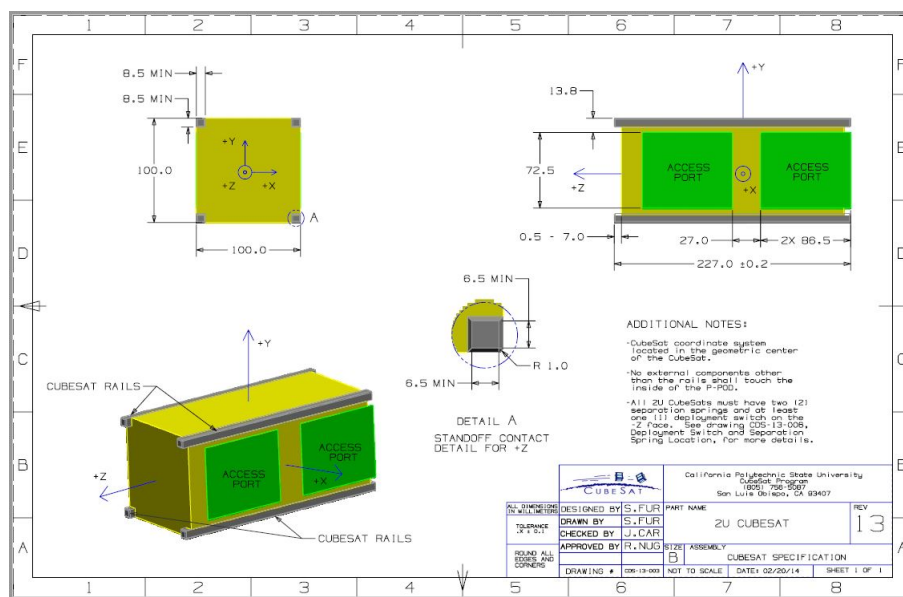


Figure 2.2: Specification drawing of a 2U, according to the Cal Poly-standard. Standardization creates off-the-shelf products that reduces the price. Source: Cal Poly

Having constellations in orbit for an operation also reduces the risk in case of technical failure. If a component in a conventional satellite fails, the worst case is that it could jeopardize the whole operation. For a group of constellations, a unit that fails can simply be replaced by another that is launched later. [5]

2.3 Standards for CubeSat

As more people started to build CubeSats, a standard was needed. It was natural for Cal Poly to take the responsibility for this since this is where it began [7]. The CubeSat Design Specification (CDS) by Cal Poly is therefore known as the general CubeSat standard. The standardization is one of the biggest advantages of CubeSat, as this creates off-the-shelf products like primary structure and solar arrays at a reduced cost compared to custom made equipment.



Figure 2.3: A dummy of a 16U CubeSat and the associated deployer. Deployers are setting most of the requirements for CubeSats. Source: ISISpace

A big portion of the CDS describes the mechanical requirements to the exterior as it is aimed towards the interface of the deployer. This is a housing that contains the CubeSats during launch, and it functions as the interface between the LV and the payload. When the right attitude is reached, a door opens up and a loaded spring pushes the payload along a series of rails and releases the payload into space. This is the general mechanical function of a deployer, regardless of the size.

Some of the most known deployers are Poly Pico Satellite Orbital Deployer (P-POD) by Cal Poly and ISIPOD by Innovation Solution In Space. What they have in common is that they can both contain 3U each, meaning the payload can be combinations of 3x1U, 2U+1U, 2x1.5U or one 3U Nanosats per deployer. Figure 2.4 from the Nanosat database reveals that the most common sizes of CubeSats are 1U (14,2%) and 3U (46%), when including launched and not launched CubeSats.

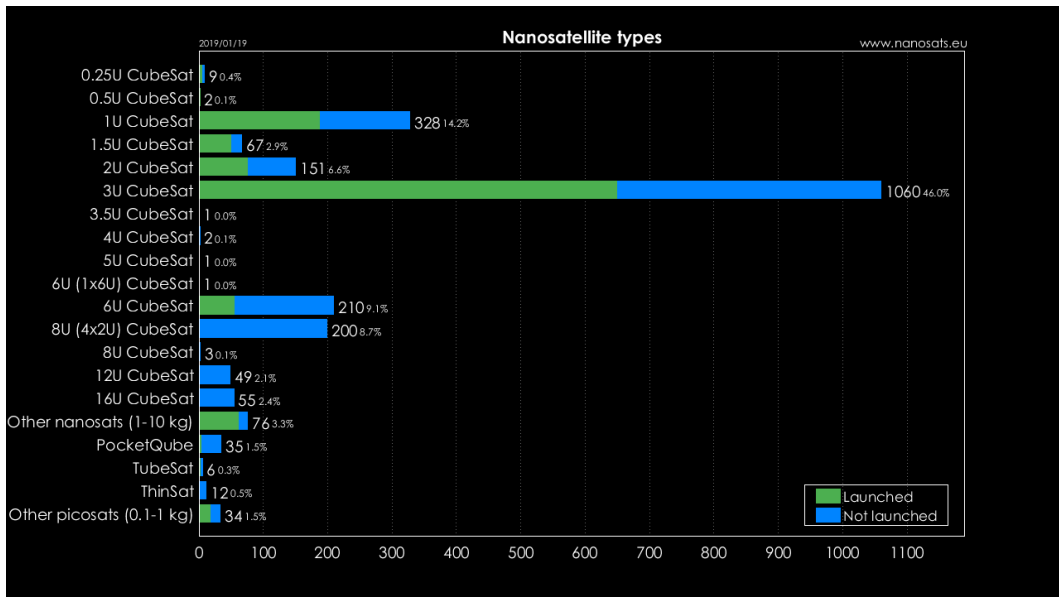


Figure 2.4: fifty-five 16U's are planned to be launched over the next six years. Facts as of January 19,2019. [1]

The following table shows the standards for mass and measurements of CubeSats in the most common sizes and 16U. The SatLight engineering team has not succeeded in finding standards for 16U. It may seem that there are no standards for CubeSats larger than 6U yet, as this is less common. Specification drawings are available, usually supplied by distributors of structures or deployers. Therefore, it seems that the design requirements for larger CubeSat are determined by the interface of the deployer.

	Length	Width	Height	Mass
1U	100	100	113.5	1.3 kg
2U	100	100	227+/-0.2	2.66 kg
3U	100	100	340.5+/-0.5	4 kg
16U	226.3	226.3	454	24 kg

Table 2.1: Some CubeSat design specifications

Other requirements by Cal Poly that should be noted are the rails and extrusions. The rails are a part of the structure and extends from the corners and 8.5 mm in the direction of the center of the wall. They are designed to slide along the P-POD's rails and function as the interface between the payload and the deployer. The fact that these are the only surfaces in contact with each other, creates a slip in between the payload and the deployer. This allows extrusions to be up to 6,5 mm from the walls of the CubeSat. This space is usually taken up by an antenna, solar panels or screw heads [8]. Other developers of deployers have adapted these rails in their design to accommodate the CubeSat Design Specification, but not all of them. The use of Planetary Systems Corporation's deployer requires that the CubeSat has a tab on two of the faces opposite of each other instead of rails. This differs from the Cal Poly standard, but the advantage is that they allow almost double of the mass per unit compared to the Cal Poly standard [9]

2.4 State of the Art

On May 5, 2018 two 6U CubeSats were launched to Mars, along with a stationary lander called InSight. The twins are named Mars Cube One (MarCO) A and B. If they make it to Mars, their mission is to relay information down to Earth about InSight's descent and touch down, and to test out new miniaturized technology. This is the first time NASA has sent a CubeSat into deep space. [10]

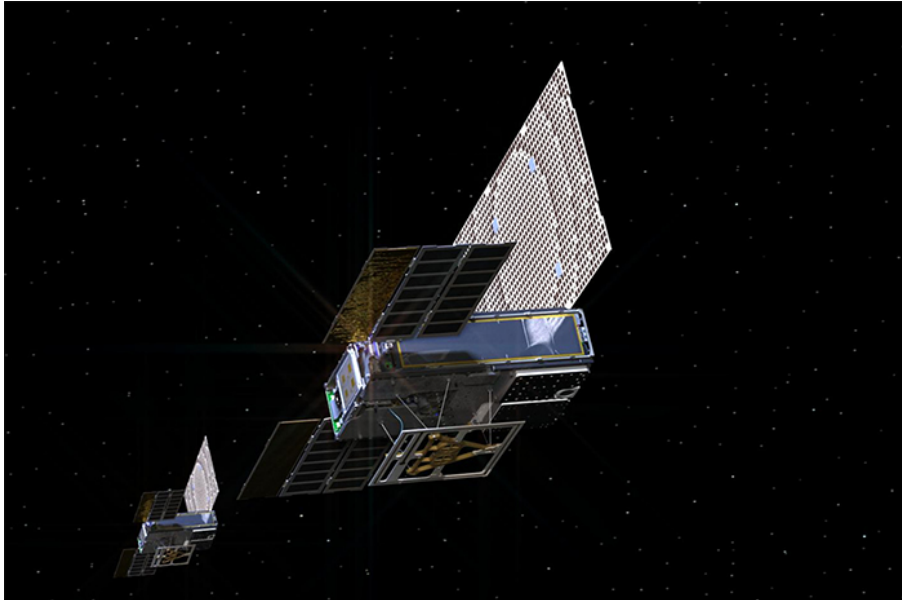


Figure 2.5: The 6U CubeSats MarCO A and B are the first CubeSats in deep space. Source: JPL NASA

CubeSats get their power from solar arrays, and most small CubeSats have their solar arrays attached on their walls, some CubeSats use Solar Array Drive Mechanisms. In addition to DSS - this project's stakeholder and commissioner, there are many companies that make these types of technology, although not many of these companies make SADMs for small satellites such as CubeSats. On NASA's Technical Report Server online [11] a report on a design of a SADM for a CubeSat [12] is available. The company that has written this report is named Honeybee Robotics who are developing space technology, and clearly the technology for SADMs on CubeSats has been around for a few years due to the fact that the report is from 2010.

Chapter 3

Project

3.1 The Team and Employers

3.1.1 Team Members



Mechanical Engineering
Product owner
Rita Hogstad
ritahogstad@gmail.com



Computer Science
Process manager
Thomas Mundal
thmundal@gmail.com



Electrical Engineering
Requirements manager
Erica Fegri
erica.fegri@gmail.com



Mechanical Engineering
Verification manager
Finance manager
Ming Kit Wong
mingkit.wong@gmail.com



Electrical Engineering
Risk manager
Håvar Østrem
havar.ostrem@gmail.com



Computer Science
Document manager
Jon Skjelsbæk
skjelsbek@gmail.com

3.1.2 University of South-Eastern Norway

The University of South-Eastern Norway has approximately 18,000 students and 1,600 employees, spread over eight campuses: Drammen, Vestfold, Kongsberg, Ringerike, Bø, Notodden, Porsgrunn and Rauland. USN aims to have a regional foundation, and with eight campuses, they have a strong and clear presence in one of Norway's most exciting and dynamic regions. USN was established on 1 January 2016, when Buskerud and Vestfold University College merged with Telemark University College [13].

Internal Sensor

Karoline Moholth Mcclenaghan

University lecturer, For the University of South-Eastern Norway campus Kongsberg

The internal project sensor's task is to censor all the bachelor's theses. Internal project sensor shall be part of a panel of three persons who will place individual grades on the project participants. Internal project sensor shall also be a support for internal supervisors during the implementation of the bachelor's thesis. The natural thing is that internal sensor is the one who controls the censorship process.

Internal Supervisor

José M. M. Ferreira

Professor, For the University of South-Eastern Norway, campus Kongsberg

The internal project supervisor's task is to support and guide the students during the completion of the actual bachelor's thesis. Internal project supervisor must also be part of the panel grading the project participants. Internal project supervisors' tasks can be divided into these three groups: project execution, assessment and additional inquiries to / by students. In addition, the internal project supervisor must be present at the weekly project meetings.

3.1.3 Division Space and Surveillance

Division Space and Surveillance is a division under KONGSBERG Defence & Aerospace. They are the leading supplier of equipment for scientific satellites, earth observation satellites, and launchers in Norway. Their expertise is development, qualification, and delivery of products like booster attachments and release mechanisms for launchers, rotation and pointing mechanisms for solar arrays and antennas, drive and control electronics, and electro-optical systems [14].

External Sensor

Aage Vejlgaard Sørensen

Department Manager, Engineering and Production for KONGSBERG

The external project sensor's task is to censor the thesis. An external project sensor will be part of the same panel as the internal project supervisor and internal project sensor. The external project sensor must be present at all three presentations.

External Supervisor

Eirik Demmo Normann

Project Engineer, for KONGSBERG

External project supervisor's task is twofold: First, an external project supervisor is a representative of those who have initially given the task and are thus a customer. Second, the external project supervisor is responsible for ensuring that the necessary resources are made available to the project group. Resources in this case include both equipment / software and professional information / guidance. The external project supervisor has no direct responsibility for grading, but he / she is expected to be present at all three presentations and is expected to participate actively in the evaluation process.

3.2 Process model

The process model used in this project is an agile model based on SCRUM. The complete model has most of the elements that SCRUM has, but with some modifications. The modifications made are listed ahead.

- Modified roles
- Custom workflow
- Modified terminology

The workflow was inspired by elements of traditional engineering process models. The reason for incorporating a custom workflow is based on wishes from our stakeholders. The reason for choosing an agile model, is to comply with requirements from the academic institution that has initialized this project. Utilizing an agile model enhanced cooperation when being a multi-disciplinary group. Being able to continuously develop prototypes and versions of the product allowed for continuous testing and validation of concepts introduced in new versions. Validation of concepts with the internal- and external supervisors was a key aspect in regards to delivering a product fulfilling the stakeholder needs. The name of this specific process model is SatLight process model.

3.2.1 Requirement analysis

An organized list of requirements shall be formed and prioritized according to the specification of the product given by stakeholders. The requirements shall be measurable and have a defined test to verify them, as well as methods for validation. User stories shall be formed from the requirements, and all tests, validations, risks and user stories shall be traceable back to the original requirement by referencing an identification for each of these parts.

3.2.2 User Stories

A user story is formed from either a single requirement, or a group of requirements that is relevant for a single function or unit. User stories shall be written in a "high level" form, so that all members of the team, including stakeholders, understand what the user story is about.

If the user story is very comprehensive, it can be divided into sub-tasks so that the different objectives contained within the story are more clearly defined. The sub-tasks are usually discipline-specific and can therefore contain a more technical description.

A user story should be traceable back to the parent requirement(s) and tests that should verify that the work generated by the user story produces a result that is compliant with the parent requirement. After user stories are formed, they are put in a product backlog.

Only when the user story is verified and validated is the story considered done. If a user story is not validated or completed before its parent sprint is expired, it should be put in a status that described where it failed, and transferred to the next sprint. This type of event should be kept to an absolute minimum.

3.2.3 User Story Workload Estimation

Every user story shall have an estimation of the complexity of the work described. The goal for estimation is to record and predict how much work the group can do during a sprint.

Estimation of user stories follows this scale, where 1 is the easiest and 10 is the most time consuming and complex:

1	3	5	7	10
XS	S	M	L	XL

3.2.4 User Story Workflow

Every user story should follow a workflow that mainly consist of the following phases:

- Design
- Production/Implementation
- Testing/validation

3.2.5 Epics

An epic is a group of user stories that are used to categorize user stories for internal organization and tracking progress on a higher level than user story level. If one part of a product requires a series of user stories to be considered complete, these user stories can be put into an epic describing the overall functionality of the particular product part.

3.2.6 Sprints

A sprint describes a time-period containing planned work. The time-period for a sprint is one week (5 work-days, 7 days total). Sprint starts every Monday at 10:00. Sprint ends the following Monday at 09:00. A sprint evaluation meeting is held every Monday from 09:00 to 10:00 where the previous sprint is finished and next sprint is planned. All user stories in a sprint should to the best of the teams ability be verified and done before the sprint expires.

3.2.7 Stand-ups

The team shall meet for a stand-up every morning at 09:00 for a time-period of no longer than 15 minutes. The stand-up shall include the following elements:

- What did you do yesterday
- What will you do today
- Do you have any problems progressing with current assigned tasks?

The aim of the stand-up meeting is to provide transparency in the group, where every group member is up to date with each others' progress. If any member has any challenges in progressing with their assigned tasks, they have the opportunity to tell the group, and request to discuss potential solutions after the stand-up. There are no discussion allowed during stand-up. The 15 minutes assigned to this meeting is purely to share information that are important and relevant for the progress of current work.

3.2.8 Scheduled Stakeholder Meetings

A meeting with internal supervisor shall be held every Friday at 10:00. A meeting with external supervisor shall be held every Wednesday at 10:00.

3.2.9 Retrospective

A team retrospective meeting shall be performed at least once every 14 days. The goal of a retrospective meeting is to improve the work environment within the team. The meeting starts with a 15 minutes talk which aims to help the team members to prepare for the next step. The next 15 minutes is silent, where every member writes their inputs on post-it notes. The notes should not be shared until these 15 minutes are elapsed. There are three categories for the notes: keep doing, stop doing and ideas/questions. In the next part of the retrospective, one team member at a time places their post-it notes in the appropriate area on the whiteboard. Each area should be restricted to notes of a given category. The member puts all their notes up in one go, and as they do, they have the opportunity to explain the meaning of the note. The notes that refer to the same or similar things are grouped together. After everyone has placed their notes on the board, the last part of the retrospective begins: voting. All members get three votes over all, and can distribute them freely on the notes on the board. The voting process continues until there are three winners. These are the three subjects that the team collectively has deemed most important, and the team should act on those subjects.

3.2.10 Process tools

- **Jira**

Jira is a project management tool made by Atlassian. This is a helpful tool for organizing agile teams with the use of Scrum board functions [15].

A big part of the process of identifying useful process models was to also find tools that would be beneficial to incorporate as part of that process model. One of the criteria for these tools was that it should be easy to use, and easy to incorporate into the process model that was chosen. After considering several online project management tools, the systems made by Atlassian proved to be the most useful.

With Jira it is possible to add User stories in the backlog, prioritize and set user story workload estimation for each user story. When user stories are added to sprint and this sprint is started, Jira keeps a good overview of what the team members are doing through the sprint board. This contributed to transparency, something that is a core principal in the chosen process model.

Information entered and store in Jira can be exchanged with other Atlassian products, such as Confluence.

- **Confluence**

Confluence is a team collaboration software provided by Atlassian [16].

Confluence was chosen as a compliment to Jira, and serves as a platform for documentation and information sharing with stakeholders. Document templates for meeting reports and product requirements proved to be very useful in this project.

- **Slack**

Slack is the application for communicating that was used internally among the team members created by Slack technologies [17].

For a neat communication, different channels were made for each use; each discipline had its own channel where all members could go in and ask about relevant things concerning the discipline's discipline. In addition, it was possible to send private messages between all group members. In addition to the mentioned channels, the group had three common channels: "To do" - which was a channel where things that had to be done were written and when it was done, the group members would press green tick on this message. In the channel "general" notices and general messages related to the project were written. The third common channel was "random" which served as a channel for less important things.

As for the written communication with our supervisors and sensors, E-mail was used.

3.2.11 Roles and Responsibilities

The role of each individual team member had to be decided at the start of the project. This was for the project group to decide, and the selected roles are explained in the sections ahead.

3.2.11.1 Product Owner

Usually the product owner in a Scrum team is the key stakeholder of the project, which can also be the customer. In our project the product owner's job is to maximize the value of the product produced by the development team. The main task of the product owner in the case of this project is to manage the backlog, which include the tasks listed below [18]:

- Clearly expressing Product Backlog items
- Ordering the items in the Product Backlog to best achieve goals and missions
- Optimizing the value of the work the Development Team performs
- Ensuring that the Product Backlog is visible, transparent, and clear to all, and shows what the Scrum Team will work on next

The product owner remains accountable for these tasks but can also delegate tasks for the development team. If any of the group members wishes to change a user story or the priority of a user story in the backlog, the product owner should be addressed, since it is the product owner's job to keep track of this. The product owner's job is to motivate the team with a clear,

elevating goal [19]. Team members know best what they are capable of, and so they select which user stories from the top of the product backlog they can commit to delivering during any sprint.

To keep an overview of the project, the product owner is responsible of making a project plan.

3.2.11.2 Process Supervisor

The role of process supervisor mirrors the role of scrum master from SCRUM. The responsibilities include making sure the teams adhere to the process, and that the process is used correctly. Process supervisor is also responsible for educating team members and stakeholders about the process. An additional responsibility added is that the process supervisor should ensure that the software tools used in conjunction with the process model are used correctly, and work as intended.

3.2.11.3 Requirement Manager

As a requirement manager the main responsibility is to ensure that the team has a system for analyzing, formalizing and quantifying the requirements. The requirement manager shall ensure that the requirements are documented in a way that connected information and traceability is achieved and recorded in a framework. When requirements are changed, removed or when new requirements arrive, it is the requirements manager's responsibility to ensure that these changes are recorded.

3.2.11.4 Verification Manager

The overall task for the verification manager is to ensure validation and verification of all the requirements. To do so, all requirements needs to be verified to ensure that the product will function as intended. Verification and validation will be performed by all the team members, but it is the verification manager's responsibility to ensure that the project test plan and specifications are followed.

3.2.11.5 Finance Manager

The main task of the finance manager is to ensure proper accounting. The finance manager is also responsible for organizing purchases and to compare the price of our product to competitors, if possible.

3.2.11.6 Document Manager

Responsibilities underlying the document manager position are related to general document structure in the form of creating templates and defining standards, as well as making sure hierarchical structure is maintained e.g. in the Google Drive archive. The document manager also has to make sure that proof-reading is properly conducted before a document is considered done, and that templates and standards are correctly used. Just like the other roles, the document manager is not single-handedly responsible for doing all of this, but has to make sure it is done.

3.2.11.7 Risk Manager

The responsibility of the risk manager is to reduce the likelihood of events that could impact the scope of the project, the stakeholders and the product itself. It is also the risk manager's responsibility to reduce the severity of an impact if possible. All members of the team are responsible for identifying risks associated with their own work and for reporting it to the risk manager. The risk manager is responsible for identifying overall project risks and to analyze all identified risks. Based on the risk analysis, the risk manager shall decide what risk management strategies should be approached to avoid or mitigate the risk, or whether the risks are acceptable.

3.2.11.8 Weekly Responsibility

For each week there was one chair person and one referent. Different team members were assigned to these roles each week in the following order:

1. Erica
2. Thomas
3. Rita
4. Håvar
5. Ming Kit
6. Jon

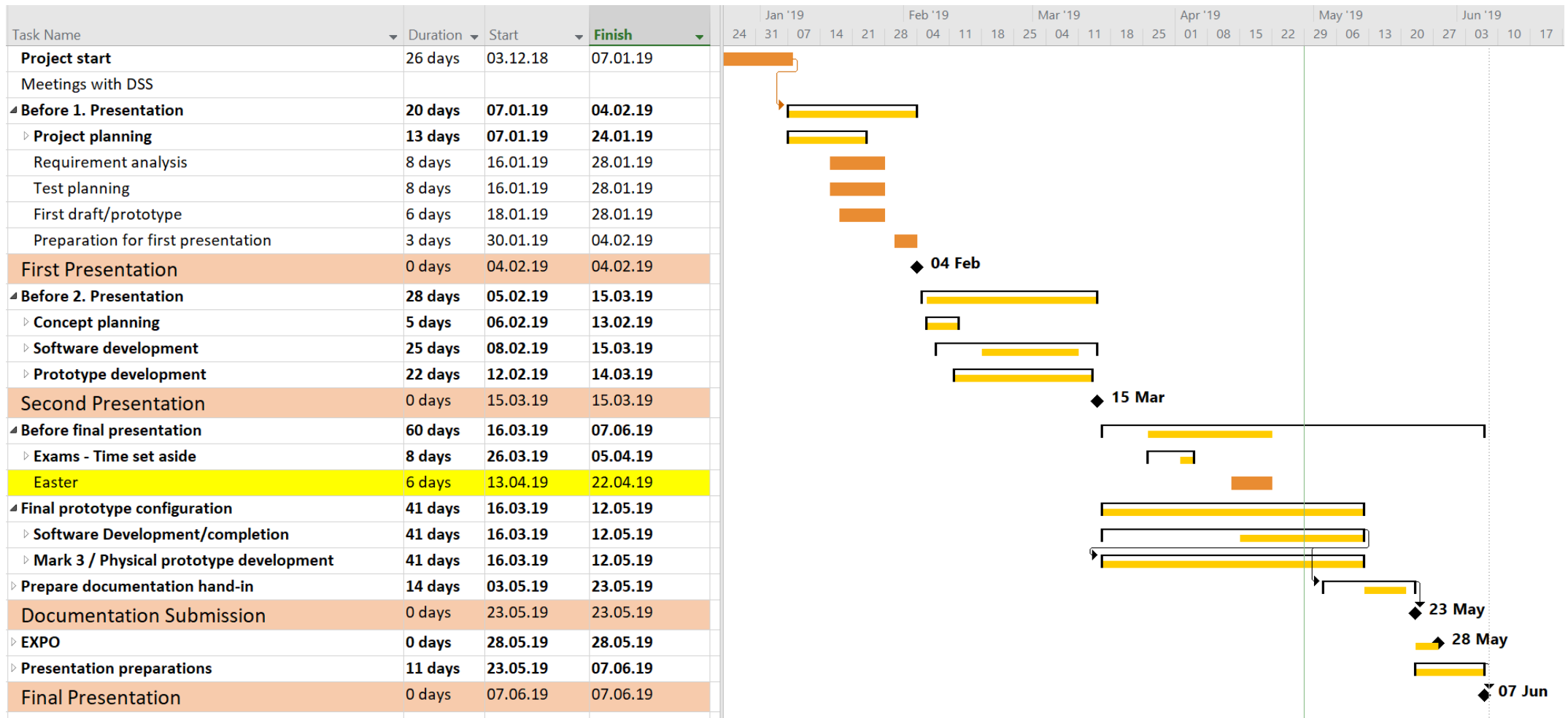
More information about our procedure is available in Appendix A.58.

3.3 Project Plan

A successful project requires a suitable project plan. It provides a common understanding of milestones, and helps the team realize the expectations of the project. This section addresses the overall project plan, events taking place during the project and the intended project time span.

3.3.1 Gantt

A Gantt chart was made at the beginning of the project to provide an indication of what the project time-line would look like. Using an iterative process model made it possible to split the different subsystems into version. There were three versions defined for the physical prototype and six for the diagnostics system. At first, the milestones were defined as the three presentation. As the concepts and design were determined between the first and the second presentation, additional milestones were created for the different disciplines. A shortened version of the Gantt chart is available at the page ahead, and the original expanded Gantt chart can be found in attachment A.52.



3.3.2 Presentations and Submissions

During the project, the team had three mandatory presentations for sensors, supervisors and those who were interested in attending. In addition, the team had a presentation at an event organized by the students at the University called “Space Night”. Just before the last presentation, an Expo is held at the school which is a good opportunity to show off the product.

The content of our first presentation described what we were going to do, who we worked for, how we would proceed, and which process model we were going to use. In the first and second presentations we had 20 minutes for presenting, which translates to 3 minutes and 20 seconds for each team member. After presenting there was 10 minutes time for questions from the sensors and the audience.

In the second presentation the goal was to have a more technical focus than in the first presentation. We chose to give a brief explanation of how our system works, how the subsystem works and what we had done up until that point. The presentation was wrapped up with the plan from that point on.

Between the second and the final presentations, students at the school organized an event called Space Night, where we volunteered to present our project and its background. Two weeks before the final presentation an Expo is organized by the school. This is a technological fair where the bachelor groups can showcase their projects. The fair is a good arena for technology talk, exchange of ideas and the development of technological innovations.

The final presentation is the final opportunity for us to show off our project and the product that we have made. The presentation lasts for an hour. 20 minutes is a sales part, the next 20 minutes is the technical part and the last 20 minutes is aside for questioning. On May 23rd, the main documentation is going to be delivered. The main documentation contains of this report and its appendixes, and accounts for fifty percent of our grade.

3.3.3 Working Hours

As the team used a scrum-based agile process, the weeks were divided into sprints of one or two weeks, where a week was preferred. In the period before the disciplines’ respective exams (total of 13 sprints) we had a core time from 09:00 to 15:00 every weekday except Tuesdays when we had other courses. This corresponded to a minimum of 24 hours a week. After Easter, Tuesdays became a part of the core time, which then corresponded to 30 hours per week.

The total amount of required working hours in the project was estimated to 264 hours before Easter and 240 hours after Easter, which means a total of 504 hours per team member and a total of 3024 hours on the whole team. Two sprints prior to Easter holidays were set aside for studying for exams. During this period, The mandatory presence was repealed.

3.4 Handling and Understanding the Requirements

The content of this section describes how we developed an understanding of the scope and context during the project.

The requirements has two fundamental categories: Project requirements and Product requirements.

3.4.1 Project Requirements

As students performing a bachelor project, the highest level of preconditions are given by the context from the university;

- **Members of the group.** USN requires that the bachelor projects are performed by groups of 4-6 students. We are 6 members from three different disciplines in our group.
- **Evaluation of the project.** The foundation of evaluation are three presentations at different stages in the process together with written documentation. The written documentation consists of both the report and its attachments. The written documentation contains chapters usually not included in product documentation written for a customer or client. These chapters are present in the report to fulfill criteria defined by the university.
- **Relationship to the commissioner.** The company that fulfills the role as commissioner in this project is KONGSBERG Defence & Aerospace (KDA), division Space and Surveillance (DSS). The three parts - the university, the team of students and KDA, has signed a contract that commits the parts to fulfill their roles. Appendix A.55. KDA provided a sensor and several external supervisors who has participated in weekly meetings for supervision during the project.

3.4.2 Product Requirements

All the requirements that is directly connected to the product belongs to the category product requirements. These requirements are functional and technical and have been divided into layers and origin. The hierarchy of this category is illustrated in figure 3.1.

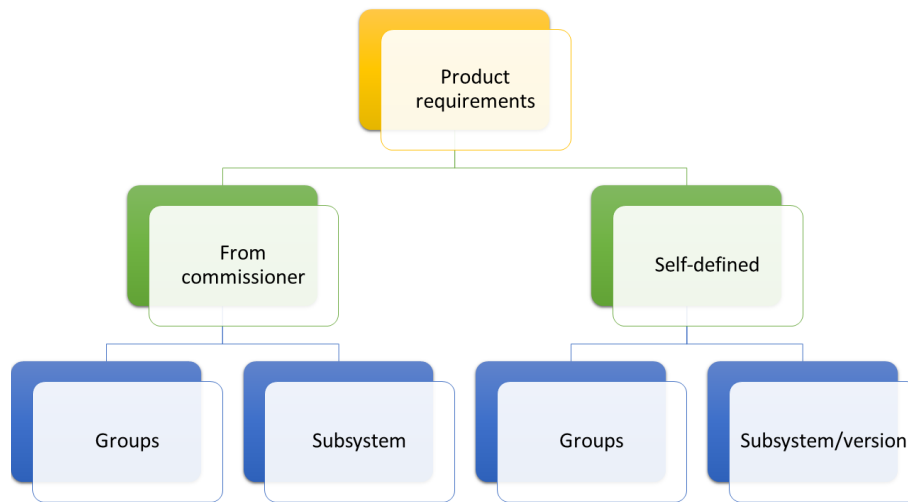


Figure 3.1: The product requirements hierarchy.

The main difference between requirements from commissioner and the self-defined requirements is that the written requirements from KDA concerns the SADM, and the self-defined concerns the DS. This is a product of the scope from the commissioner. KDA asked for a proof of concept for the SADM, and in order to include the software students in the group, KDA encouraged the team to expand the project's scope. Therefore, the team has developed all the requirements for the DS and the scope for this part of the project. These requirements were continuously validated with the commissioner over the course of this project.

As figure 3.1 shows, requirements were first divided into *groups* according to their category, sorted in a spreadsheet. As an example; Group 2, R2 - are "Functional requirements" and group 4, R4 - are "Interface requirements". The spreadsheet design was based on reasonable categories for different requirements, and aimed to standardize the template for every requirement.

The Requirement spreadsheet is attached in the appendix A.61, and the description for the spreadsheet with details about the information columns is explained in appendix A.60. The challenge in this process was to find a suitable template that was creating a universal understanding, ensure traceability and structure so that every member of the group could use it as a tool.

Later in the process, the team learned that in order to take advantage of Confluence, the tools' ability to implement and list requirements should be utilized as much as possible. Each requirement in Confluence is traceable back to the Requirement spreadsheet by a requirement ID. In Confluence, the requirements are categorized by *epics and versions*. Examples of epics and versions are "SADM - drivetrain", "DS - Software Layer version Beta 1" and "SADM - motor". This structure simplified dealing with requirement and enhanced traceability.

Using an agile development process required interpretation of requirements before rewriting them into user stories in Jira. Doing this made it easier to understand the context and function of requirements. The intended outcome was to use the user stories created for each requirement as subjects in the backlog and process one or several stories for each sprint. This process is illustrated in figure 3.2.

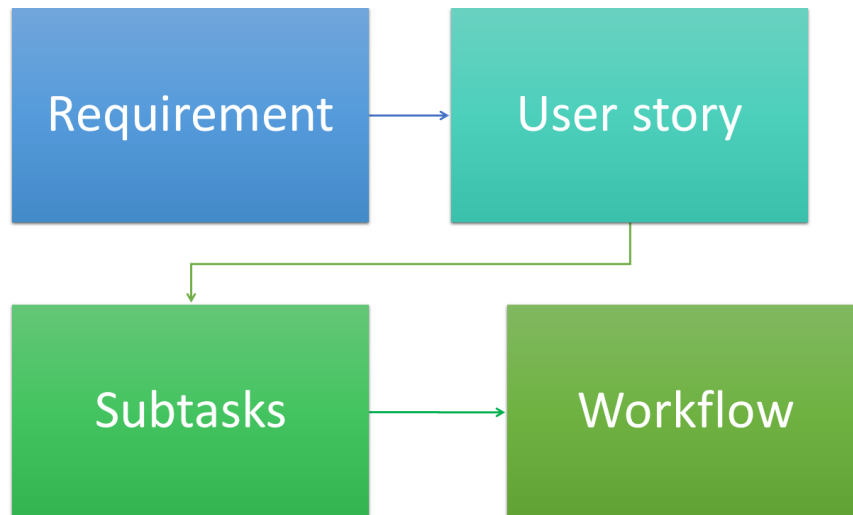


Figure 3.2: The intended flow for each requirement

The process of handling requirements illustrated in figure 3.2, succeeded for the requirements related to the Diagnostics system. For the project requirements given by the commissioner KDA, the team learned that they will not be able to adapt to this system of handling the requirements. The reason for this is that the team failed to divide the user stories and additional subtasks into tasks that were small enough to follow the fast iterative process model used during this project. Because it was difficult and time consuming to adapt the technical requirements for the SADM to this model at an early stage, the team decided to discard this method for the technical requirements.

Another related aspect is the frames given to the requirements for the SADM. The functional requirements (R2 - see appendix A.61) were set, but for many of the other requirements the teams' own assessments was to be taken into consideration. Which requirements this applies to are marked with colour in the Requirement spreadsheet. Many of the requirements are technically complex and heavily related to the space industry. Research and learning was necessary in order to evaluate if these requirements should be taken into consideration during the project. Incorporating these requirements into an agile process model and fast development failed at the start of the project. It should be taken into account that in the attempt to fulfill this flow, the team acquired knowledge and understanding of the prototype they designed.

After rejecting the requirement handling process for the requirements connected to the SADM-system, it was decided to frequently develop user stories not directly related to the requirements. The SADM-requirements were now processed in a more flexible way, illustrated in figure 3.3. The requirement analysis is inspired by [20, Chapter 2.3].

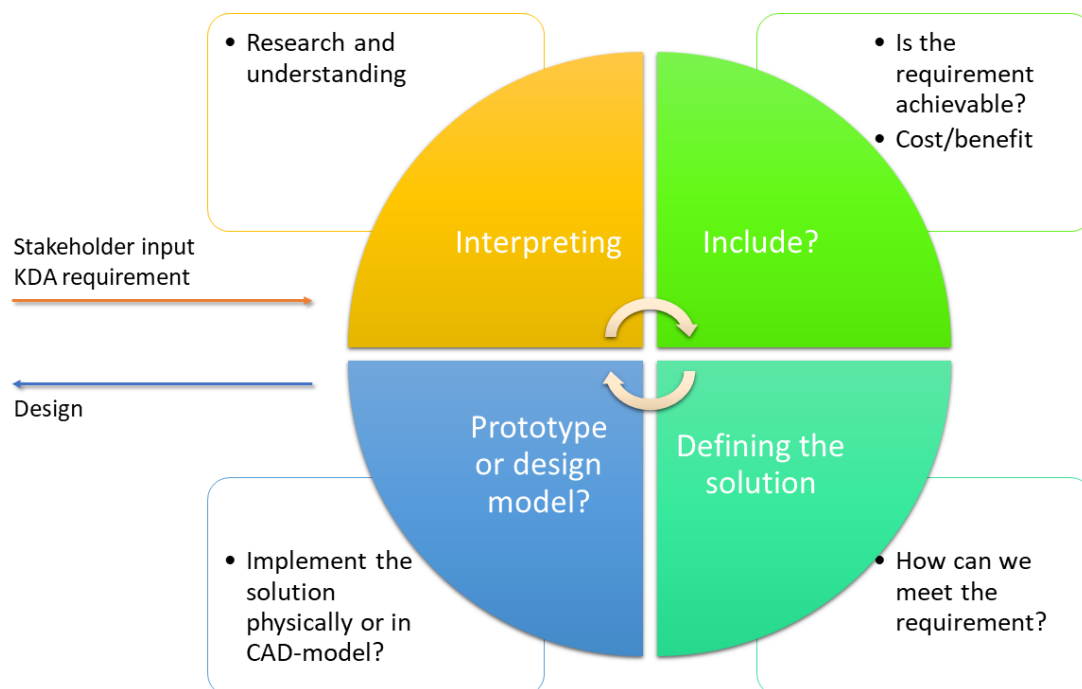


Figure 3.3: The updated requirement analysis.

3.5 The Process of Validation and Verification

These subsections will describe the validation and verification process in the project.

3.5.1 Validation

The definition of validation by ECSS' standard ECSS-S-ST-00-01C is the process that demonstrates that the product is able to accomplish its intended use in the intended operational environment. In other words, validation is the process where you ensure that you are making a product that works the way the customer intended, and that the requirements are covering the customer's needs. This is mostly done after requirement analysis and as a part of system development. The most certain way to do that is by communication with the main stakeholders, the customer or users of the product, which in the case of our project is DSS.

Since the design and production of our prototypes utilizes an iterative process, there were regular changes to the product. The weekly supervisor meetings were used to validate both the technical and academical work. Prototypes and redesigns were shown during the meeting to get feedback. Sometimes changes were done based on the feedback.

Other stakeholders we talked with was DSS' test engineers. They provided useful information on how they test, what they test and current state of the art. When it comes to the DS, talking with test engineers also confirmed that the team were on the right track when it comes to the development of the test environment. Later in the project, when the software layer of the DS was ready for testing, people who were not involved with developing of the DS tested the program. This revealed the user friendliness of the program, and changes were made according to the feedback.

The SADM is meant to be an off-the-shelf product when finished. Therefore, in order to make the product competitive in the market, it is important to create a design that is cheap to produce and easy to assemble to reduce the overall cost.

Talking with the machinists and vendors of machine parts gave us insight in how we can improve our design. An example is the structure of our second prototype, M2. A CAD-model was sent to Tronrud Engineering for a price estimation for 3D-printing of the part in titanium. The response from Tronrud Engineering was that the model was not optimized for 3D-printing, which made it unnecessarily expensive. Thinking this was over budget, even after optimization, the designers started to redesign and consider other production methods immediately. Discussing this with the external supervisor, made it clear that it was not too expensive if we have good arguments for the original design and why we should 3D-print. This validation revealed that DSS does not necessarily want the cheapest possible product as long as the choices we make can weigh up for the price.

Spending one day a week at KONGSBERG Innovation Center gave us the opportunity to use HoloLens and to create physical prototypes for validation rapidly. When changes were made to the design, the HoloLens was used to get acceptance from our external supervisors before making a physical prototype.

The most frequently used method for validation turned out to be showing and discussing around the physical prototypes and CAD-models. This proved to be the most secure way for communication as it is nearly impossible to misunderstand functional physical parts. Details about the prototypes are available as appendices.

3.5.2 Verification

This term is defined in ECSS-S-ST-00-01C [21] as a process which demonstrates through the provision of objective evidence that the product is designed and produced according to its specifications and the agreed deviations and waivers, and is free of defects.

When the requirements are set and the work to fulfill the requirements is done, the work or the system needs to be verified. One or several verification measurements are created to ensure that the work that has been done, is done correctly. This means ensuring that the system is functioning or performing as intended, documents and drawings are right or the system is built correctly according to drawings or models. This will prove whether the system

meets the applicable requirements and whether it is capable of fulfilling its role during the mission life.

The process of verification begins with registering the requirement that created the work. This is done in the requirement sheet that is stored in our Google Drive. A requirement should be formulated in a way that makes it able to be verified. After that, the same requirements need to be imported to the Verification Management Document A.63, which is another sheet stored in Google Drive.

The purpose of the VMD is to register how the engineering team is planning to verify the work that was done to fulfill the requirement, and to make it possible to trace back to previously verifications. Some of the specifications in the VMD is what requirement it is related to, how the verification is to be performed and what kind of verification it is.

The verification measurements are categorized in five different methods

- **Test** - this is the most reliable way to verify the system. It can be used to demonstrate the functionality and the performance of the system, sometimes in the worst case conditions.
- **Analysis** - This is verification done by finite element method (FEM), simulations, calculations or other sorts of empirical evaluation. This is important when testing is not possible.
- **Review** - Reviewing can be proofreading of documents, or comparing solutions to drawings. This can be done by team members or a supervisor who is not involved in the work of making the document/drawing.
- **Inspection** Visually inspect that the solution is produced according to drawings or description in documents. This may be for example assembly of prototype or circuits.
- **Unit test** - This is a method for testing smaller segments of the code simultaneously or individually.

All requirements need to be verified. This means that every requirement creates at least one verification, but sometimes it creates several. An example is requirement R2-11 that states:

The SADM shall be able to oscillate or rotate around the X- axis in both directions (CW and CCW), with a total angular range of minimum +/- 180 degrees

This can be verified with simulations of a CAD-model and by testing of a physical prototype. Both methods had to be registered in the VMD where they had separate and unique verification-IDs, even when they were related to the same requirement. The right procedure for the engineering team was to verify with both methods if there is a CAD-model and prototype available, since testing of the prototype don't necessarily guarantee a correct CAD-model and vice versa. The VMD was also stated with how the verification is to be performed.

After the verification was done, the Pass/Fail-status were updated with the date for when it was done and the name of the person who did it.

3.6 Project Risk

The Risk management in this project is largely based on a scrum approach. This means the rapid iterative project sprints contributes to minimize risks by itself [22][23]. However, as this project is not entirely based on scrum and contains aspects that are difficult to iterate as rapidly, it also includes more traditional approaches. Nevertheless, no particular risk standards are used and is mostly defined by the team, while however, using inspiration from a number of different standards.

The risk management work in this project uses a risk matrix and a spreadsheet calculating the impact based on weights of affection. This is a qualitative risk assessment in contrast to a quantitative risk assessment [24]. Four of the five weights are fairly typical in project risk management. The weights chosen are project scope, project velocity, project cost, project quality and project credibility. Where the last of these are concerned with the fact that this is a student project and thus, it is evaluated on other terms, relative to commercial projects.

3.6.1 Risk Matrix

Risk Matrix							
Probability:							
Will most certainly happen	4	4	8	12	16	20	24
High chance of happening	3	3	6	9	12	15	18
Medium chance of happening	2	2	4	6	8	10	12
Low chance of happening	1	1	2	3	4	5	6
Impact:		1	2	3	4	5	6

Figure 3.4: Six by four risk Matrix

The risk matrix in figure 3.4 shows a the risk as a product of impact and probability. Low numbers means low risk and is colored green. The higher the numbers, meaning higher risk goes from green through yellow to red. The impact is scaled from one to six, while probability is scaled from one to four. The reason impact have a longer scale than probability is that the impact is considered a more important factor than the probability in this project.

3.6.2 Project Risks

Project risks are concerned with project management incidents that may negatively impact the project. This can either be an incident that impacts the project management or it may be a consequence of the project management. For most cases, it is a combination of both.

Risk ID	RISKS	Possible causes	Impact	Probability	Risk Product	Mitigation Action
RP-	Project Risks					
RP-01	One member quits	Illness, Lack of motivation, social conflicts	6	2	12	Encourage each other, show appreciation for the work that is being done
RP-02	Several members quits	Social conflicts	6	1	6	Encourage each other, show appreciation for the work that is being done
RP-03	One member does less work than a minimum of what is expected	Illness, Lack of motivation, social conflicts	3	3	9	Transparency is high valued in the project, communicatin is key
RP-04	Several members does less work than a minimum of what is expected	Illness, Lack of motivation, social conflicts	6	2	12	Meetings and retrospective
RP-05	Requirements misinterpreted by the team	Miscommunication, lack of knowledge	4	2	8	Research, Communications and meetings within the team and with supervisors
RP-06	Requirements interpreted differently within the team	Miscommunication	3	2	6	Research, Communications and meetings within the team and with supervisors
RP-07	A few less important requirements are not met	Neglectance, too little time, bad assumptions	2	3	6	Less important requirements may have to be accepted. If they are not acceptable they are important

Figure 3.5: Selected project risks

In figure 3.5, one can see a table of a few selected project risks. Project risks all have a risk ID starting with RP (risk project), followed by a number. The risks are in a chronological order, for when they were identified. For the complete risks table with additional information, see appendix A.64.

3.6.3 Technical Risks

Technical risks are concerned with risks, that may cause, and/or be caused by technical difficulties, late deliveries, insufficient preliminary work or calculations. As well as aspects regarding the usage of the products.

Risk ID	RISKS	Possible causes	Impact	Proba- bility	Risk Product	Mitigation Action
RT-	Technical Risks					
RT-01	Flex radiates too much heat	High current, miscalculated derating	2	3	6	Hard to test and verify
RT-02	Wires radiates too much heat	High current, miscalculated derating	2	3	6	Hard to test and verify
RT-03	Electrical connections fail due to heat	High current, miscalculated derating, bad soldering	2	2	4	Assure good soldering
RT-04	Flex is too stiff	Too high percentage of copper, miscalculated TS geometry	3	3	9	Adjust TS geometry
RT-05	Flex does not fit Twist Capsule	Miscommunication between electro and mech	4	3	12	Tight collaboration between electro and mech
RT-06	Motor has too little power	Too much resistance/torque, wrong ball bearings	2	2	4	Buy as powerful a motor as possible, use good bearings
RT-07	Motor is susceptible to emc noise	Bad shielding	2	3	6	Hard to test, use sufficient shielding

Figure 3.6: A few selected technical risks

Figure 3.6 shows the first seven technical risks. The structure and listing order of this table is equivalent to the project risks table in figure 3.5. Both tables, with all risks and additional information can be found in the risk tables in appendix A.64.

3.7 Additional tools

- **Cadence Release 17.2-2016**

Software package including Cadence CIS for schematic design, Alegro Padstack Editor for designing padstacks for PCB design, Alegro PCB Editor for designing footprints for PCB and for designing PCB [25].

- **GIMP 2.8**

Image editing tool. Used for logo design [26].

- **Google Calendar**

Calendar shared between the team members. Used for planning events and meetings. When there was time for presentations and meetings, Google Calendar was used to send out meeting invitations where the invited could accept or decline the invitation.

- **Google Drive**

Google Drive is a file storage and synchronization service made by Google [27].

The team used Google Drive for documentation storing and share documents. A folder was created and shared with all team members. An internal folder structure was created to optimize use of the service. The drive folder was segmented into sections relative to disciplines, tasks and other major components of the documents and documentation that was handled. Parts of the drive structure was made available to supervisors with a shared link. Built in apps included in google drive was used to create spreadsheet for requirement and verification handling, as well as recording time spent in the project.

- **Google Hangouts** In some instances there was a need to use Google Hangouts for supervisor meetings. It allowed for video conference with supervisors when there was need to communicate remotely.

- **Microsoft Hololens**

Hololens is a mixed reality device made by Microsoft [28]. This is a physical device, but libraries related to create content for the device was used in conjunction with Unity to create the MR prototype.

- **Overleaf**

Overleaf is an online LaTeX editor that encourage collaborative writing of various kinds of documents. [29] This report and document templates were made using Overleaf. The process master contributed a lot towards the structure of the report, as well as simplifying inclusion of appendices.

- **SolidWorks**

SolidWorks is a tool for working with CAD models created by Dassault Systèmes [30].

Used by mechanical students in the team to 3D-model the prototype designs. All three prototypes was designed using this software.

- **SolidWorks Electrical** Solidworks Electrical was used to develop the design and electrical schematics of the control cabinet and the power transfers.

- **Visual Micro [31]**

The hardware layer of the diagnostics system (section 4.10.2) is controlled by an Arduino [32] MCU [33]. These MCUs are typically programmed using the Arduino IDE [34], but in this case the Visual Studio IDE turned out to be the better option. The reason for this is related to some of the statements in the previous section, and are further described in section 3 of the "HWL decisions" document, appendix A.24. For Visual Studio to support development of applications targeting Arduino MCUs, the Visual Micro extension is required. This adds features to the IDE which makes it possible to upload programs to the MCU.

- **Visual Studio community 2017**

Visual Studio community 2017 is an IDE developed by Microsoft [35]. It is commonly used for many different applications, and supports various programming languages including .NET C# and C++ which are used in this project. Visual Studio allow for development of multiple projects in one solution [36], and as the diagnostics system (section 4.10) required two different projects, this feature made cooperation between the two much more efficient.

- **Unity**

Unity is a game engine and 3D renderer made by Unity Technologies [37]. The tool was used to create the virtual prototype and the MR prototype.

Chapter 4

Technical Work

4.1 Prototypes

The main purpose of a prototype is to achieve proof of design and proof of product. This is a major part of verification and validation as the prototype will demonstrate the functionality and performance of the product. Several prototypes were made to test different functionalities or to validate the concepts. Each prototype was named Mark, followed by a number, to give every version of the prototype a unique designation, for example Mark1, Mark2, Mark3.

None of the prototypes were made with space grade materials, as this is very expensive and common material with similar properties will serve the same purpose in the case of a prototype. A CAD-model based on space graded material will be used to ensure that the design will meet all the requirements with proper material. This model is also used for simulations and analysis.

The prototypes have also shown to be useful in discussions, inspirations and for the collective understanding of problems and solutions.

4.2 Concept Development

During the conceptual phase, Lego was used for inspiration to develop ideas for different solutions. Lego parts were used that matched the main components of SADM to meet the requirements of the components, i.e shaft, motor, and twist capsule. Bearings are the only main components that was left out to simplify the model. It was considered unnecessary to include this because bearings are always mounted on the shaft. Lego solar cells were used to simulate movement, although the solar cells are not part of the product.

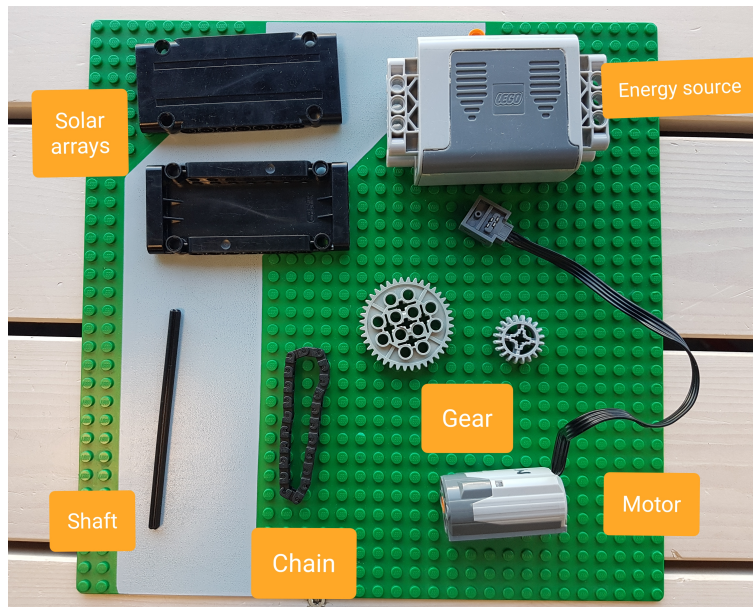


Figure 4.1: Lego components

Different variations of setup was explored such as several gears, motors or shafts as well as having the motor perpendicular to the shaft. They all had their advantages and drawbacks that are reasoned in appendix A.3. Looking into all these configurations was helpful in deciding the right design for the product.

The product goal is a solar array drive mechanism that is meant to be compatible with any other cubesats in a certain size, an off-the-shelf product. To be competitive in the market, the product had to be lightweight, reliable and affordable. Based on these requirements, the configuration Motor parallel to the shaft was chosen as the proper design. This seems to be the configuration with the highest reliability and the lowest weight due to few components.

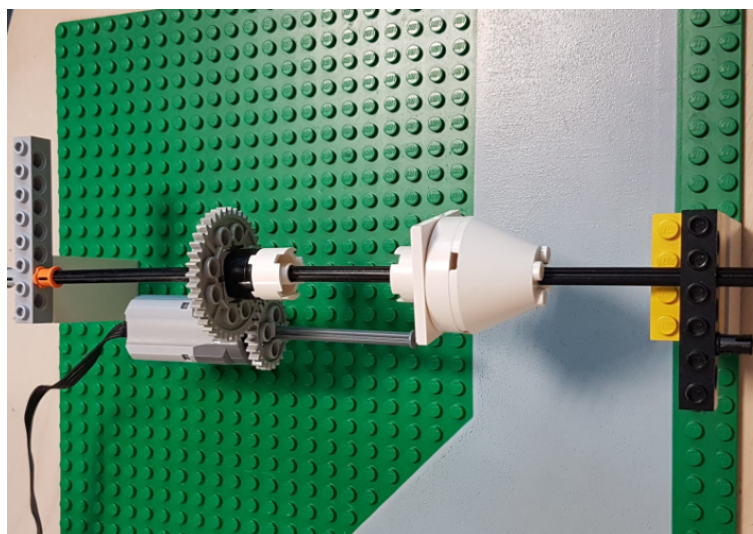


Figure 4.2: Motor perpendicular with shaft

Using Lego was a simple way of proving the functionality of the concept, but it did not confirm if it was possible to fit all the components in the limited space inside the SADM. A drawing was made in 1:1 ratio in addition to building Lego. This was made to get an idea of how to utilize the space inside the SADM as the space of 2U is only 200 x 100 x 100 mm. The dimensions of the components were only a rough estimation, and slightly oversized to demonstrate a worst case scenario of placement inside the SADM envelope.

The drawing became a great way of illustrating where all the parts would fit, because of the limitations on how large the parts could be. The building of Mark2 was based on this drawing.

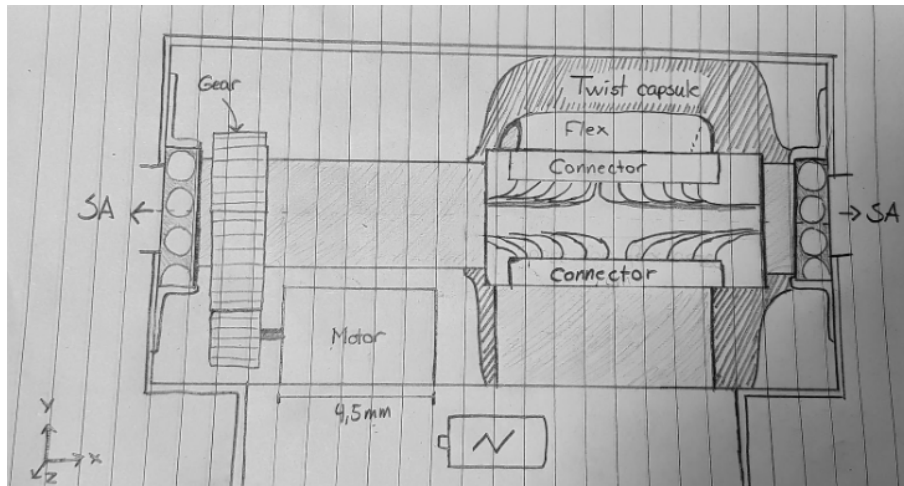


Figure 4.3: Drawing of the inside ratio 1:1

4.3 Concept Model

A concept model is a prototype that represents the product in a simple way. Concept models are usually not in a one to one size, but advantageous in easily explaining and understanding the concept. The important thing when it comes to size is that the relationships between the parts are consistent. The main function of the concept model is to use it for validation and to help stakeholders and other people to understand the project.

A misunderstanding at the beginning of the project led us to believe that the SADM should be part of an 8U Nanosat. After showing the model of an 8U Nanosat, it was clarified that it is in fact the solar array that is in the size of 8U, meaning that the satellite itself is 16U.



Figure 4.4: Concept model of 8U (left) and 16U (right)

This will affect the mass and the center of gravity for the whole satellite, which are crucial parameters for tests and simulations. The doubling of size of the solar array will also affect the moment of inertia and finally affect the torque in the system and power of the motor.

By using a concept model for validation at an early stage, a few mistakes were ruled out and some requirements were clarified.

4.4 Mark1

The first prototype version was made in time for the first presentation. This prototype was inspired by the concepts that were displayed during the first external supervisor meeting. Requirements regarding the size and functionality of the unit were the main focus while designing the prototype in Solidworks. Mark1 was made to demonstrate the concept, and requirements regarding weight and strength were not considered.

The design started with the envelope using measurements dictated by the requirements, followed by the shaft. All the parts, except for the bearings, are based on materials and production methods that were available at the workshops at USN, making it possible to produce the parts without relying on a third party.

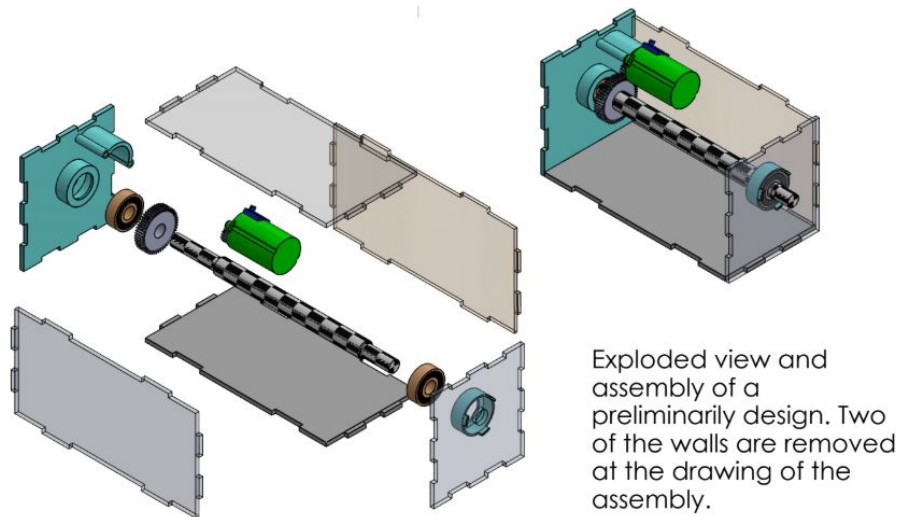


Figure 4.5: Mark1 exploded view

The housing was made in laser cut Plexiglas so that it was possible to see the components inside the unit. The shaft was made from a 16 mm aluminum shaft procured at the school workshop. Using a lathe, the diameter of the shaft ends was made smaller to allow the bearings to be fitted. The bearings was procured from Biltema which worked splendidly for the first prototype.

The gears were 3D printed in plastic, and in the size ratio of 3,25. A stepper motor was used to turn the shaft. An Arduino was used to control the motor with a program to turn the shaft 180° in each direction.



Figure 4.6: The inside of the physical prototype Mark1

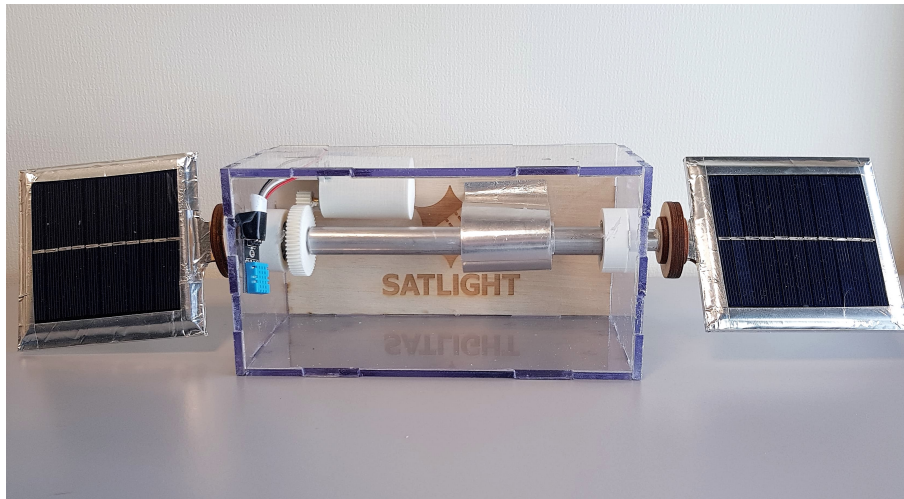


Figure 4.7: Mark1

4.5 Mark2

The second prototype, Mark2, was finalized before the second presentation. The design was based on an analytic process that identified the most efficient concept. The methods and techniques for identifying this concept is outlined further into this chapter.

The main purpose of the SADM is to rotate solar arrays so that they face the sun in the most efficient angle, and also rotate the solar arrays away from the sun when it is saturated with power. Analysis suggested that the most reasonable configuration is a SADM with one motor that runs the shaft with two spur gears in parallel. The shaft is hollow, so that the cables from the solar arrays are coming through the shaft and into the twist capsule where it will be connected to the flex. The power will then be transferred further into the battery in the satellite.

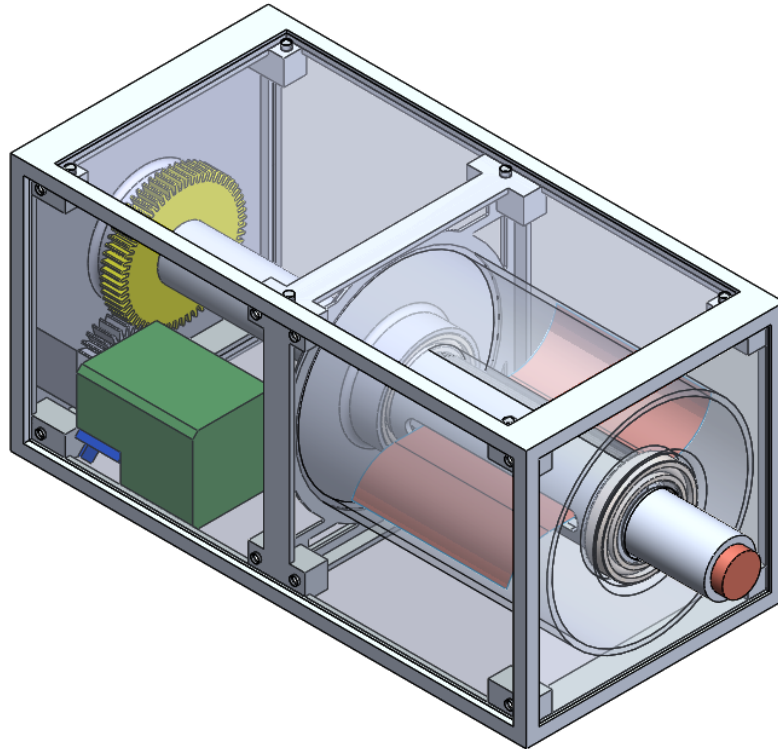


Figure 4.8: Mark2

The design of the gears in Mark2 is a modified version of the gears in Mark1. No calculations were made on either gears or bearings in this version. The motor is based on a Nema 11 Bipolar stepper motor that was considered for Mark2, but was not ordered. A casing was therefore made which made the previous stepper motor look like a Nema 11 motor.

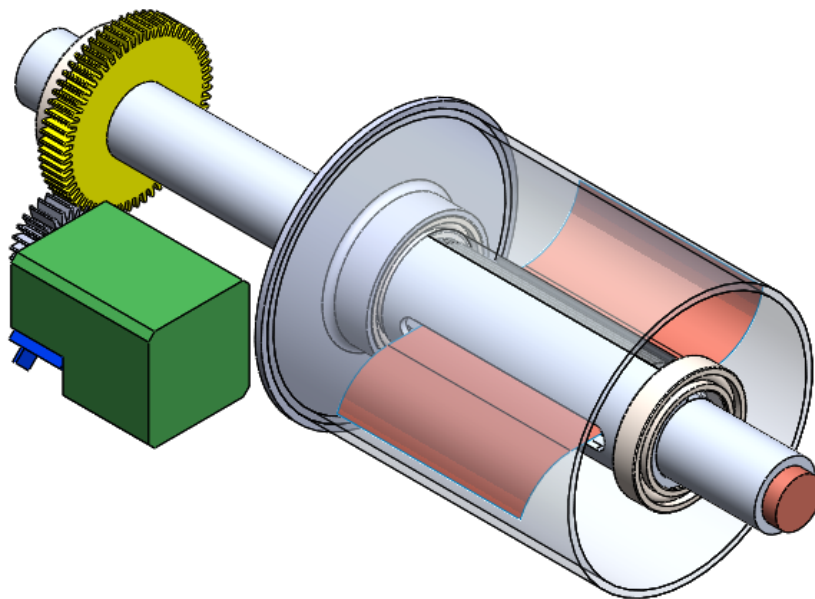


Figure 4.9: Mark2 without structure

4.5.1 The Structure

The structure of Mark2 was 3D-printed in plastic in one piece. This is a very expensive design if it is to be manufactured using conventional manufacturing methods to make the part in metal, such as milling. To make the structure in one piece by milling, one would have to start with a solid block that is at least 100x100x200 mm and then remove mass until only the shape of the structure is left. For aluminum 7075, which is a common type used for parts to aerospace, the initial mass of the working piece would be 5620 gram. After milling with a CNC-machine, the final part would have a mass of 146 gram, which is less than 3% of the origin. What makes it expensive to manufacture this part with milling is that the mass which is removed is considered waste, and it also requires a lot of time and labor to remove this amount of mass. 3D-printing in metal was therefore considered for Mark3.

External supervisors were highly interested in making the structure in one part, as this will completely remove the risk of assembling the structure in a wrong way. 3D-printing of the structure is also state of the art according to NASA's report about State of the Art of Small Spacecraft Technology [38]. Tronrud Engineering (TE) was therefore contacted for an estimation of price for 3D-printing the structure in metal, as they claim to be the leading manufacturer within additive technology in Norway [39].



Figure 4.10: The structure of M2 in the orientation it would have been printed.

The left side of figure 4.10 shows a plastic model of the structure that was used on M2 and a CAD-model of this was sent to TE. The response from TE was that the design was not optimized for 3D-printing. The design would require a lot of support material that needed to be removed afterwards, resulting in higher cost. This was known to the team, as 3D-printing in plastic is almost the same process as in metal.

The yellow areas in middle picture of figure 4.10 visualizes some of the support material needed to 3D-print the structure. The machinist suggested that supporting walls should be added 45 degrees from the vertical beams to support the horizontal beams during printing. The structure on the right side of figure 4.10 demonstrates what the original structure would have looked like with supporting walls. They could have been removed by milling after printing.

The cheapest metal TE could offer was titanium grade 5, Ti64. The estimated price for printing, annealing, removing support material and surface treatment was 25.000 NOK for the part in Ti64. The estimated price for the same part made with aluminum AlSi10Mg was 20% more. In comparison, the price for a space grade 2U-structure that needs to be assembled is 2950 euros (\sim 29.000 NOK) [40].

Redesigning the prototype to be optimal for 3D-printing could reduce the manufacturing price of the prototype, but the idea was discarded due to the fact that it would still be unreasonably expensive for this project. A new structure was therefore designed for M3.

4.5.2 Twist Capsule

One of the biggest challenges in design was the space restrictions, but also the design of the shafts inside the twist capsule. The twist capsule could not have a diameter larger than 100 mm because of the 100 mm x 100 mm cross section available, but there was also need for some space outside of the twist capsule so that wiring could fit in the space around it.

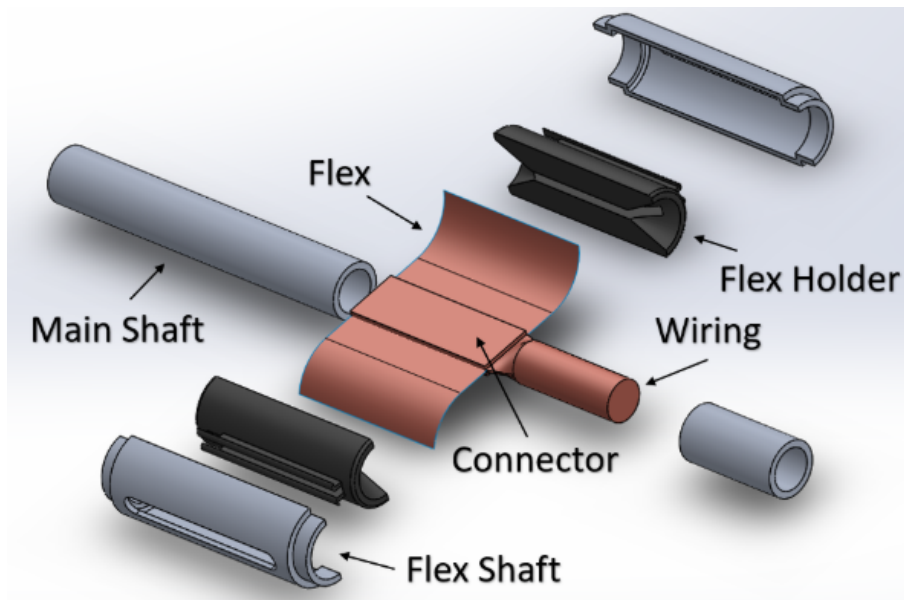


Figure 4.11: Twist Capsule components for Mark2

The power transfer and flex were the main focus when dimensioning the shafts. The size of the wires through the main shaft gave the foundation of how broad the diameter should be, but it was also important that it was not too wide so that there was as much space as possible for other components in the SADM. Inside the flex shaft the wires are soldered to the connector part of the flex, therefore it was important that this connector could fit inside the flex shaft with some margin, without increasing the total mass of the SADM or affecting the ratio to the capsule too much. Therefore, the dimensioning of the flex shaft became a trade-off between the flex, twist capsule and the total mass of the SADM.

4.5.3 Flex

The flex is a flexible printed circuit board (PCB). Flexible PCBs are used for a number of purposes in electronics, with or without components [41]. For transferring power from a rotor to a stator, such in this projects case, from the flex shaft to the capsule housing, it is possible to use wires. However, a flexible PCB have several benefits over wires. The main advantage of a flex, is the ability to perform its task for a longer period compared to wires, without fatigue. An additional advantage is weight. Wires with insulation, contributes to a greater mass, than a flex for the same cross section of electrically conductive material. This also implies that for the same mass, the flex is a much better conductor, and thus, is able to transfer more power than wires. The twist capsule consist of the flex shaft, flex and capsule housing which is the interface between the drivetrain and the power transfer line. As the flex shaft rotates, the flex moves inside the twist capsule with a relative unpredictable behavior, compared to strictly rotating parts.

4.5.3.1 Flex Length and Twist Capsule Geometry

The twist capsule geometry is largely determined by the length and behavior of the flex. However, the flex shaft and capsule housing are also determined by other factors, which in turn determines the length of the flex. It is therefore necessary to calculate all these parameters in relations to each other. A somewhat idealized geometric calculation was carried out to determine the parameters, such as the inner radius of the capsule housing r_c and the outer radius of the flex shaft r_s .

One concern about the behavior of the flex is that it should have as low of a curvature as possible. However, it should not be completely flat at any point and it should always curve, only in one direction, regardless of the flex shafts angular position. This is to ensure the least amount of mechanical fatigue for the copper traces inside the flex. Whether the flex is in a tight configuration or a loose configuration, depending on the angular position of the shaft, the points where the flex contacts or is mounted to other surfaces (the flex shaft and capsule housing), should be tangential to that surface. Before determining the radii of the twist capsule and the exact mounting positions for the flex, the length of the flex can be approximated in terms of three circle arcs.

$$l_f = L_{free} + L_c + L_s \quad (4.1)$$

Equation 4.1: Flex Length in terms of arcs

Consider equation 4.1. l_c refers to the arc where the flex contacts the inner wall of the capsule housing. l_s refers to the arc where the flex is wrapped around the flex shaft. l_{free} is the part of the flex in free space, meaning it is not touching any surfaces. Presumably, this is the largest possible arc that fits between the two other arcs. The reason for this assumption is that the stiffness of the flex will want to be as flat as possible, if the material don't contain any defects and if no other forces are applied to it. The two external forces acting on the flex is the capsule housing pushing the flex and the flex shaft stretching it.

In reality, a large section of the flex might tend to a shape, closer to that of a logarithmic spiral in most configurations. However, this approximation should be adequate for calculating the minimum length. An arc is a product of its respective radius and the angular difference of its endpoints. Assuming that l_{free} is as large as possible, and that the endpoints touch both the flex shaft and the capsule housing tangentially, the endpoints will be half a revolution apart and the distance between them is $r_c + r_s$ and thus the radius is $r_c + r_s$ divided by two. expanding equation 4.1 in terms of $l = r\theta$ we get the following equation:

$$l_f = \pi \left(\frac{r_c + r_s}{2} \right) + r_c(\theta_c - \theta_{co}) + r_s(\theta_s - (\theta_c + \pi)) \quad (4.2)$$

Equation 4.2: Flex length in terms of radius-angle products

In equation 4.2, r_c , r_s and θ_{co} are parameters, meaning they will be fixed values when the SADM is assembled. θ_s is the point where the flex is mounted to the shaft. Thus, θ_s varies depending on the shafts angular position, which must have a total angular range of at least one revolution.

All of the terms in this equation, must always be positive, and the angles should be relative to the flex. In other words, if the mounting point of the flex to the shaft θ_s is turned a number of revolutions relative to the mounting point of the flex to the capsule housing θ_{co} , then the number of revolutions must be included in the difference of these angles. The point where the flex goes from contacting the capsule housing, to not contacting the capsule housing (θ_c), as well as the point where the flex goes from contacting, to not contacting the shaft ($\theta_c + \pi$), are both abstract angles, that are not fixed relative to the flex.

As the flex shaft rotates and the flex moves, these abstract angles will move as well. They are always half a revolution in relation to each other. However, they will not move by the same amount as the shaft. Their movement depends on the ratio of the outer radius of the flex shaft r_s and inner radius of the capsule housing r_c . Note that if the point θ_c is less than the capsule housing's mounting position θ_{co} , the term l_c is negative and thus the flex is no longer mounted tangentially to the capsule housing. In this configuration the flex is too tight. Likewise, if the flex shaft's mounting position θ_s is less than $\theta_{co} + \pi$, the term l_s is negative, and therefore the flex will not be mounted tangentially to the flex shaft. In other words the flex is too loose.

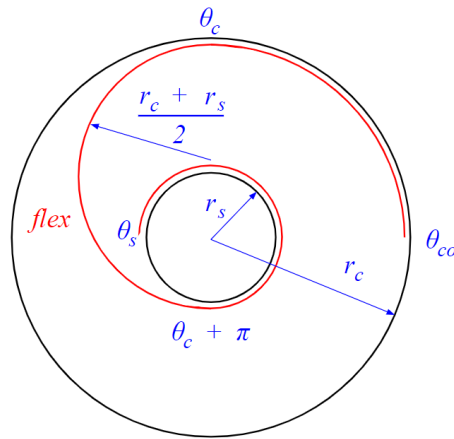


Figure 4.12: Relationship between minimum flex length, shaft radius and capsule radius

Figure 4.12 illustrates the geometry of the twist capsule and the mathematical terms for each of the three arcs.

As an example, assume that variable angle θ_{co} is zero, meaning the flex is mounted at the right side of the capsule housing. Furthermore, the contact point between the flex and the capsule housing θ_c is $\pi/2$ or at the top of the capsule housing, 90° from the mounting slot. This implies that $\theta_c + \pi = 3\pi/2$. Furthermore, assume that the mounting point of the flex shaft $\theta_s = 3\pi$, meaning that it is at the left-hand side of the flex shaft rotated one and a half revolutions, compared to the origin. The radius of the capsule housing r_c is $40mm$ and the radius of the flex shaft r_s is half as big, $20mm$. Inserting these values into equation 4.2 gives us the following flex length,

$$\pi \left(\frac{40mm + 20mm}{2} \right) + 40mm \cdot \left(\frac{\pi}{2} - 0 \right) + 20mm \cdot \left(3\pi - \frac{3\pi}{2} \right) = 80\pi \approx 251mm \quad (4.3)$$

Equation 4.3: Example: flex length for arbitrary parameters

Given the values in equation 4.3, we can get an idea of what will happen if the shaft is rotated. The important thing is that θ_s should be within a 2π range, without any of the terms becoming negative.

Given that the flex has a constant length and that l_{free} is constant, then the sum of l_c and l_s is also constant. This implies that if we turn the shaft clockwise (thereby loosening the flex) to the point where $l_s = 0$, meaning $\theta_s = (\theta_c + \pi)$, then the new l_c is equal to the sum of the old $l_c + l_s$. We can then solve for $(\theta_c - \theta_{co})$.

$$\begin{aligned}
l_f &= \pi \left(\frac{r_c + r_s}{2} \right) + r_c(\theta_c - \theta_{co}) \\
r_c(\theta_c - \theta_{co}) &= l_f - \pi \left(\frac{r_c + r_s}{2} \right) \\
\theta_c - \theta_{co} &= \frac{l_f - \pi \left(\frac{r_c + r_s}{2} \right)}{r_c} \\
\frac{251mm - \pi \left(\frac{40mm + 20mm}{2} \right)}{40mm} &= \frac{5\pi}{4}
\end{aligned} \tag{4.4}$$

Equation 4.4: Solving for $\theta_c - \theta_{co}$

Because of the constant $\theta_{co} = 0$ then $\theta_c - \theta_{co} = \theta_c = 5\pi/4$ and because $\theta_s = \theta_c + \pi$, then $\theta_s = 9\pi/4$. Likewise, we can set $l_c = 0$ and solve for $\theta_s - (\theta_c + \pi)$

$$\begin{aligned}
l_f &= \pi \left(\frac{r_c + r_s}{2} \right) + r_s(\theta_s - (\theta_c + \pi)) \\
r_s(\theta_s - (\theta_c + \pi)) &= l_f - \pi \left(\frac{r_c + r_s}{2} \right) \\
(\theta_s - (\theta_c + \pi)) &= \frac{l_f - \pi \left(\frac{r_c + r_s}{2} \right)}{r_s} \\
\frac{251mm - \pi \left(\frac{40mm + 20mm}{2} \right)}{20mm} &= \frac{5\pi}{2}
\end{aligned} \tag{4.5}$$

Equation 4.5: Solving for $\theta_s - (\theta_c + \pi)$

Since $\theta_{co} = 0$ and $\theta_c - \theta_{co} = 0$, then $\theta_s - (\theta_c + \pi) = \theta_s - \pi = 5\pi/2$ and $\theta_s = \pi + 5\pi/2 = 9\pi/2$. The difference of the extreme values of θ_s shows whether the range of θ_s is equal to, or greater than 2π . The range must be larger than 2π in order for the flex to be of adequate length.

$$\begin{aligned}
\Delta\theta_s &= \theta_{s(max)} - \theta_{s(min)} \\
\frac{9\pi}{2} - \frac{9\pi}{4} &= \frac{9\pi}{4} > 2\pi
\end{aligned} \tag{4.6}$$

Equation 4.6: Flex length in terms of radius-angle products

Equations 4.4, 4.5 and 4.6 show that this particular case contains adequate values for a design. However, there are too many parameters for the equations by themselves to be efficient. Also it is useful to check and compare different parameter values fast and also add some buffer length at the ends, as well as taking the connectors at the end into account.

FLEXCULATOR			
Shaft radius [mm]	20		
Capsule radius [mm] {must be <100}	40		
Ratio (shaft/capsule) {must be <0,1>}	0,5		
	[revolutions]	[2 pi]	[radians]
Capsule mounting angle	0	6,28319	0
Capsule kissing angle	0,5	6,28319	3,141592654
Shaft kissing angle	1	6,28319	6,283185307
Shaft mounting angle (tight)	2,25	6,28319	14,13716694
Shaft mounting angle (loose)	1,25	6,28319	7,853981634
Shaft mounting angle (zero)	1,75	6,28319	10,99557429
SMA(l) - SKA {must be positive!}	0,25	6,28319	1,570796327
Buffers length and connectors		Shaft	Capsule
Flex length [mm]	251,3274123	20	20
			TOTAL:
			291,33

Figure 4.13: Flexculator

Figure 4.13 shows a spreadsheet type calculator for adjusting the parameters and calculating the flex length. It contains the same parameter values as mentioned, as well as the length of the rigid parts. It also calculates the ratio of the two radii r_s and r_c . Another feature of the flexculator is a built in warning function. If one attempts to use invalid number combinations, a cell will detect the error and switch color. If a cell contains a marginal error, it will be coloured yellow. If a cell contains an absolute error, it will be coloured red.

Examples of errors include:

- The radius of the capsule housing r_c or the flex shaft r_s is larger than 50 mm, so that the twist capsule is larger than the SADM.
- The flex shaft radius r_s is larger than capsule housing r_c .
- The flex is too short and will not be able to rotate a full revolution.

4.5.3.2 Flex Width

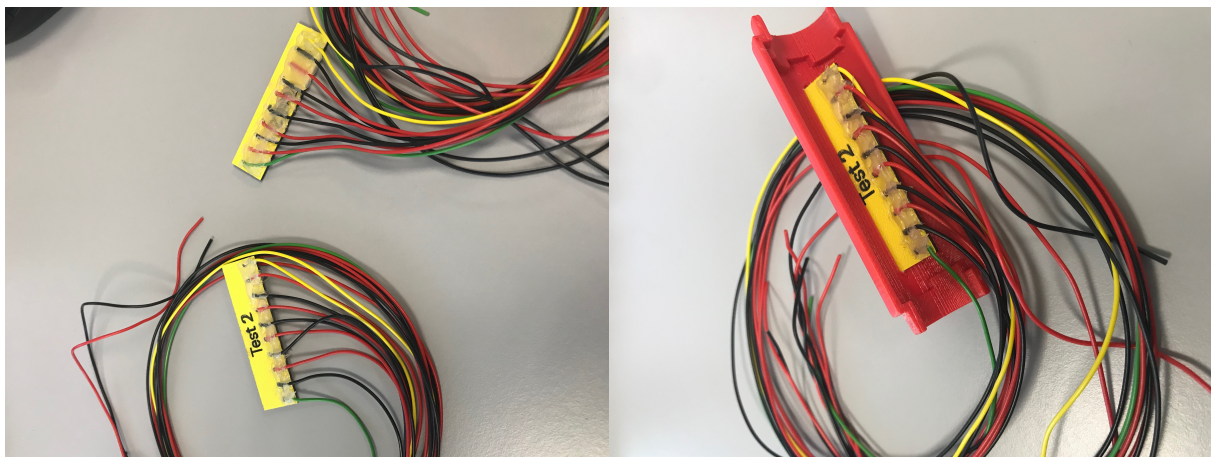
While the minimum length of the flex and its mechanical behavior is determined by the radii of the twist capsule, the minimum width is first and foremost determined by derating according to the maximum current, in other words it is determined the cross section of copper required for transferring 20A. The upper bound is constrained by the space provided for each subsystem in the SADM. Since the width of the flex attributes to only one spatial dimension, which also happens to be the dimension with the most room available, this is less of a concern than the length of the flex. This is applicable, despite the twist capsule not sharing the room with any

other subsystems, in the other two dimensions. That is, those of which the length occupies. Furthermore, if it is practically sound, more flexes can be added to provide a larger combined width, and thus, a larger copper width, while limiting the actual width of each flex.

The main concern with having more flexes is that the size of the flex shaft inside the twist capsule must not only be wider than the flex itself, but it will also have a larger radius in order to fit the rigid part and their electrical and mechanical mounting connections for each flex. This ultimately affects the length and behavior of the flex. In order to decide the best compromises for the physical dimensions of the flex, In appendix A.9 there is one colour coded comparison table for the width and one colour coded table for the length of the flex based on the data from the flexculator.

4.5.3.3 Rigid prototype

One of the required properties for the shaft and flex shaft was to have space for the wires and the interface between flex and the wires. To verify this, the team used laser engraving to produce a very first prototype of the rigid part of the flexible PCB. The rigid part prototype had the same measurements as the design, and also the correct number of holes to replicate drill holes for soldering the wires. See figure 4.14. The wires were glued instead of soldered.



(a) Interface between flex and wires.

(b) Inside flex shaft.

Figure 4.14: The interface between flex and wires must fit inside the flex shaft.

From this prototype, the team learned:

- The shaft diameter can be 20 mm, should be at least 10 mm in order to have space for the wires that goes out to the solar arrays.
- The slot where the flex enters the flex shaft should be redesigned to prevent an aggressive angle on the flex where it enters. See section 4.6.5.1. The slot opening should have angled inner walls that will provide less mechanical stress on the flex. In other words the flex should be guided into the flex shaft as tangential as possible.

- If the slot opening is moved 45 degrees as illustrated in section 4.6.5.1, it will be easier to mechanically attach the rigid part inside the flex shaft. The method of attachment can be glue or by screws.
- Soldering is the process that is most suitable for fastening this amount of wires. By increasing the number and use thinner wires would have caused disadvantages by making the soldering hard to perform by hand. Less, thicker wires would not have been flexible enough for the small space, and the wires or soldering could have been damaged over time.
- Protective Earth (PE): The flex leads PE for both the solar arrays and the redundant PE for the flex shaft and shaft. Redundant PE can be provided by attaching the rigid part mechanically to the flex shaft using screws, but if attainments are achieved with glue, this can be a challenge due to PE.

For visual inspiration, the flex was replicated in copper tape. The copper tape in figure 4.15 shows the copper tape flex implemented with the flex shaft.



Figure 4.15: Copper tape flex.

4.5.3.4 Prototypes of Flex Design

For early flex designs, we used one ground trace and two solid copper planes, one forward and one return, rather than multiple traces. The larger the number of traces, the larger the number of spaces in between, resulting in less area being used to transfer power. Copper planes then, is the most efficient use of the area of the flex.

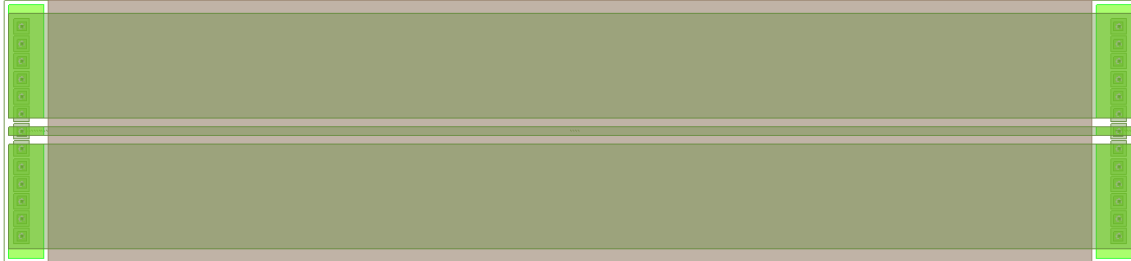


Figure 4.16: Flexible PCB with no mechanical connectors

Figure 4.16 shows the first flex design. It is produced using OrCAD PCB Editor. [25] The design is as simple as possible and only essential elements are included in the design, such as the flexible area, two rigid areas, one at each end of the flex, spanning the same width as the flexible area. Also, two copper planes with one grounding trace in the middle running along the length of the flex. The rigids at the ends are stacked with as many via-connectors that could fit along the width of each rigid. No exact calculations were made for this design. The length was roughly approximated with the flexculator and the width of the flex including the planes was approximated by the flex width comparison table. The connectors used are from one of the software's own libraries.

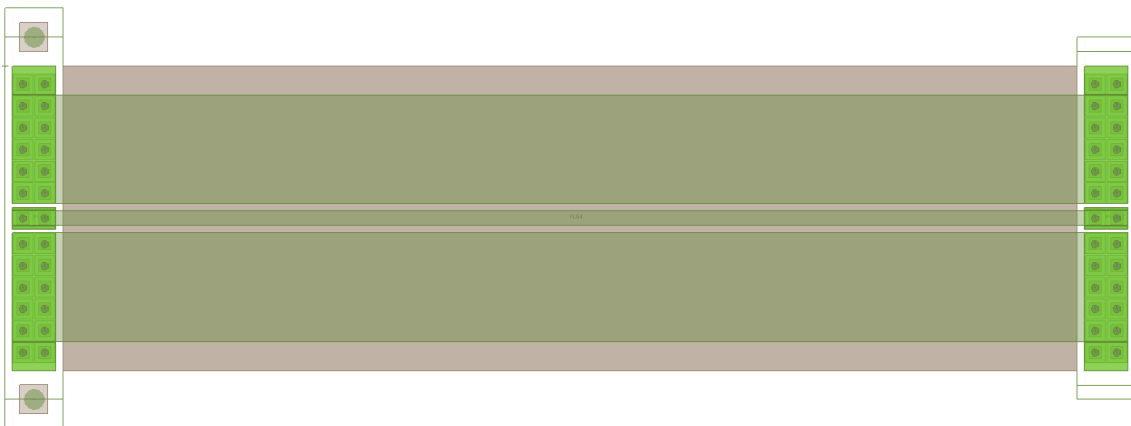


Figure 4.17: Flexible PCB, widened connectors

The first modification of the first flex design adding more soldering points and a wider rigid with mechanical connection points for the outside of the capsule housing. See figure 4.17

4.6 Mark3

Mark2 provided the basis for design of Mark3 and the placement of components that were needed. Mark3 is an improved version of Mark2, with focus on production methods, that it should be easy to assemble and it should be light weight. This is the last prototype that was built and it will represent the final design of the SADM, that is named M3-Space. There will still be some minor differences between M3 and M3-space, such as materials, components and production method, due to the expensive space components.

The subsections below will explain the changes that were made and the process of the design. Technical 2D-drawings are available as appendices.

4.6.1 Drivetrain

The design of the drivetrain was heavily influenced by Requirement R4-05 that states: "The maximum current transfer in the power lines shall be 20A forward and 20A return; - 10A each direction each solar array." In other words; there was a need to transfer a fair amount of electricity through the SADM. Another requirement R2-11 states: "The SADM shall be able to oscillate or rotate around the X axis in both directions (CW or CCW), with a total angular range or minimum + 180 degrees".

Based on these requirements the main tasks for the drivetrain was to rotate and transfer power.

4.6.2 Motor

The following subsection will describe decisions and considerations taken in conjunction with the choice of stepper motor. The content of this section is inspired by *Embedded Systems Circuits and Programming*, [42, Chapter 15].

The motor's function is to transform electrical power and signals, to rotate and position the shaft and solar arrays. The motor is a stepper motor from Sanyo Denki and was selected by analyzing related requirements and by guidance from OEM Motor's sales office.

The key parameters for selecting motor:

- **Functional:** The motor's operating sequence is rotating slowly to a given angular position. The motor must be strong enough to rotate the shaft, solar arrays and additional equipment.
- **Physical:** The motor must be as small and compact as possible to not seize physical space and weight, and yet strong enough to be able to perform as required.
- **NEMA-standard:** If the motor follows the standard NEMA dimensions, the motor can easily be replaced by a motor suitable for operation in space.

- The motor should have an operating voltage around 17-24VDC. This is because the team was told by the commissioner that they can consider about 17-19VDC supply voltage in a satellite, and the level of voltage can have an impact on the motor's performance. The voltage level 24VDC is standard for traditional components and therefore the best alternate in order to find available components with an acceptable cost.
- Power is a limited resource in a satellite, and it is therefore desirable to keep the power consumption low. Due to stepper motors complexity, the best way of verifying this is to measure the power consumption with maximum load at the highest angular velocity required.

The background for the use of stepper motor in this project is the requirement R4-08 that states: "The stepper motor electrical shall have the following nominal electrical interface:

- 2 phase, bipolar communication with 1.8 degrees full step angle
- Phase resistance: 3.5 ohm/phase
- Phase inductance: 1.2 mH/phase

Stepper motors have a discrete output as a product of pulses on each winding controlled by a motor driver. For the application described by this report, where the desired output movement is a specific angle in an open-loop configuration, a stepper motor is perfect.

NEMA stands for National Electrical Manufacturers Standard and their homepage [43] says:

"A standard of the National Electrical Manufacturers Association defines a product, process, or procedure with reference to one or more of the following:

- *Construction*
- *Dimensions*
- *Operating characteristics*
- *Performance*

... "

OEM's sales office claimed in an e-mail 15.03.19, appendix A.8, that the motor is according to NEMA 17, which the team has not managed to verify. A requirement review with external supervisor concluded that most important part of the requirement R4-09: "The motor interface shall be according to standard NEMA dimensions." is that the flange should be standardized so that it can be replaced without further corrections. The team learned that information of this importance should be verified before the component is ordered. According to NEMA's standard

NEMA ICS 16 - Industrial Control and System - Motion/position control motors, controls and feedback devices [44, Chapter 4.1.2.3], table 4, indicates that a NEMA 17 motor (where 17 is the flange number) indicates that the flange is square with 1.7 inches side walls. Since 1.7 inches equals to 43.18 mm and the size of the flange for our motor is 42 mm, therefore is this requirement considered not to be fulfilled.

The motor is controlled by a motor driver, which is a part of the DS. The motor driver regulates the behavior of the motor, such as step size, speed, current, acceleration and more. The motor driver and parameters are discussed further in the chapter of Diagnostic System, 4.10.2.2.1.

In order to receive the motor in time for testing, it had to be ordered before the mechanical design was ready. Based on an early concept design, the team could estimate approximately torque needed to rotate the solar arrays with following equation

$$\tau = I\alpha \quad (4.7)$$

Equation 4.7: Torque in rotational motion, a function of inertia and angular acceleration

The inertia I was calculated to be 0.0023 kgm^2 based on the design of Mark1, as if it was made with aluminium. Since there was no formal requirement regarding angular acceleration α from the commissioner, this was set to one radian per second. This is over exaggerated, as it equals to 57 degrees per square second, and the requirement regarding our angular speed was up to 5 degrees per second. With this data, the torque is

$$\tau = I\alpha = 0.0023 \text{ kgm}^2 * 1 \text{ rad/s}^2 = 0.0023 \text{ Nm} \quad (4.8)$$

Equation 4.8: Calculation of torque with estimated inertia and angular acceleration.

Later on the project, simulations of the final design revealed that the estimated inertia was 18 percent off from the correct value. A thorough elaboration of the calculation is available in the appendices.

The only powerconsuming component in the SADM is the motor. The motor's power consumption was first tried carried out by calculations based on figure 4.18 and equation 4.9 [2, Chapter 5.2].

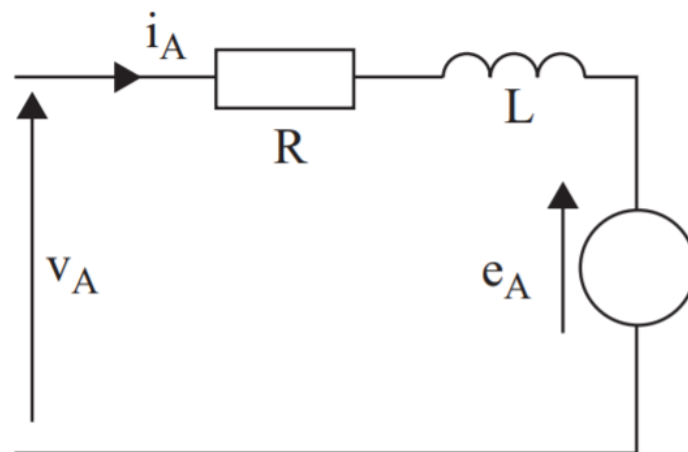


Figure 4.18: Circuit model of one phase in a stepper motor. [2, Chapter 5.2]

$$V_A = RI_A + L \frac{dI_A}{dt} + e_A \quad (4.9)$$

$$e_A = \frac{d\Psi_A}{dt} = \omega \Psi_M \cos(\omega t - \delta) \quad (4.10)$$

- e_A : As the motor rotates, there is induced voltage in the phase windings, often referred as back-EMF, represented as e_A (A for circuit A - bipolar motor has two circuits, A and B). Higher speed leads to higher induced voltage and lower pull-out torque because this voltage will cause the rotor resist its own angular velocity.
- Ψ represents the magnetic flux linking the permanent-magnet with the phase windings. This flux varies with the position of the rotor as it changes due to rotation. Ψ_M represents the maximum flux. As equation 4.10 indicates, the voltage induced, e_A , depends on the changes in the flux.
- ω : for angular velocity.
- δ : The load angle accounts on the rotor's lag - displacement - behind the phase equilibrium due to load on the motor. δ is a constant calculated by integration of load angle.

As generally known, the power is calculated by $P = U * I$. The motor driver, who's internally circuit is unknown, decides the voltage level applied on the windings in order to keep the current in the motor's windings at a certain level. The total power consumption depends on both the motor and the motor drive, and it is therefore more accurate to measure the power in each winding under operation.

4.6.3 Gears

To get the main shaft rotating using the motor, we used power transmission via spur gears. The reason for choosing these types of gears was that they are easy to produce with a high quality result, they are the most widely used gear types, and they are readily available.[45]

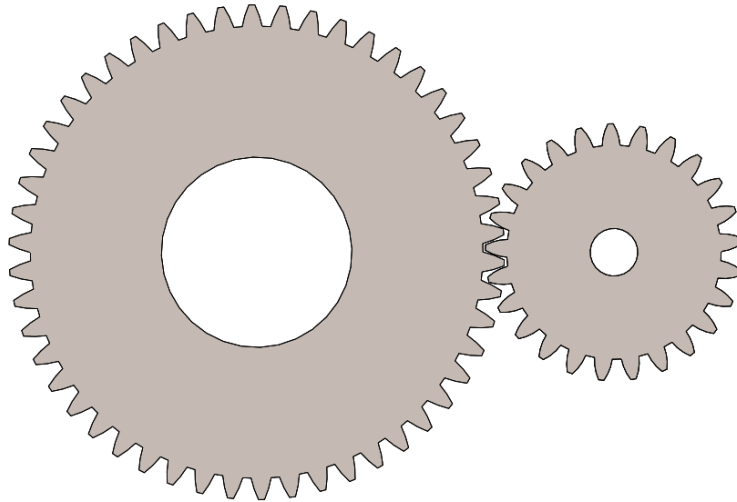


Figure 4.19: Gear Assembly

The gears in Mark1 and Mark2 were gears that were self made, drawn from scratch. The new gears in Mark3 were taken from the Toolbox function in SolidWorks, and chosen based on available space between the main shaft and the envelope walls of the SADM.

4.6.4 Bearings

As there are rotating components inside our SADM, there was need for bearings to support the shafts while they could still rotate. Without a bearing, a lot of power would be wasted on overcoming friction between surfaces. As early as Mark1 we had bearings in the prototype. These bearings were bought from Biltema, and was just an improvisation to demonstrate that the shaft would rotate.

In the design of Mark2, bearings was extracted from the Toolbox function in SolidWorks as there are a library for bearings from SKF available. A 61805-2RS1 bearing was bought from Kulelagerhuset AS [46]. This bearing is a single row deep groove bearing and some of the advantages with these are [47]

- Low starting and running friction, except at very high speeds
- Ability to withstand momentary shock
- Accuracy of shaft alignment

This is considered to be suitable for this use, as the speed is less than five degrees per second and shock is possible.

As for Mark3, the 61805-2RS1 bearing was also used. For further information see the Design Description A.4.

4.6.5 Twist Capsule

The twist capsule is a part of the solution for transferring power over a rotational axis. In order to protect the wires from damage caused by moving mechanical elements - shaft - another solution than leading power through wires is necessary. The twist capsule solution is inspired from the Karma-series, which is a part of DSS' product portfolio.

Most of the components of the twist capsule did not change diameters, although there were some adjustments made on the design of the flex shaft and likewise with the short and the long main shaft. The splitting of the flex shaft proved to be challenging in terms of composing the components.

4.6.5.1 Flex Shaft and Main Shafts

A large part of the challenges faced from a mechanical standpoint was to create a system of parts that made it possible to implement electronics in the easiest way. Developing the Flex shaft inside the Twist Capsule was one of the parts that was the most time-consuming. The function of this part was to keep the wires coming from solar arrays connected to the flex in place.

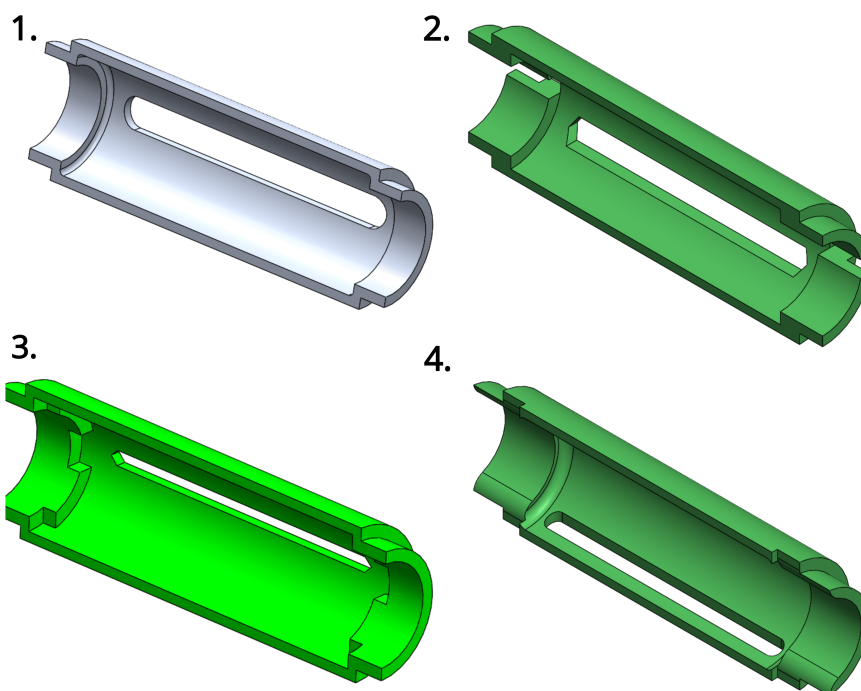


Figure 4.20: Versions of Flex Shaft

An early plan for connecting the main shafts and the flex shaft was to connect the three parts by friction welding, this would not make it easy to connect the wiring and the flex together. Already in Mark2, a decision was made to split the Flex Shaft, as this would provide better access to connect the electronics on the inside. As shown in Figure 4.20, there are four versions that were considered. In order to verify the various versions, a collaboration was made with the Mechanical Lab at KDA. After most of the visits, small changes had to be made to the design, making it easier to manufacture the parts, as these would be manufactured with a combination of milling and lathing.

The slot, or cut, in the side is where the Flex will go through, and on the inside it was attached to a connector which connected the Flex and the wiring from the solar arrays. Initially this slot was placed at the very middle of the flex shaft, but after some time a decision was made to not place the slot in the middle of the Flex Shaft, but rather place it further downwards in the Y-direction. The reason for this being that the connector would get more space if it was placed at an angle. Another reason was that the flex should not get a sharp bend at the end, but rather go tangential with the Flex Shaft.

The inside and outside of the flex shaft has changed and developed in parallel with the main shafts, as the flex shaft can be considered a part of the main shaft. Figure 4.21 corresponds to Figure 4.20 in what version of the main shaft belongs to which version of the flex shaft.

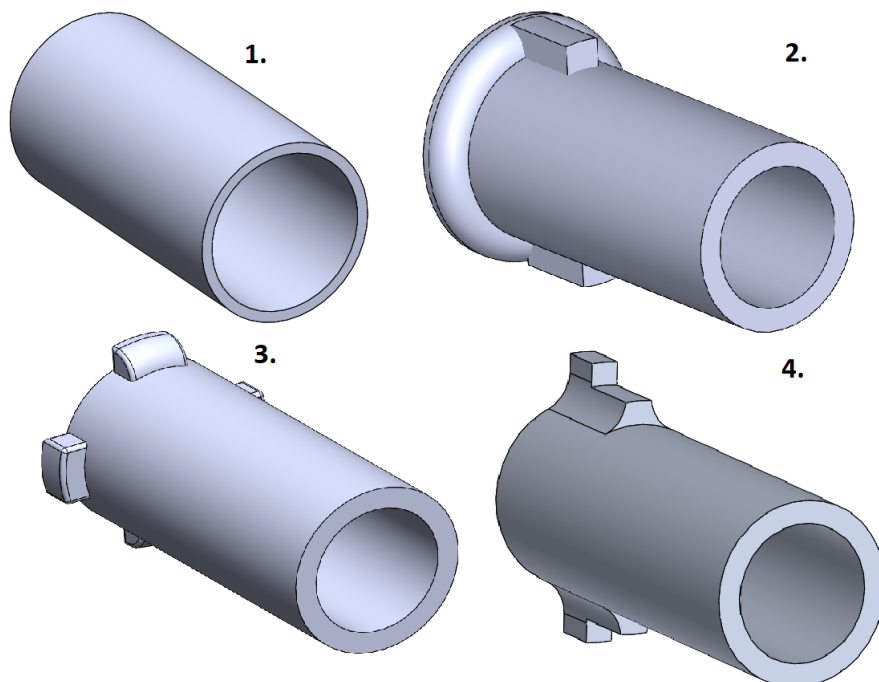


Figure 4.21: Main Shaft Versions

The very first design of the main shaft was a straight simple hollow shaft, but was further developed with ways of connecting to the flex shaft. In version no. 2 illustrated in Figure 4.21 there was added a ring and flanges at one end of the shaft. Since the flex shaft has a larger diameter than the main shaft, there was need for locking the rotation of the main shaft

tangentially and it had to be prevented from moving axially as well. Design no. 3 was given the name "Cross shaft" and the thought behind it was to lock the main shaft tangentially in the flex shaft, though this design did not prevent movement axially.

The Cross shaft proved to be difficult to produce, due to the small size of the grooves where the main shafts were supposed to be attached. And after consulting mechanical workshop the design needed to be redesigned once more. This revision resulted in the final design, number 4 in Figure 4.20 and 4.21.

4.6.6 Power Transfers

The power transfers shall bring the power from the solar arrays through the rotating axis and through the SADM for distribution in the satellite. The main challenge in this part of the system is transferring power over a rotating axis in a reliable way.

The power transfers can be configured in many ways. It was preferable that the solution was flexible, since the configuration of solar arrays nor the circuit configuration in the satellite was described in the scope. Several power transfers in parallel will achieve this flexibility. Some solar arrays has several cells, which can be wired in different ways: in series to increase voltage, parallel to increase the current or a combination. The requirements for further power distribution, such as battery charger and type of batteries, inside the satellite decides which configuration that is best.

When choosing the number of forward and return wires, the following aspect were considered:

- One large lead for each direction (forward and return current) will force the configuration to have one circuit for each solar array wing. Having only one circuit for each solar array wing will not be optimal, since a short circuit would be critical. The battery charger inside the satellite is presumably secured with fuse(s), which will cause the whole solar array wing to be out of order and the power loss will be at least 50% due to one of two solar array wings is down.
- One large lead is stiff and hard to handle. The limited space inside flex shaft could cause damage to the wires.
- In order to make the flex flexible, the flex should either way have several tracks. Interface between several tracks and one wire can be difficult inside flex shaft.
- Too many wires to each solar array wing would cause practical difficulties, as shown in figure 4.14 and discussed in section 4.5.3.
- The European Cooperation for Space Standardization's standard for derating [48] describes a reduction factor for wires in a bundle. The more leads, the lower this factor becomes, which can cause challenges according to the inner diameter of the shaft.

- 2 · 4 leads for each solar array - total 2 solar arrays - is the chosen design. This configuration follows from the interface between SADM and satellite to the flying leads.

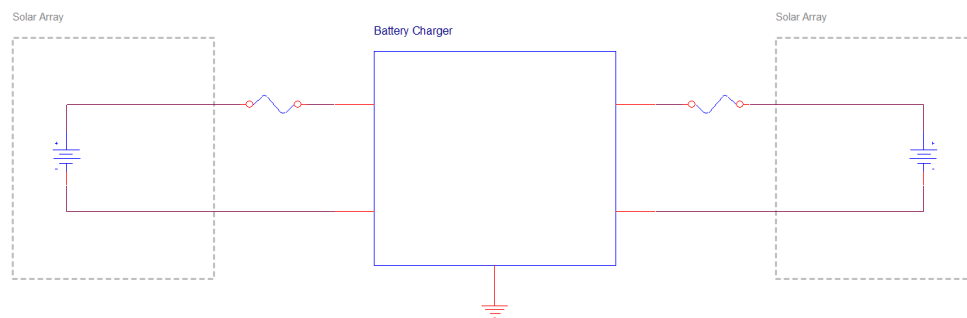


Figure 4.22: Configuration with one, large lead each direction for each solar array.

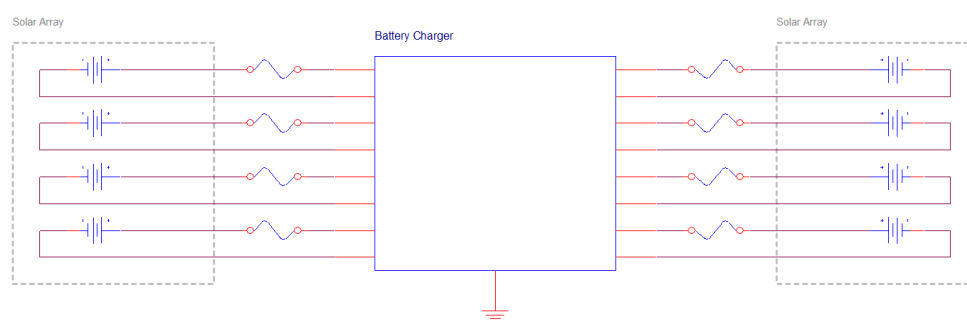


Figure 4.23: Several alternatives are achievable by having more leads to each solar array.

Figure 4.22 shows that by choosing just one lead to each solar array wing, even if the solar arrays are configured in series or parallel, there can only be one circuit for each solar array, that will break in case of failure due to short-circuit. Four leads to each solar array wing in the other hand, allows the configuration to be divided in until four circuits.

Protective earth (PE): The requirement specifications from the commissioner describes that the grounding (PE) between shaft and stator side shall be redundant, see R4-06 in appendix A.61. When designing the flex shaft and the interface between leads to solar array wing and flex, there was a challenge due to this requirement, as discussed in section 4.5.3. The best solution for redundant grounding is to connect a lead between flex shaft and capsule housing or other part of the stator side. This can be achieved in several ways;

- Connect the grounding lead with screw inside flex shaft.
- Soldering the lead to the flex shaft. The flex shaft is made of aluminum, which will require a specific soldering process.
- The track for redundant grounding in the flex should be connected to the mounting holes internally in the flex's rigid part. The team did not manage to fulfill this design of the flex.

The team decided not to fulfill the requirement for redundant grounding, as the best way to mechanically fastening the flex inside the flex shaft was with glue. With the right type of glue, the glue can function as both insulation between the power transfers and mechanical attachment.

Further technical information about the design of power transfers, connectors and grounding is described in Design Description A.4

4.6.7 Flex

The final flex design was designed with the OrCAD software package, as the Mark 2 prototypes. However with the addition of using Cadance CIS to draw the schematic design before designing the flex in PCB editor. The advantage of this, was to output the a netlist, from the schematics, making it simpler to use several traces, rather than solid copper planes. The reason for having traces rather than solid copper planes like before, is that a flex with several traces are more versatile, meaning that it could be used different purposes, than intended. A flexible PCB should have traces that are more than five times the copper thickness to prevent fatigue and traces breaking. However, because of the relative large, required current capacity, the traces should be fairly wide anyway. The final result ended up with eight, 5 mm traces, and two, 2.5 mm traces for Solar array ground and SADM ground.

4.6.7.1 Scematical Drawing

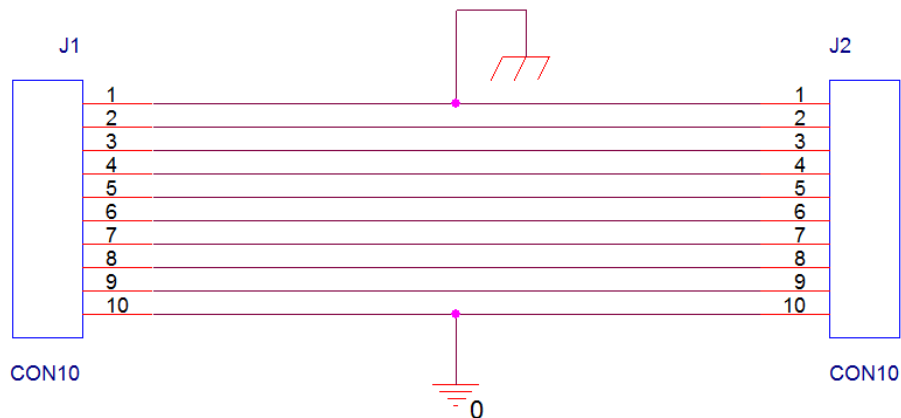


Figure 4.24: Flexible PCB, second iteration

Figure 4.24 show the schematic of the flex. It consists of two ten-pin connectors connected together with parallel wires. The schematic show two different grounds to distinguish solar array ground and SADM ground in the PCB design. However, there is no particular meaning behind the usage of the two different ground symbols, except for distinguishing them in the netlist and footprints, which are used connect the traces, to and from their respective solder pads in the correct order.

4.6.7.2 Padstacks and footprints

The padstacks were designed in Alegro padstack editor, with pad dimensions approximately, according to level A IPC-2221 standard [49]. This essentially means that the size of the pads (copper surrounding connector), drill holes for the connectors should be based on the diameter of a pin or in our case, the tip of a wire, to make it simple to solder and to assure a good solder joint.

The footprint for the two connectors were designed in Alegro PCB editor, using the padstacks mentioned above. They were placed 6 mm apart from another to fit the predetermined dimensions of the flex. The designs of the connectors were modified by the PCB producer to simplify the production Although the spacing remained the same. For more detailed information about the dimensions, see appendix A.4, Design description 7.4 (table 6).

4.6.7.3 PCB Layout

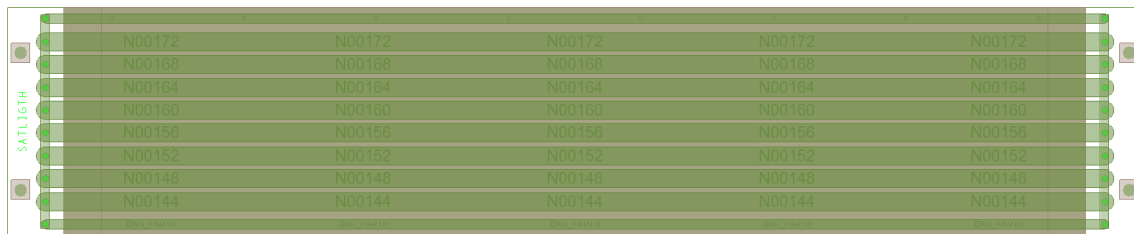


Figure 4.25: Flexible PCB, second iteration

Just like the footprint for the connector, the PCB layout were designed in Alegro PCB editor. Figure 4.25 show the teams flex design, which were submitted to the producer. After communicating with the producers and being recommended, to let them modify the design to simplify the production, a decision of letting the producer modify parts of the design. However, the teams main interest were kept and important parameters based on requirements, especially regarding derating were not altered by the producers.

4.7 Space Environment Effects

In the chapter below we will address space environmental hazards that must be considered when dealing with space technology. The most part of the information below is cited from NASA.[50]

Space is a rough environment, and for materials and components that go out into space there are many threats that need consideration. Our SADM and its 16U satellite will be out in the Low Earth Orbit (LEO) which is defined as 200km to 1,000km above the Earth's atmosphere. We also have a requirement - R8-01 that states: "The SADM shall have a minimum operational life time capability of 5 years in space for 600km Sun-synchronous orbit (SSO) application". Out here there are hazards that degrade materials, such as vacuum, solar ultraviolet (UV) radiation, charged particle (ionizing) radiation, plasma, surface charging and arcing,

temperature extremes, thermal cycling, impacts from micrometeoroids and orbital debris, and environment induced contamination. Materials are an important aspect of designing space components, and we needed to figure out which materials are best suited for space environment.

4.7.1 Vacuum

Vacuum is a quantum state with the lowest possible energy, generally it contains no physical particles which also means that there is no pressure. In LEO or more precisely 600 km above the atmosphere the pressure is 10^{-6} to 10^{-9} torr. Materials used in space are subjected to outgassing due to the high-vacuum environment. Outgassing can include evaporation and sublimation and is the release of gas that was trapped or dissolved in the material during the time on earth. Emitting of these gases can cause to contamination of other materials if it gets absorbed by other materials nearby. Some materials will also have reduced mechanical properties. It is therefore very important that the materials have very low outgassing rate.

4.7.2 Atomic Oxygen

Oxygen comes in several different forms. The oxygen that we breathe is called O_2 , it is comprised of two atoms of oxygen. O_3 is ozone, such as occurs in Earth's upper atmosphere, and O with one atom, is atomic oxygen (AO). Atomic oxygen can be very helpful in many ways, such as restoring art and is also used in helping human health [51]. But for components in space the atomic oxygen is not helpful, on the contrary it will degrade the materials. This degradation has the potential to compromise the performance and lifetime of missions with significant time in LEO.

4.7.3 Ultraviolet Radiation

Earth's atmosphere filters out most of the sun's damaging light, but materials in LEO bear the brunt of solar photon damage. While AO may bleach materials, UV generally darkens them. UV radiation needs to be considered when choosing materials in space.

4.7.4 Particulate or Ionizing Radiation & Hydrogen

The three main sources of charged particle radiation naturally occurring in space are galactic cosmic rays, solar proton events, and the trapped radiation belts. Similar to the UV damage, the particulate radiation can embrittle a polymer structure. Metals can also be embrittled by absorbed hydrogen to such a degree that the application of the smallest tensile stress can cause the formation of cracking.

4.7.5 Plasma

Plasma is the word given to the fourth state of matter (solid, liquid, gas, plasma). A plasma is a gas that is so hot that some or all its constituent atoms are split up into electrons and ions, which can move independently of each other. This Plasma can erode or change a materials thermal and mechanical properties.

4.8 Materials

Since our project deals with space technology and the parts are exposed to extreme environments as described in section 4.7, the parts must be able to withstand this in the best possible way. At the beginning of the project, the plan was to concentrate a reasonable amount on material selection and research on the use of composites in space context. After the second presentation, when we were to do the final considerations in materials, a decision was made to prioritize the design and production of the prototype, instead of using a lot of time on finding materials that suited our needs. Therefore, we chose to inquire with our stakeholders on what material was used in their products, and took inspiration from it.

By 3D printing most of the parts in Mark2, we used a lot of plastic. Or more specifically Acrylonitrile Butadiene Styrene (ABS), which is commonly used in LEGO bricks [52] and is a hard plastic. 3D printing gave the team an opportunity to verify whether the parts would actually fit together or if we would have to redo the design, which was the case in some instances. 3D printers were readily available, therefore it was frequently used so that verification could happen rapidly and therefore changes on the design was made all the time.

A few weeks before the documentation hand-in, the production of Mark3 began. Initially the final design of Mark3 was 3D printed in ABS plastic, but quite early in the project a decision was made that the parts would be produced in aluminum and it is also a preferable material recommended by the CubeSat Standard which states: "The use of Aluminum 7075 or 6061-T6 is suggested for the main structure. If other materials are used, the thermal expansion must be similar to that of Aluminum 7075-T73 (P-POD material) and approved by Cal Poly launch personnel" [53]. In the SolidWorks model, Aluminum 7075 was used on the panel walls since this is what was recommended from our stakeholders at DSS. The 7075 alloy is a high strength aluminum, and the mechanical properties are listed below.

Mechanical properties - Aluminum 7075-T6					
Temper	Density (g/cm ³)	Elastic modulus (GPa)	Yield Strength (MPa)	Tensile Strength (MPa)	Elongation (%)
T6	2,80	71,0	505	570	11

Figure 4.26: Aluminum 7075-T6 Mechanical Properties [3]

The tempering T6 means that the aluminum alloy is solution heat-treated and then artificially aged. the 7075 alloy is widely used in aerospace, military operations, marine and transportation industries, and the reason for this is because high strength and light weight properties are critical there [54].

For the production of Mark3 we were in contact with KDA's mechanical workshop. After some visits and changes in the design, it was made clear that they would be producing the shafts and the panel walls, but for the Right Lid, Middle Wall and the Left Lid we contacted WayKen Rapid Manufacturing Limited in China. Their company name did not disappoint; the parts were manufactured and delivered within two weeks.




Item	Part Name	Picture	Description
1	Right lid		Material: AL6061 Process:CNC Finishes:machining finishing
2	Wall B, middle 100x100		Material: AL6061 Process:CNC Finishes:machining finishing
3	Left Lid - M3		Material: AL6061 Process:CNC Finishes:machining finishing

Figure 4.27: Offer from WayKen

The description of the parts in Figure 4.27 states that they are produced in Aluminum 6061 so the parts are compatible for space according to the statement from the CubeSat Design Specification [53].

4.9 Finite Element method

As one of our methods of verification was analysis, we chose to use the Simulation feature of Solidworks to expose SADM to forces, vibrations, and thermal challenges. We got a number of requirements from our employer that were very much optional, but we chose some that we prioritized and these are included in this chapter. Note that the structure and some twist capsule components are not included, so the analysis are simplified and narrowed down to the drivetrain only, with fixtures on the bearings. In the subsections below there are results from a launch simulation, forces applied,torque and vibrations. All applied on a simplified drivetrain. Thermal analysis on the left lid to see how it affects the bearing.

4.9.1 Launch

To illustrate how much force the drivetrain would experience we used data from a rocket that is commonly used for CubeSats, which is Rocket Lab Electron. This launch vehicle has a maximum axial acceleration of -4 to $+8$ G [55]. In the analysis of gravity pictured in Figure 4.28, a 10G gravity force was used to illustrate where the critical point would be, the value 10G was used to give a tiny factor of safety to the results. The drivetrain is turned upside down to show where this critical point is, which is on the underside of the main shaft. Maximum Von Mises stress at this point is 0,86 MPa.

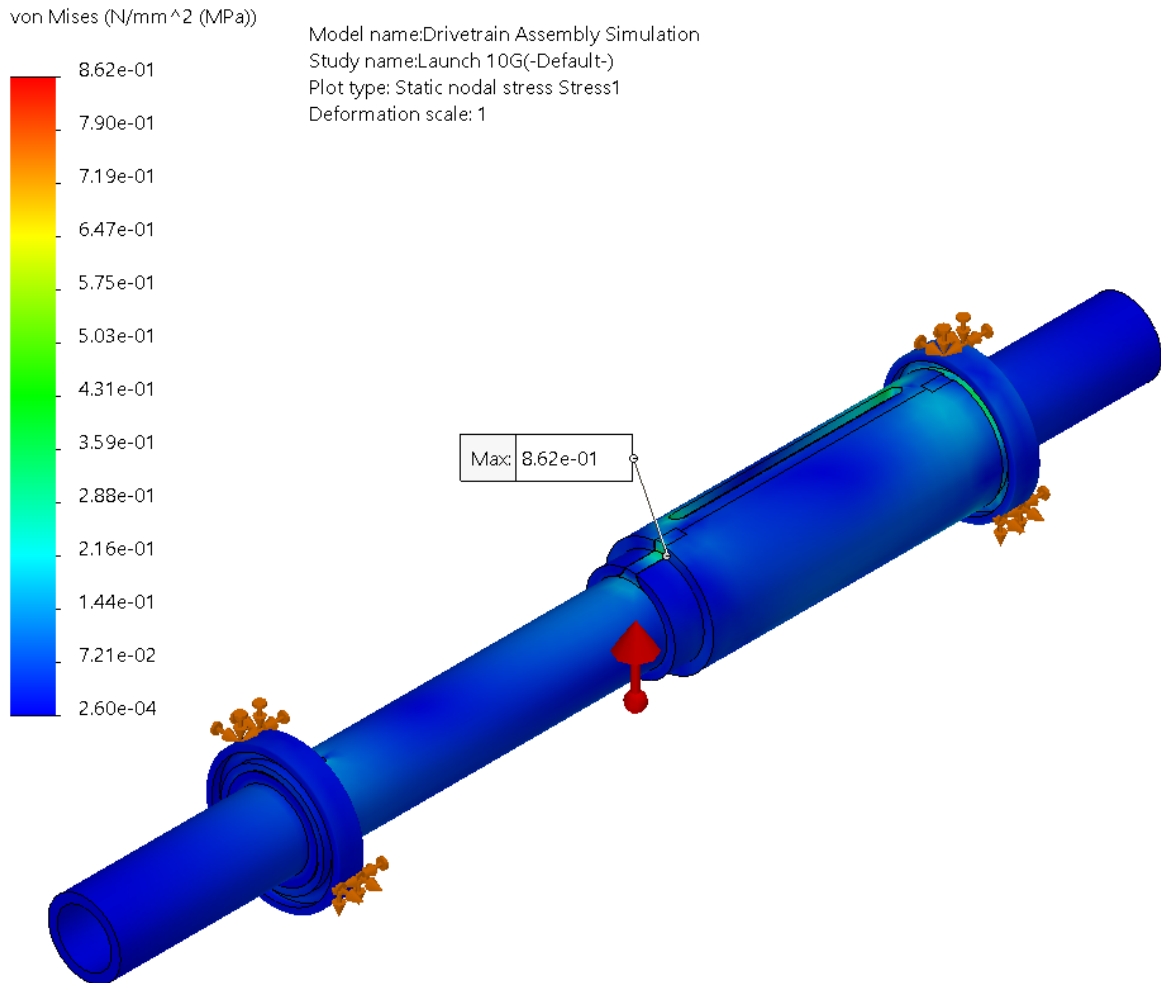


Figure 4.28: Launch 10G

4.9.2 Radial, Axial and Torsional Forces

Requirement R5-06 shown in Figure 4.29 gives us an indicator of which quasi-static loads the SADMs interfaces and drivetrain will be exposed to. To satisfy this requirement a number of tests were executed with FEM.

R5-06: The SADM shall be able to withstand the following quasi-static loads applied in the SADM solar array interface and with the SADM constrained by its normal screws in a rigid (infinitely stiff) spacecraft interface	
Maximum moment: Axial: +- 0 N Radial: +- 1500 N Bending: +- 300 N	Maximum radial and axial loads: Axial: +- 2000 N Radial: +- 1500 N Bending: +- 260 Nm

Figure 4.29: Requirement R5-06

In Figure 4.30 the Maximum Moment requirements forces and torque is applied to the drivetrain. The deformation scale is set to 6 so that it would be easier to see where the deformation would happen. Note that with deformation scale 1 (true scale) there was no visible deformation. As expected the solar array interfaces would be the place where most of the deformation would take place, though the Maximum stress was placed at the flange on the main shaft.

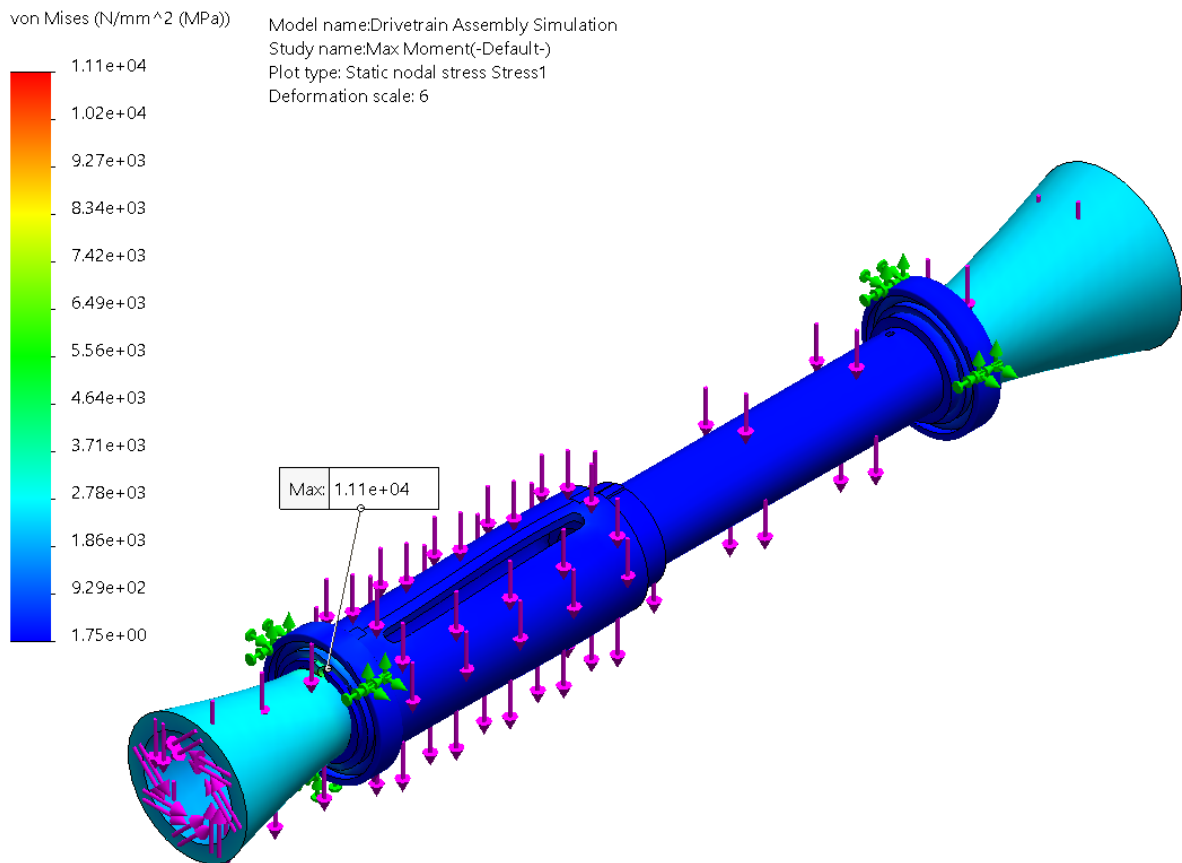


Figure 4.30: Maximum Moment

The second part of requirement R5-06 was based on maximum radial and axial forces, 1500N and 2000N respectively. Figure 4.31 illustrates the results in true scale. Maximum Von Mises stresses are at the same spot as with the torque result.

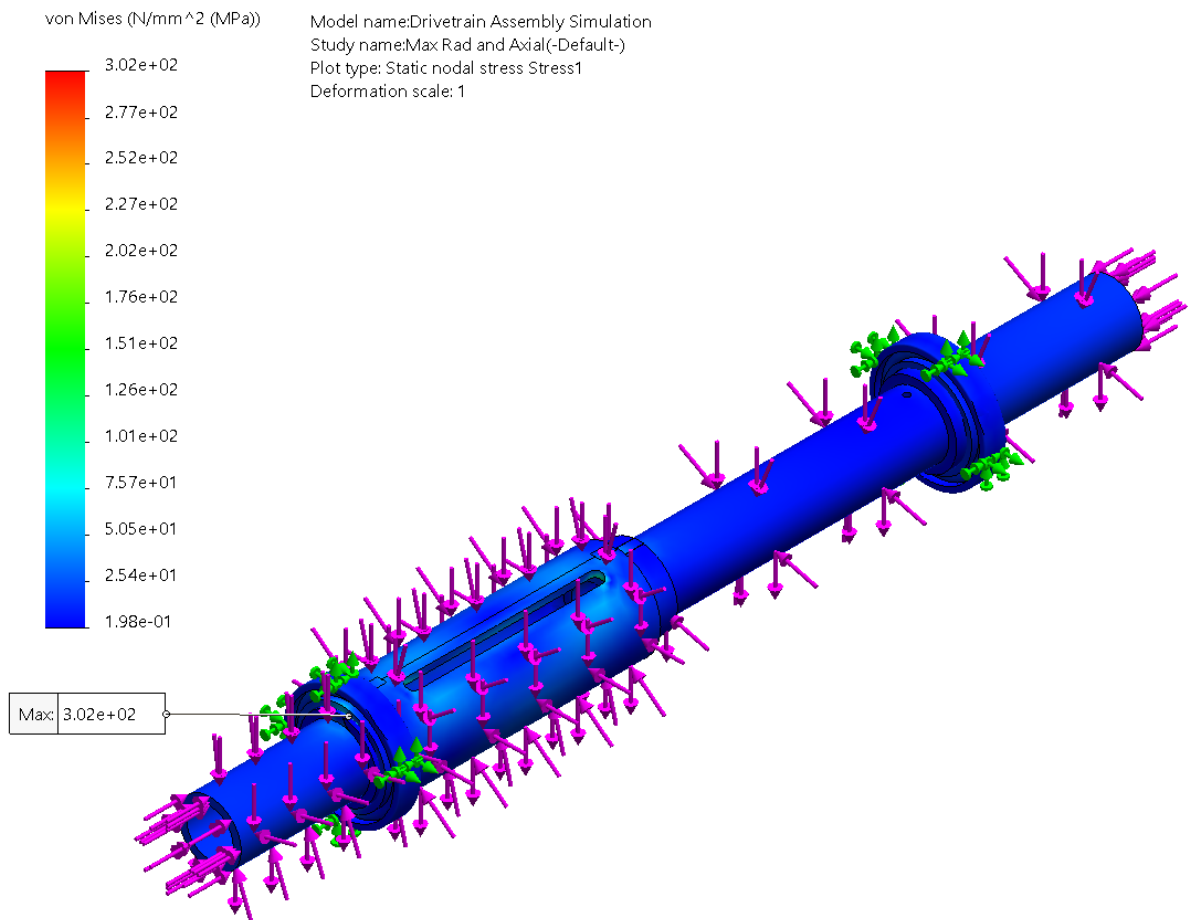


Figure 4.31: Maximum Radial and Axial force

Figure 4.32 illustrates where the deformation would have taken place, and is scaled up 150 times its true scale to exaggerate the result so that it would be easier to see where the critical points are.

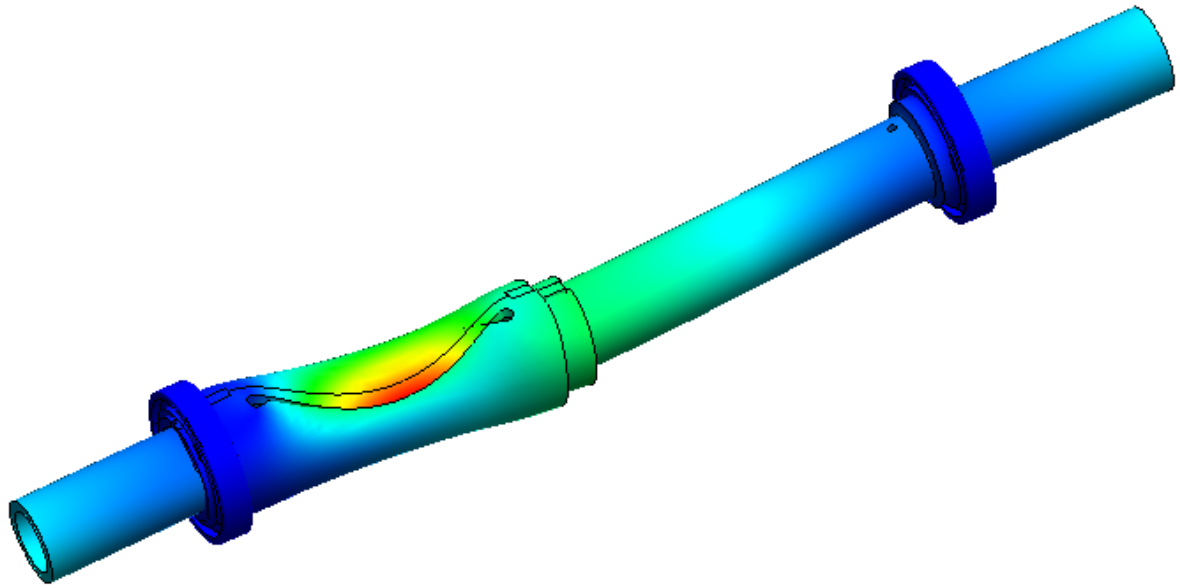


Figure 4.32: Max Radial and Axial scale 150

4.9.3 Vibration analysis

In the requirement sheet there are listed one requirement for random vibrations (R5-08) and one requirement for sinusoidal vibrations (R5-07). For that reason, we performed vibrational testing in SolidWorks with the result illustrated in Figure 4.33.

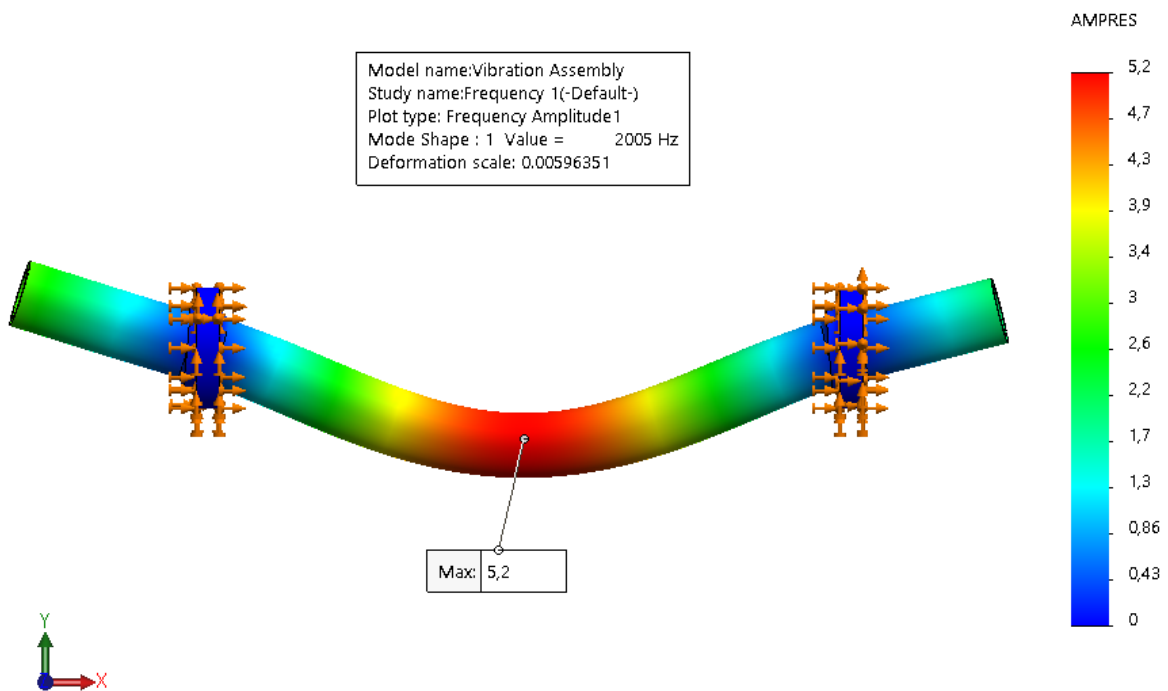


Figure 4.33: Simplified drivetrain 2000Hz

A damping ratio of 0.02 was used [56], and as Figure 4.33 shows a simplified model of the drivetrain and bearings were used to illustrate the whereabouts of the maximum amplitude result and its value. It was made clear that when exposed to 2000 Hz the middle of the drivetrain would be the place where most of the deformation and stresses will take place causing an amplitude result of 5,2mm. Note: The mode shape amplitudes have no units and they do not depict actual displacements. Mode shapes illustrate the profile of the mode only (i.e., the displacement of nodes relative to each other) [57].

4.9.4 Thermal analysis

Since performing thermal analysis on the whole structure would be too demanding, the team chose to focus on one of the lids. This is because the walls are the components that are the most exposed to temperature fluctuations and the lids consists of parts with two different materials. The challenge with having parts made of different type of materials is that some materials contract or expand more than others due to temperature changes.

M3-space is theoretically made of aluminium 7075. This material's thermal expansion coefficient is $2.36e-05$ per Kelvin. The bearing that is mounted on the lid is made of steel 440C with thermal expansion coefficient $1.01e-05$ per Kelvin. This means that the bearing holder will expand more than the bearing when the heat is increased and the bearing will be exposed for compression by the bearing holder when the temperature drops.

The temperature range for this analysis was -40 to 120 degrees Celsius, as specified in the requirements, with the heat coming from the outside of the lid. The analysis was performed on both components separately.

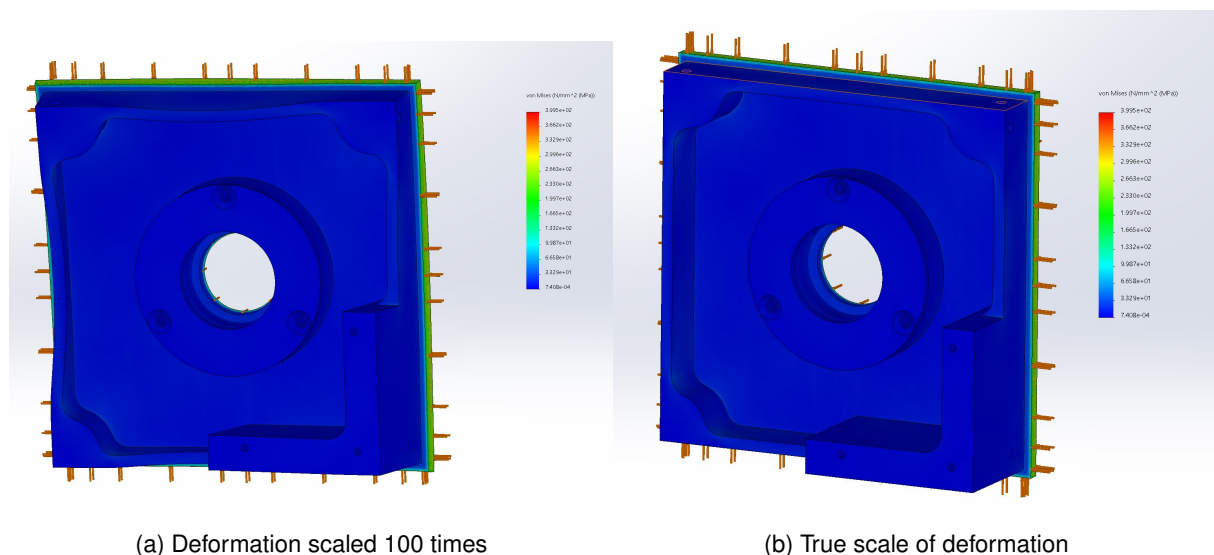


Figure 4.34: Thermal analysis with SolidWorks

After deformation, it turned out that the bearing would not fit inside the bearing holder. This led to a change in design of the bearing holder. The second version of the bearing holder is slimmer, as this will reduce the stresses caused by compression.

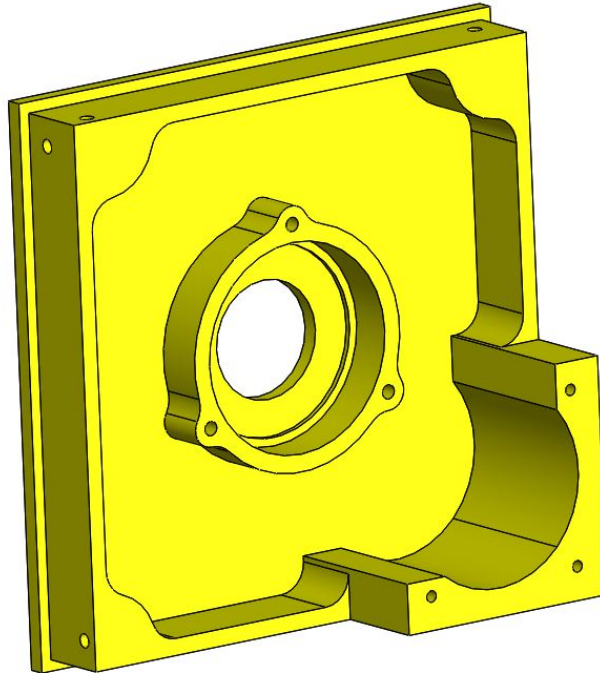


Figure 4.35: The left lid after modification.

Explaining this to the external supervisors also revealed that most of the heat transported in to the SADM is by the shaft. Therefore the heat source in the simulation should have been from the bearing, as this is the only component that is in contact with the shaft. A new test was not performed due to time limitations.

4.10 Diagnostics System

When developing a Solar Array Drive Mechanism (SADM), there are a lot of requirements that has to be fulfilled. To ensure that the SADM will work as expected both before and after launch, verification methods has to be defined and carried out for every requirement. The different verification methods varies in difficulty, and often takes a lot of time to carry out. This is where the Diagnostics System (DS) comes into play.

The main purpose of the Diagnostics System is to act as a combination of the Solar Array Drive Electronics (SADE) and the Main Computer that would be present for a complete assembly of a satellite. The SADE and Main Computer are when combined responsible for facing the solar panels towards the sun by controlling the SADM based on received data . These three components are developed individually before assembly. This means that the SADM has to be tested and verified to ensure that it is able to properly act upon control signals, in addition to fulfilling the given requirements. The combination of a SADE and a Main Computer are replicated in the Diagnostics System, but in this case, they are defined as two subsystems called

Software Layer and Hardware Layer. These subsystems are not only responsible for controlling the SADM, but also collecting feedback data from sensors mounted on or in the SADM. This feedback data is used to verify if a given test is considered passed or failed, where as the test itself is carried out by controlling the SADM in various ways through the software layer HMI.

4.10.1 Software Layer

The software layer of the diagnostics system aims to provide a human-machine-interface (HMI) towards the hardware layer. For rapid prototyping and simplified implementation, it was decided that .NET C# would be used to implement the features needed in this software. There are a lot of available libraries and built-in functionality for this technology that accommodates the needs dictated by the requirements.

Technology stack

- .NET Framework 4.6.1 [58]
- C# [59]
- JSON [60] [61]
- BSON [62]
- NoSQL databases [63] - Specifically document oriented databases [64]

Libraries used in Software Layer

- Newtonsoft JSON.NET [65] [60] [61]
- LiveCharts [66]
- LiteDB [67]
- OxyPlot [68]

The complete API documentation for the software layer can be found here: <https://web01.usn.no/grupper/web-gr5-2019/apidoc/api/SatStat.html>.

The repository containing the complete code solution for both the software layer and the hardware layer can be found here: <https://github.com/thmundal/SatStat>.

4.10.1.1 Alpha 0.1

The Alpha 0.1 version of the software layer is a pre-alpha version to explore possible technologies and libraries that could be used in developing solutions for the requirements. The state of the software at this point is purely conceptual and used mainly for research purposes in conjunction with preparing for architecture design.

4.10.1.2 Alpha 1

The alpha 1 version of the software layer is a pre-release version where we prototype different functionality, and perform unit and operational tests to verify if these are valid solutions to the requirements. This is the first version released as functional test build.

The state of the software at this point is seen as the first iteration, with bare minimum of functionality implemented for something that operates within selected requirement parameters.

The requirements designed and implemented in this version are listed below, and defined in appendix A.11.

- R4-12A: The DS Software Layer should use JSON for input data
- R4-13A: The DS Software Layer should use JSON for output data
- R2-27: The DS Software Layer shall display diagnostics data in real-time

The main goal for this version was to enable communication with a standardized format between the hardware layer and the software layer. The format that was chosen was Javascript Object Notation (JSON), which is ideal for encapsulating data as key/value pairs. JSON is widely used, and libraries for serializing and deserializing information in JSON format are accessible both for C++ and .NET C#.

The secondary goal for this version was to display data collected from sensors in the hardware layer in real-time on a screen through the software layer. This version displays temperature in real-time in a LiveCharts cartesian chart (an x/y graph).

Timeline

Version alpha 1 was planned due Feb. 17 and was delivered on time.

Design

The state of the software layer's design in version Alpha 1 is based on use cases that are spawned from the requirements (figure 4.36). The architecture of the software is defined in a class diagram displayed in figure 4.37. This design is the first iteration of the architecture, and is expected to evolve in further iterations.

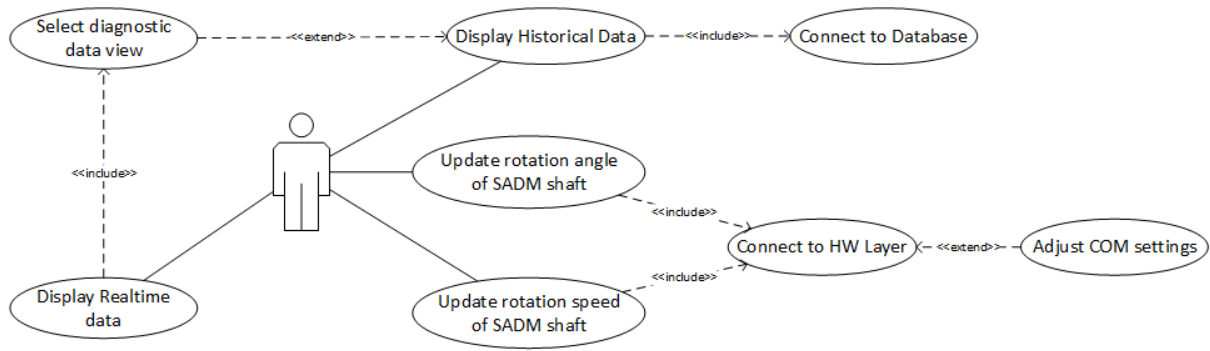


Figure 4.36: Use case diagram DS SWL Alpha 1

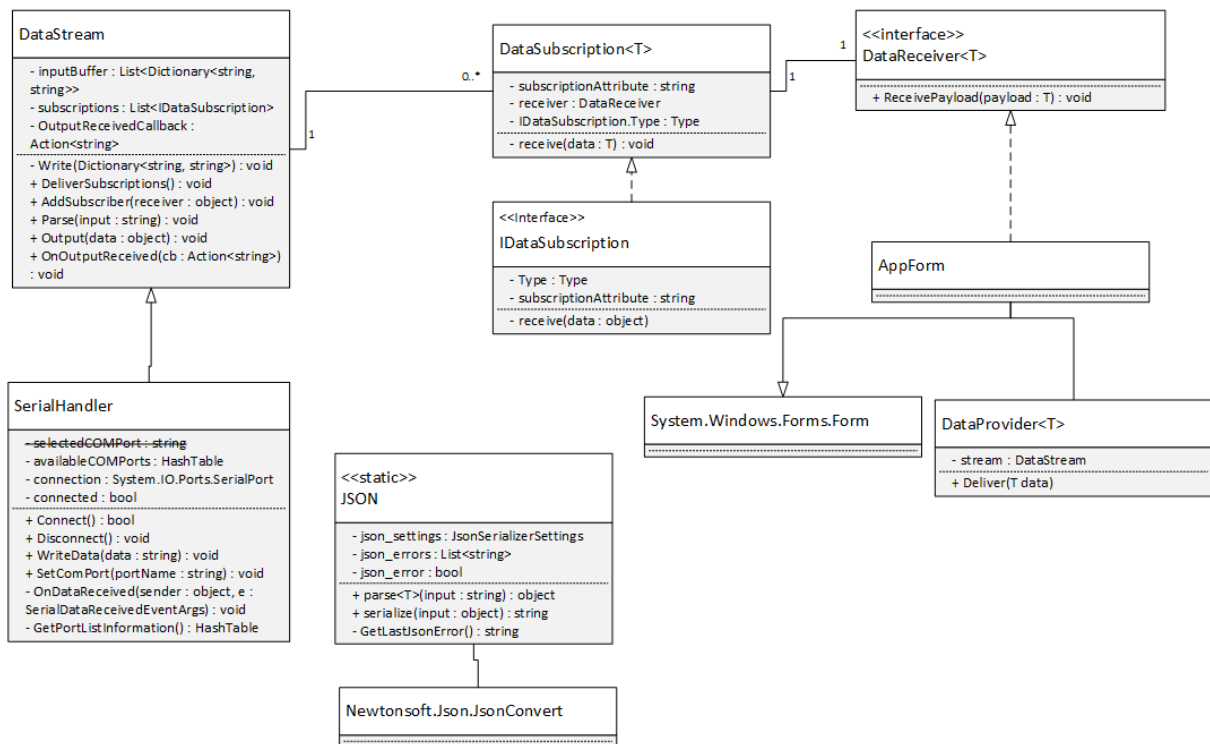


Figure 4.37: Class diagram DS SWL Alpha 1

Architecture

The main philosophy of the architecture, is a subscribe/deliver system. Where some data receiver can subscribe to some data from a data stream. The receiver can act on the data as it is delivered through the overload of `ReceivePayload` method. This means that any data receiver is defined through a class that implements the `DataReceiver` interface. The data that is exchanged through the stream is expected to be a string formatted as JSON, and has to be parsed by a JSON interpreter before the receiver can use the data for anything. As a result of this, there is a need to specify what data type the data is so that it can be cast to the correct type, and in turn be used in the methods defined in the receiver itself. This part proved to be challenging in terms of scalability and ease of use, and was one of the concepts that was iterated on and heavily changed in further versions.

Testing

As the architecture and design of classes and class hierarchies was expected to change during iterative development, a series of unit tests was defined for the core functionality concerning data communication. This proved to be extremely helpful, and helped with verifying that the requirements were still met even though the design and architecture changed several times. The test report for the unit tests performed for alpha 1 can be viewed in appendix A.16.

4.10.1.3 Alpha 2

Alpha 2 is the second iteration of the software layer, and is the second functional build. In this version, the main goal was to establish a standard communication protocol for exchanging data between the hardware layer and software layer of the diagnostics system. The communication protocol is defined in appendix A.22.

The requirements fulfilled in this version is listed below, and described in appendix A.12.

- R4-14: DS Software layer GUI shall support changing underlying COM port settings
- R2-28; The DS Software Layer shall have modular design for easy addition of diagnostic components
- R2-32: The DS Software layer shall be able to calibrate position and speed of the SADM drive shaft
- R4-20: The DS Software Layer should provide an interface for adjusting calibration settings such as position and speed of the SADM motor
- R4-25: Shall follow a unified communication protocol

Timeline

Originally this version had a deadline Feb 23, but was moved to Mar 2. There was a lot of changes done in this iteration, and the team needed more time to fit it all together. It became increasingly challenging to synchronise the advancements between the software and hardware layer. To have a functional version, both layers have to be compatible.

Design

The design of Alpha2 is the next iteration of the design of Alpha1. The previous design was taken as a baseline, and developed further. With a set of unit tests already defined for the core functionality, they were heavily used as a source for test driven development. The class diagram displayed in figure 4.38 is generated from the code that resulted from the development.

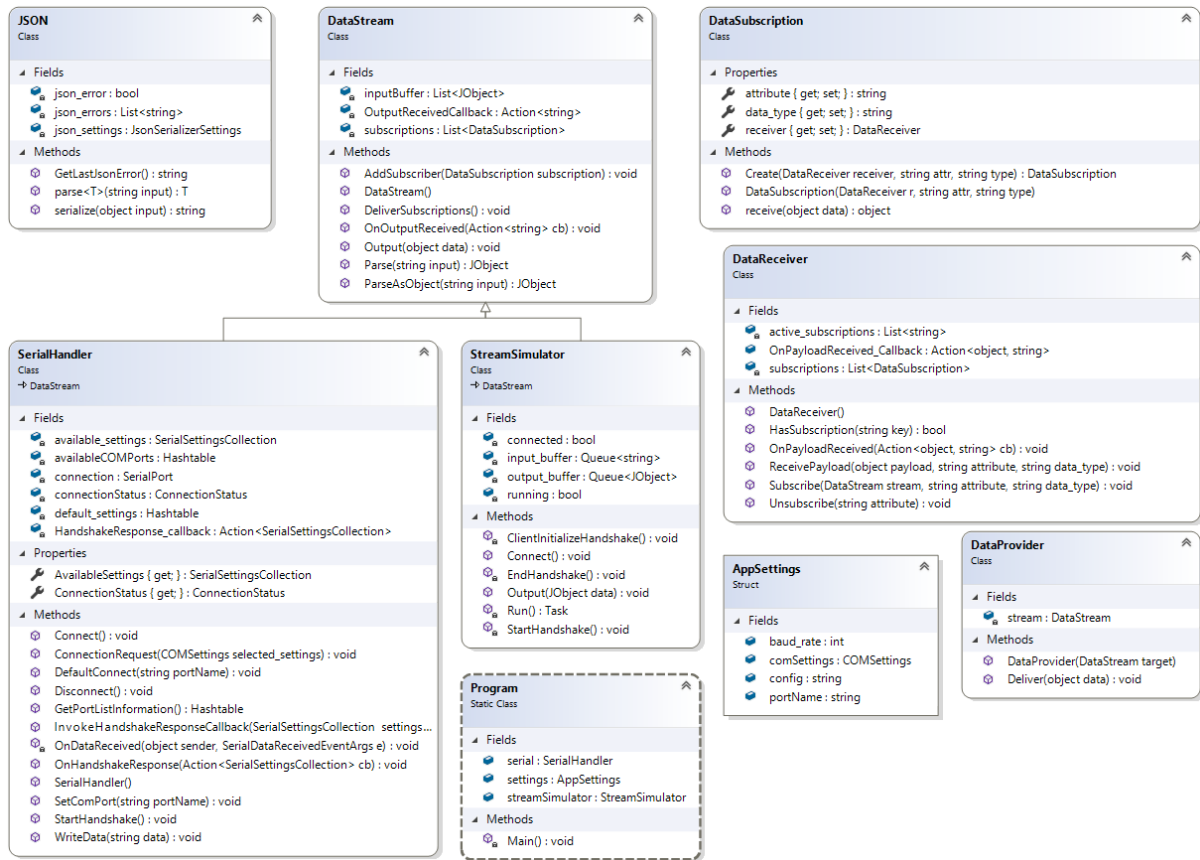


Figure 4.38: Diagnostics system software layer alpha 2 class diagram

Architecture

The previous concept of a subscribe/deliver philosophy is still present in this version, and remains the core architecture of the software. However in this version, all generics are removed, and interfaces that were previously needed to list generic objects of different generic types are no longer needed. The data types associated with the data received from a stream, is now cast and handled at the very end of the object’s lifetime, just as it is about to be used by a data receiver. This made it very much simpler to handle lists of subscriptions on the data streams.

Testing

The collection of unit tests evolved with the architecture, and was rewritten to accommodate the new structure. The unit tests were heavily used in conjunction with development and provided a base for test driven development. There was also a number of inspections done through code auditing, and functional testing to verify requirements that could not easily be tested through unit testing. A unit test report for this version can be viewed in appendix A.17.

4.10.1.4 Alpha 3

Alpha 3 is the third iteration of the software layer, and the third functional build. New features in this version include the ability to select specific data to view, saving viewed data to database to look at later, as well as viewing stored data from a database. A database controller was added, and LiteDB [67] was used for this. The requirements fulfilled in this version are listed below, and described in Appendix A.13.

- R4-15: DS Software layer GUI shall support selection of diagnostic data output view
- R4-16: DS Software layer GUI shall support displaying historical data
- R2-29: The DS Software Layer shall save diagnostic data to a database

Timeline

The release tag was created and finalized March 22. The development of this version took more time than estimated due to preparations for the second presentation as well as some problems encountered with optimization of the plot view. An optimization pass of the plot view was planned for a future version, but the slowdown was so severe that the entire plotting library was changed from LiveCharts [66] to OxyPlot [68]. A key factor in getting rid of the slowdowns now rather than later was that the software was actually being actively used in another project at the university at this point.

Design

New elements taken into the design in this version is database handling and some updates to the network communication handler. A framework for database handling was introduced, and LiteDB [67] was selected for this purpose. The reasoning behind using LiteDB as the database framework, was to avoid having to run a dedicated server software to host database collections. LiteDB is a serverless local database framework, and is a document oriented no-sql database. This makes the installation more compact, and there is no need for advanced quering of data in this software, even though document oriented databases can provide good quering as well.

Alpha 2 formalized new use-cases related to database handling and connecting to different data sources. The internal stream simulator was previously implemented in Alpha 2, but was introduced as a part of the design in Alpha 3. This lead to further generalization of the data stream class and allowed for multiple different data sources to be configured, not only through serial or USB.

Alpha 3 Use Case Diagram

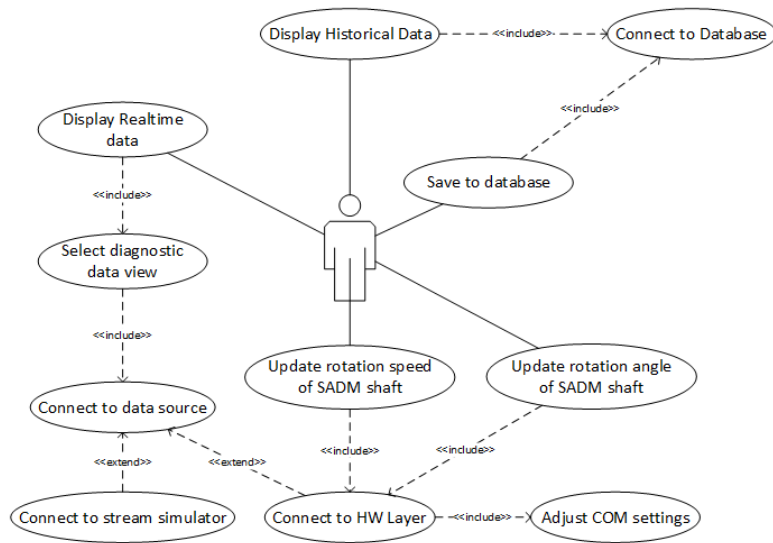


Figure 4.39: Alpha 3 use-case diagram

Architecture

The big additions in the architecture in this version is the implementation of database communication for saving collected data to a database, and viewing this data at a later point. Other than this, the architecture for the rest of the software remains more or less the same.

Testing

Numerous unit tests were performed to verify that database saving and querying yielded correct results. Some functional tests were done for verifying that the UI behaves as expected, and that the system indeed displays the correct data that was saved to database when trying to view it.

4.10.1.5 Beta 1

The release of version Beta 1 spawned the beta branch, and was planned to implement higher level features and functionality within the software. Beta 1 specifications can be viewed in appendix A.14.

Requirements considered in this version:

- R2-33: The DS Software layer should be able to remember previous COM settings and try to establish a connection to HW layer with these
- R2-27: The DS Software Layer shall display diagnostics data in real-time
- R2-34: DS SW should be able to configure desired maximum and minimum limits for operational sensor values to display status indicator on live input values during diagnostics run. Values outside accepted values should indicate deviation
- R2-34.01: DS SW should provide functionality to save diagnostics settings to a template
- R2-34.02: DS SW should be able to load and apply settings from a diagnostics settings template

Timeline

Beta 1 was finalized April 15, and was delivered on time with respect to a revised roadmap after the delay of Alpha 3.

Design

The concept of observable values was introduced with this version. The aim was to have a way of observing data that is received from a data stream, and determine if the data received is valid with respect to set upper and lower bounds. With this feature, the software should be able to inform the test engineer if the system deviates from its expected behaviour during testing or calibration.

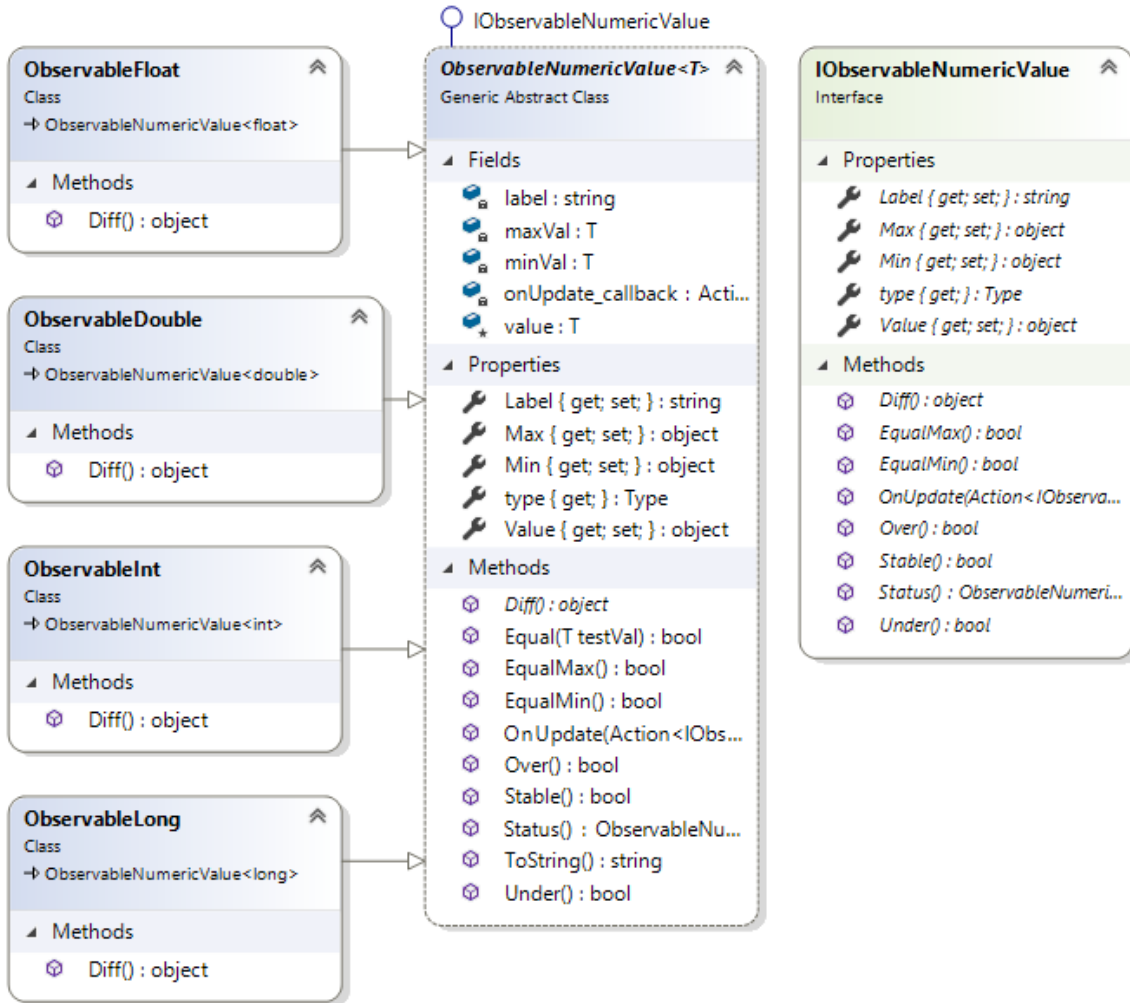


Figure 4.40: Class diagram for the ObservableNumericValue system

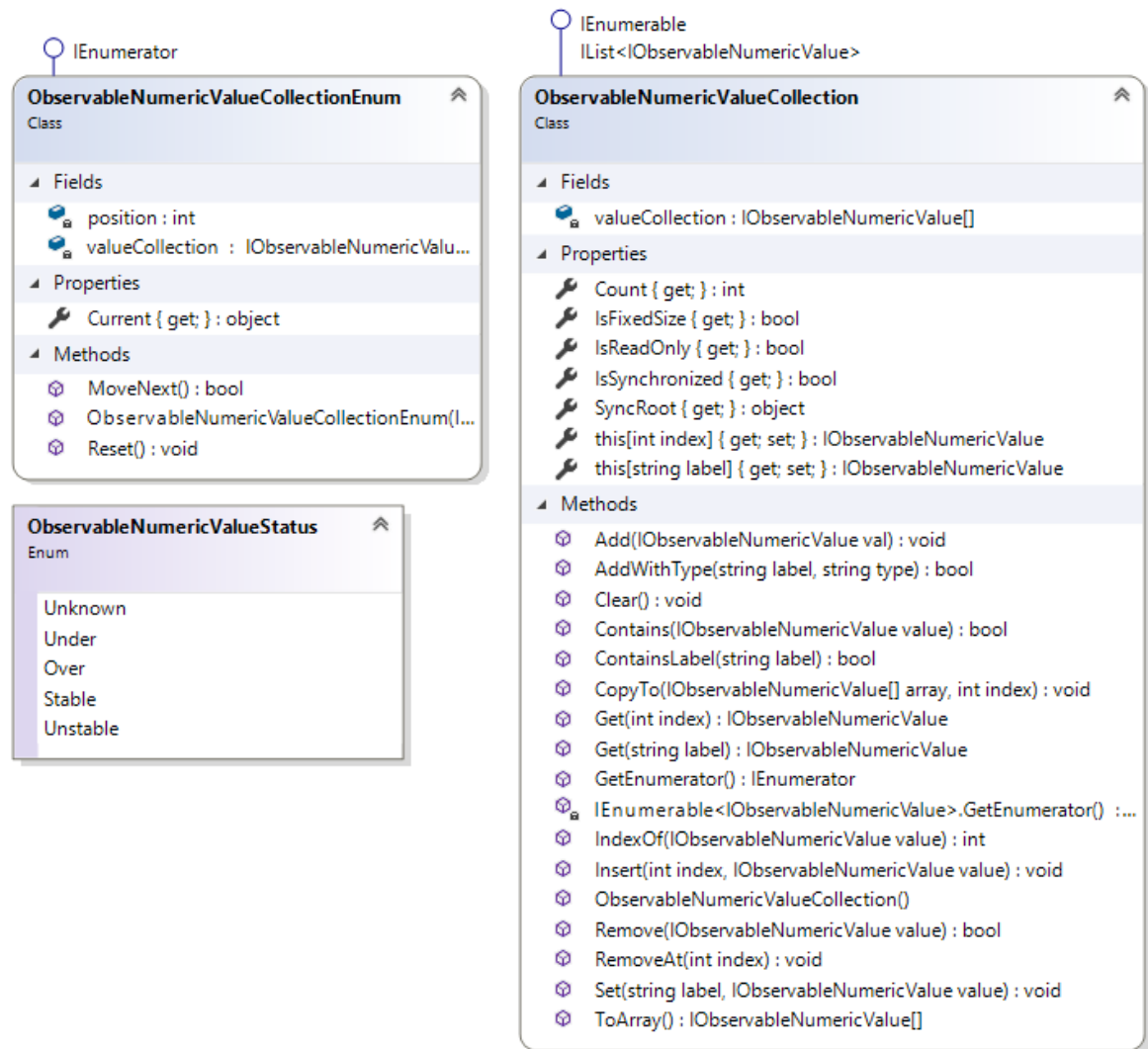


Figure 4.41: Class diagram for the ObservableNumericValueCollection system

Architecture

An observable numeric value is an object of a class that implements the interface `IObservableNumericValue`. A numeric value can contain a value of type `int`, `float`, `double` or `long`. There exists both a non-generic interface, and a generic abstract class that in combination enables lists or collections of templated observable values containing values of different data types. A list of observable numeric values would be initialized as `List<IObservableNumericValue>` to which any type of observable value that inherits from the generic abstract class `ObservableNumericValue` can be added. An actual observable value that contains for example an `int`, is instantiated through a subclass of `ObservableNumericValue`, for example `ObservableInt`. Subclasses of `ObservableNumericValue` must override the method `Diff` in order to calculate the difference of the value against another value of same type. An example of using an observable numeric value is shown in listing 4.1.

```
List<IObservableNumericValue> observableValues = new
    List<IObservableNumericValue>();

ObservableInt oInt = new ObservableInt {
    value = 42,
    label = "Answer to life, the universe and everything"
};

observableValues.Add(oInt);
```

Listing 4.1: Instantiating an observable numeric value

Observable values can be kept in a special collection built to handle these types of objects, called `ObservableNumericValueCollection`. This collection also has helper methods for adding a numeric value of a type based on a string. This is handy for values that are delivered to the software from an external data source, such as the hardware layer. The data passed in this manner only contains JSON data that is passed as strings, where the explicit data type for each of the fields in a JSON structure is not known at runtime. An example of using a collection and adding a value in this manner is shown in listing 4.2.

```
ObservableNumericValueCollection collection = new
    ObservableNumericValueCollection();

collection.AddWithType("answer", "int");
collection["answer"].Value = 42;
```

Listing 4.2: Using a `ObservableNumericValueCollection` with `AddWithType`

Testing

Numerous unit tests and functional tests were performed to ensure the functionality of observable values and the other implementation. Refer to appendices A.19 and A.20 for the verification reports.

4.10.1.6 Beta 2

Beta 2 builds on the previous iteration and aims to implement a functional interface and system for creating, performing and saving automated tests. Through an interface, a user can specify instructions to run on connected hardware in a queue, and gather sensor data during the test to monitor the state of the mechanical system during operation. Requirements specification for Beta 2 is included in appendix A.15

Timeline

The planned release for this version was Apr 28, but was delayed until May 13. Beta 2 was meant to include report generation for automated testing, but as of the time of writing, the implementation of this has not been achieved.

Design

Classes representing a test configuration and instruction were introduced. These play along with observable values and data streams to perform tests and measure results gathered during the tests. A fully featured socket server was formalized and finalized to allow for connections towards external simulations. This feature had existed since Alpha 2, but not been an official part of the software or even the requirements. Using this feature with a virtual representation of the hardware, the system as a whole could be tested conceptually.

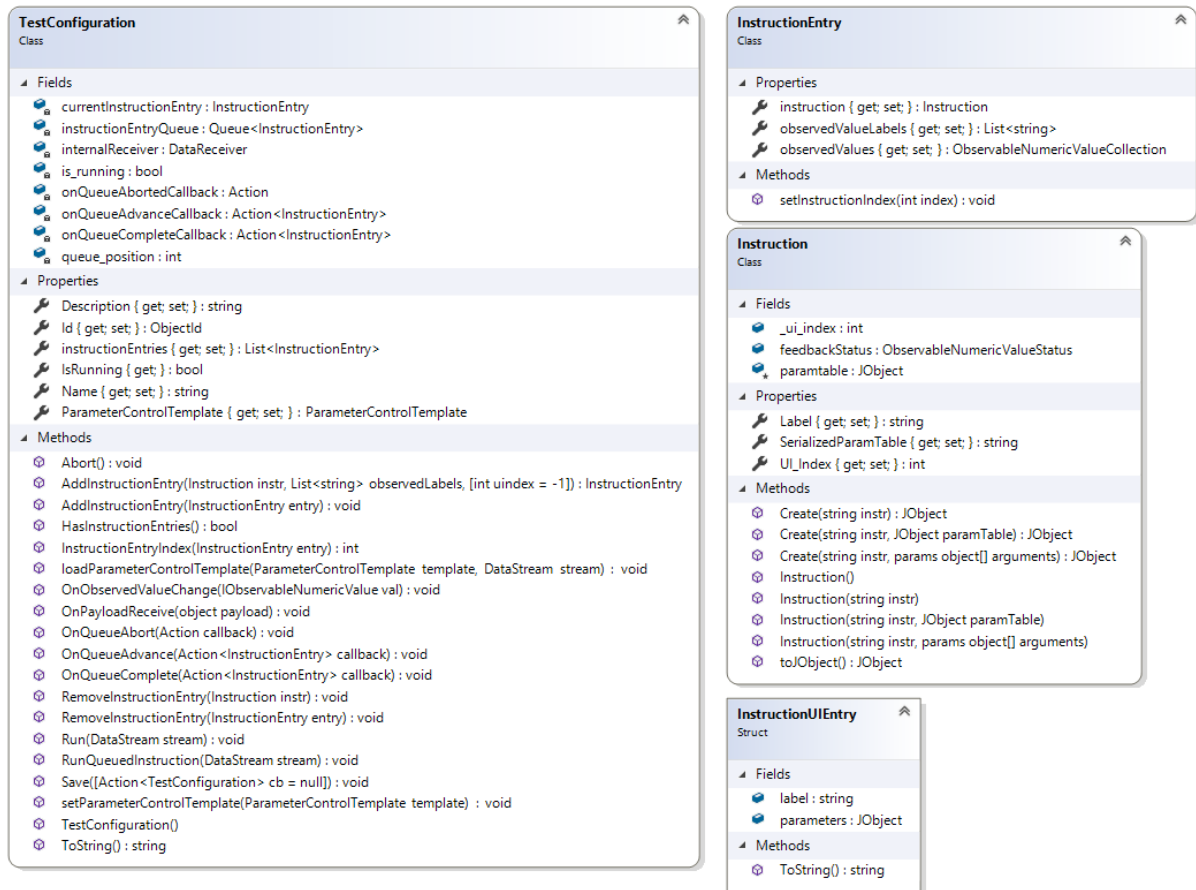


Figure 4.42: Class diagram concerning test configuration

The use-case diagram for beta 2 is populated with all the use cases for this current version. The diagram is cleaned up a bit from the previous version for better readability, and introduces a use case for connecting to an external simulation, and using the test configuration system.

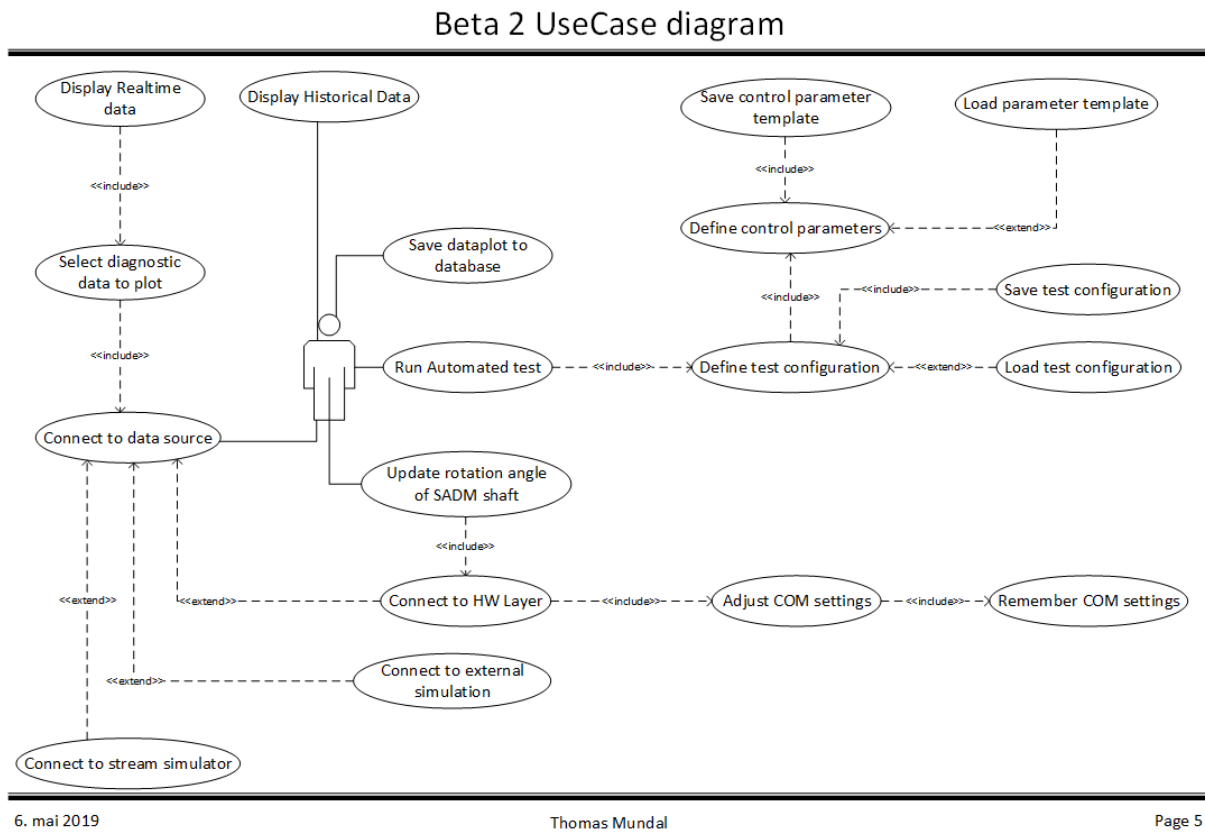


Figure 4.43: Beta 2 use case diagram

Architecture

A test configuration essentially is a queue of instruction entries. Each instruction entry contains an instruction to run on the connected device, and a set of parameters to watch. A test configuration notifies when a queued instruction is finished, and can act internally when an observable value changes.

Testing

This version was tested extensively through functional testing. Even though there are only two tests defined, the behaviour of the software heavily depends on that these tests were successful.

4.10.2 Hardware Layer

4.10.2.1 Hardware

The hardware of the diagnostic system consists of control cabinet with implemented electronics, attached wires and sensors.

4.10.2.1.1 Control Cabinet The function of the cabinet is to have a practical, easy and safe way to assemble the DS with the SADM for test and running. The main components in the cabinet are the Arduino and motor driver with implemented firmware and voltage supplies to provide power for the equipment, run the SADM and test the power transfer system in the SADM. The electrical schematics for the control cabinet are documented in appendix A.7.

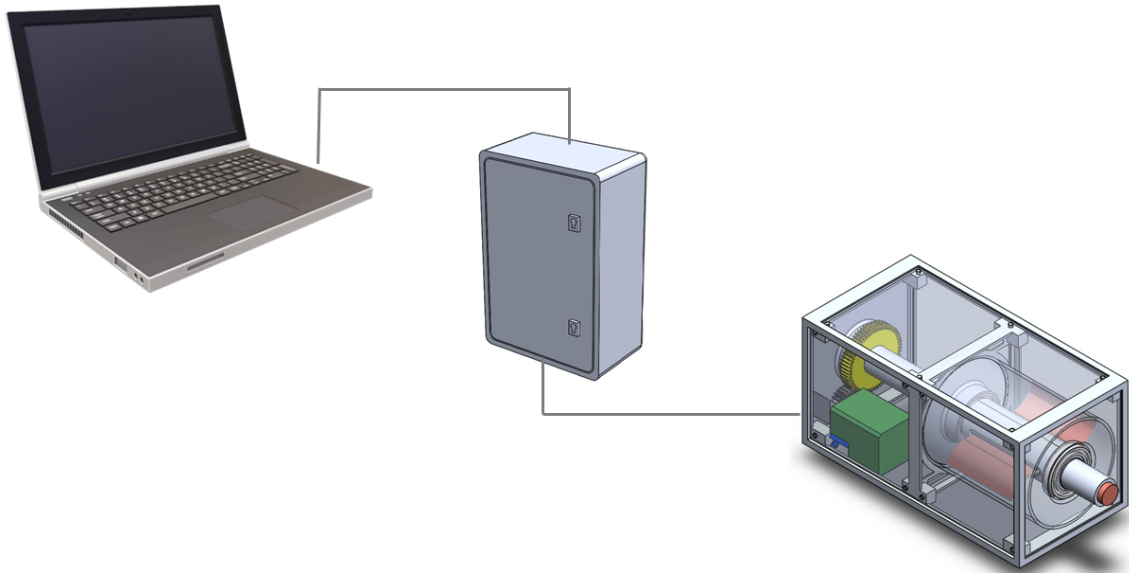


Figure 4.44: The control cabinet's role in the system

The figure 4.44 illustrates how SWL, HWL and SADM are connected. SWL is illustrated as a laptop.

The components inside the cabinet:

- G1: 24VDC Power supply. To provide power to the motor, motor driver and the angular position sensor, a 24VDC power supply is used to transform 230VAC to 24VDC. The expected voltage level from the rest of the satellite is about 19VDC. We chose to use 24VDC because this is a standard voltage level for control systems and there is a wide selection of equipment that operates at this voltage level. In order to test the power transfer line with a proper load, the 24VDC power supply has to be able to deliver a current of up to 20A.
- G2: 5VDC Power supply. The firmware runs on an Arduino Mega [69], which requires 3-12VDC voltage. This power supply transforms 24VDC voltage to 5VDC and supplies the Arduino Mega and its digital inputs.
- J1: Motor drive. To run and control the stepper motor in the SADM. Takes input from preprogrammed parameters and the firmware.

- J2: The Arduino Mega running the firmware.
- S1: Switch for operating the the SADM manually.
- F0, F1 and F2: Fuses for protecting the equipment.
- K1, K2 and K3: Relays triggered when using the manual switch to give feedback to the firmware that the SADM is running manually and cut the pulse signal from the Arduino Mega to the motor driver.

Complete list of hardware connected to the control cabinet is listed in appendix A.47.

4.10.2.1.2 Feedback and Sensors The elements of biggest interest due to feedback from the SADM to the DS are the angular position on the shaft and the temperature changes in the twist capsule when the power transfer line is loaded. Sensor data is sent to the firmware for interpretation before it is sent to the SWL. The sensors' position in the DS is illustrated in figure 4.45, where an angular position sensor is used in the example.

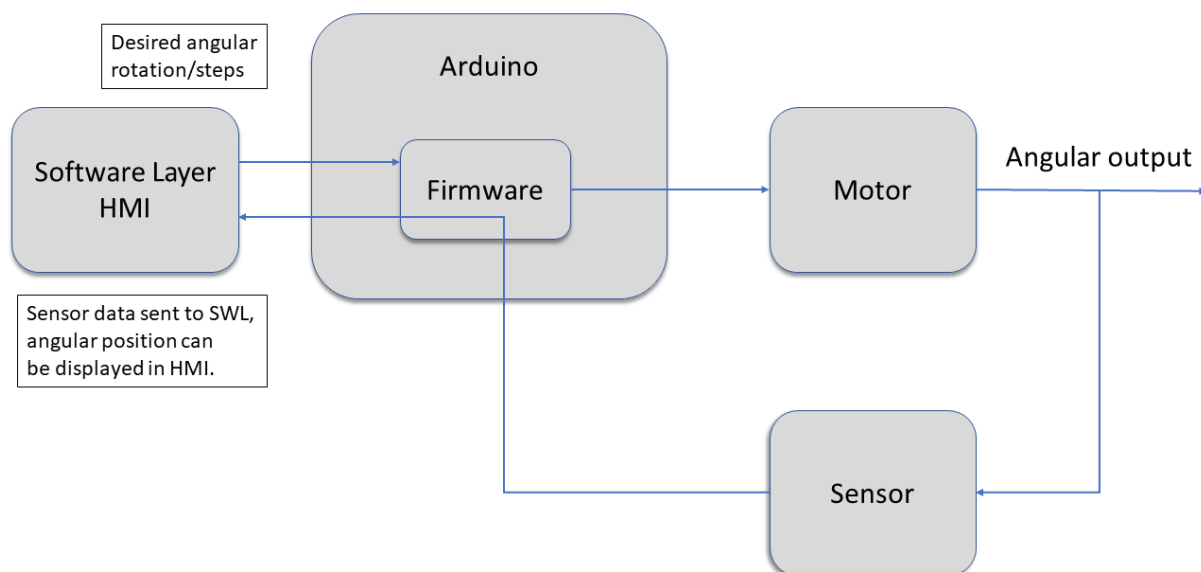


Figure 4.45: The sensors send data to the firmware for interpretation before it is sent to the software layer

In order to control the solar arrays' angle, a more complicated control system was suggested; using a PID-controller that received data from an angular sensor and processed the output due to the desired angle on the solar arrays. The idea was discarded - with time and maturation the team decided that this solution would become an over-doing implementation and would cause more cost than profit.

- A stepper motor's angular output is discrete. The mathematics needed to determine the number of steps necessary to provide an angle are simple.
- The system is stable.
- Validation with external supervisor and other sources from commissioner confirms that an implementation of an angular position sensor on a cubesat will be superfluous. This is discussed in the section below.

Angular position sensor: The angular position sensor was important to implement for feedback to DS so that the whole product, DS and SADM, could be verified. The angular position sensor gives input to the firmware to determine if desired angle on the shaft is achieved. The team decided that the position sensor should be a part of DS, not the SADM. This is explained by the background for the project; cubesats is supposed to be simple, cheap and effective to produce. Only the necessary components should be implemented. An angular position sensor is redundant in this context, partly because of the stepper motor and there are no heavy requirements for accuracy. The stepper motor's output is discrete which simplifies the control of the angular output.

The needed technical characteristics for the position sensor is documented in the requirement sheet A.61 as R2-38.

The VERT-X sensor from Contelec was chosen, and meet the requirements [70]. The sensor's output is a voltage level between 10 and 90% of the input voltage - 5V, provided by angular displacement from the reference. Some important characteristics from the data sheet are:

- Resolution: 12 bit

$$\frac{1}{2^{12}} \cdot 360^\circ = 0.088^\circ \quad (4.11)$$

- Output: A voltage level between 10 and 90 % of the supply voltage 5VDC: 0.5V equals 0°, 4.5V equals 360°.
- Independent linearity: +/- 0.3% of measured range (360°).

$$\frac{360^\circ}{100\%} \cdot 0.3\% = 1.08^\circ \quad (4.12)$$

- Max. hysteresis: 0.1°
- Max repeatability: 0.1°

The total error in worst case is:

$$1.08^\circ + 0.1^\circ + 0.1^\circ = 1.28^\circ \quad (4.13)$$

Another aspect of resolution in sensor readings the resolution on the Arduino Mega's input, which is 10 bit. The correct resolution in degrees is therefore:

$$\frac{1}{2^{10}} \cdot 360^\circ = 0.352^\circ \quad (4.14)$$

For the sensor to provide readings on the shaft, a gear is mounted on a shaft and one on the sensor's shaft. The gear ratio is 1/1. Thus, all of the gears are 3D-printed, and error due to mechanical fittings needs to be considered in this context. This taken into account, the total error caused by the sensor may be negligible in connection with the mechanical slack.

Temperature sensor: Temperature changes inside the SADM, especially inside the twist capsule, when simulating current from solar arrays is a parameter of interest. Due to major temperature changes and the need to protect the materials and electronics - including the power transfer line that provides power to the satellite - the heat dissipation from the electric circuits should be as low as possible. The electric circuits that transfer power will heat up if not correctly dimensioned according to standards.

Sadly, the flex had not arrived when it was time for the corresponding tests to be carried out. It was therefore not possible to fulfill these tests. In order to perform qualified tests, the SADM should be in a vacuum chamber, and the temperature element needs to fulfill specific criteria and should be placed inside the twist capsule. The team wanted to do a less qualified test out of curiosity.

4.10.2.2 Firmware

The hardware layer (HWL) of the diagnostics system is the interface between the software layer (SWL) and the physical hardware components. The main responsibilities of the HWL is to control the SADM according to instructions received from the SWL, as well as providing feedback sensor data in order for the SWL to monitor the results of an executed test. The decisions made during the development of the HWL are described in detail in the document "HWL Decisions", appendix A.24.

Technology stack:

- Arduino Mega 2560 [69]
- C++ [71]
- Visual Studio IDE [35]
- Visual Micro [31]
- Doxygen [72]
- TeXworks [73]
- RS232 protocol [33]
- JSON [60] [61]

External libraries used in hardware Layer:

- ArduinoJson [74]
- DHT Library [75]
- Adafruit DHT Humidity & Temperature Sensor Library [76]
- Adafruit Unified Sensor Driver [77]
- QueueArray [78]
- Stepper [79]

Libraries developed alongside the HWL:

- LinkedList by Nils Erlend Heggem & Jon Skjelsbæk A.41
- Subscriber System Template Library (SSTL) by Jon Skjelsbæk A.42

The API documentation for the different hardware layer versions can be found in appendices A.43 through A.46. The repository containing the complete code solution for both the software layer and the hardware layer is available on github [80].

4.10.2.2.1 Motor Driver and Parameters The motor driver [81] is the kind *bipolar chopper* that is suitable for the motor [82]. Bipolar chopper indicates that the motor driver varies the voltage to keep the current approximately constant. The benefit of this configuration is that the motor can be driven at higher torque - and speed, which is not of interest due to this task other than the constant-voltage configuration [42, Chapter 18.1.1].

The parameters for the motor drive are set by using its associated software; EmentoolLite [83]. The parameters can be divided into two groups;

- Parameters for any operation; current, stepping mode, holding current.
- Parameters for manually operation: Run direction, speed, number of steps and ramp time.

The parameters for manual operation are activated by a voltage input in one of the four input pins on the motor driver, which is triggered by a manually operated switch on the control cabinet. The motor driver also has a pulse input pin, which in this configuration is fed with a digital output from the Arduino microcontroller. The driver responds to every rising edge on this pin by moving the motor one step. Using this input, the speed and the angular displacement of the motor shaft can be controlled by the firmware on the Arduino for automated operation.

The parameters for any operation:

- **Stepping mode:** This motor driver can provide microstepping. One step can be a full step or $\frac{1}{2^n}$, $n : 1, \dots, 6$. Microstepping results in a smoother operation of the motor [42], which is interesting in this context due to vibrations that are undesirable and resolution on the output angle. The motor has an 1.8 degrees full step angle, which means that the number of steps in order to rotate one round is:

$$\frac{360^\circ}{1.8^\circ \cdot 1/1} = 200 \quad (4.15)$$

In full step mode the motor needs 200 steps to complete 360 degrees.

- **Current after 1s stop:** The motor's holding torque is dependent on the current in the windings when not running. In this context, capability to hold the solar arrays still when the satellite has rotational accelerations or is affected by other force, the holding current is desired to be as high as possible. Can be set to between 0 and 100% of the phase current.
- **Phase current:** Can be between 0.1 - 6A. Since the motor rate current is 1A, this parameter must not be set over that limit in order to prevent damages to the motor.

The parameters for only manual operating using preset function:

- **Ramp time:** Desired ramp time is considered due to acceleration and power consumption. For this configuration the ramp time should and can be high. Can be set between 0.1 - 5 seconds. The ramp time is only used when running manually with preset-function, so the ramp time when running it automatically from DS has to be set in the firmware (Arduino microcontroller).
- **Run directions:** Can be set to clockwise and counterclockwise.
- **Run frequencies:** This is the parameter to set the stepping frequency in Hz, and the angular velocity ω is decided by this parameter. When running automatically mode, the frequency on the pulse input decides this factor.

To calculate the angular velocity on the output shaft, the full step size, stepping mode and gear ratio (driving/driven) between motor shaft and shaft needs to be considered:

$$\omega = f \cdot \text{steppingmode} \cdot 1.8^\circ \cdot \frac{1}{2} \quad (4.16)$$

In order to achieve an output angular velocity of approx $3^\circ/s$ when using manual operation/preset function, the following values of parameters can be used;

- Stepping mode: $\frac{1}{16}$
- Ramp time: 5 seconds
- Current 1 second after stop: 100% (1 Ampere)
- Phase current: 1 Ampere
- Preset run directions: Both is in use, turn manual switch to left to go counterclockwise and right to go clockwise.
- Preset run frequencies: 53 Hz

The output velocity when running manual mode is therefore:

$$53Hz \cdot \frac{1}{16} \cdot 1.8^\circ \cdot \frac{1}{2} = 2.99^\circ/s. \quad (4.17)$$

Since the parameter setting in EmentoolLite [83] is so user friendly and the parameters can quickly be changed, several test runs is preformed. The most important thing the team learned from these tests is that the current level needs to be 90-100 % of maximum rated current, and the performance does not seem to be affected by the step rate. When the motor was assembled in the SADM the temperature on the motor's surface became lower.

4.10.2.2.2 Alpha 0.1 Version Alpha 0.1 is a preliminary version prior to the Alpha and Beta pre-releases. For the HWL specifically, this version was used to test communication with the SWL through the serial port, and controlling the stepper motor based on data received from the SWL.

4.10.2.2.3 Alpha 1 The diagram shown in Figure 4.46 represents the planned architecture and interconnections of the HWL for Alpha 1. The requirements fulfilled in this version is available in appendix A.25.

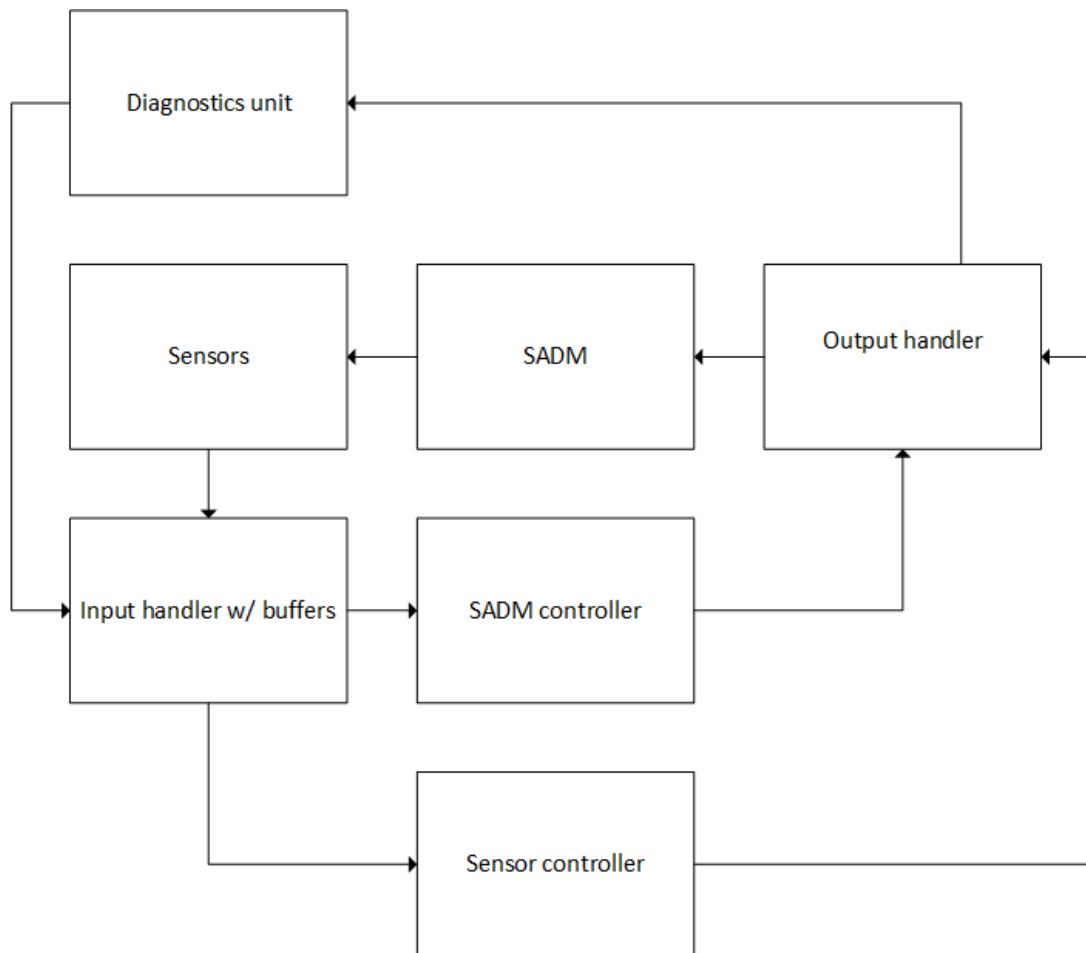


Figure 4.46: HWL interconnection diagram

The diagnostics unit in this diagram represent the SWL of the diagnostics system. The SWL is described in section 4.10.1.

The input handler was the receiving end of the hardware layer. This module was responsible for buffering the instructions received from the SWL in order for the different controllers to fetch the instructions when ready.

The output handlers' main purpose was to translate the input data to a format the device to be controlled would understand. An example could be that the SADM controller fetched an instruction from the input handler, and passed it to the output handler. This data was then processed by the output handler to properly control the SADM.

The intended purpose of the different controllers was to work like an interface between the input- and the output handler. The idea was that there should have been a designated controller for every device to be controlled. A controller would have known what input handler buffer to fetch instructions from, and what output handler function to use in order to properly control the corresponding device. There were two planned controllers for Alpha 1. One for the SADM, and one for the sensors to be used in the physical test rig. These controllers were never implemented in this version because the main focus was to implement the handlers.

At this point, it was decided that different sensors were going to take place in the testing rig, but what kind of sensors was unknown at the time. This is why the sensors are represented as a "black box" in the diagram.

Alpha 1 was tested by reading the DHT11 temperature and humidity sensor and sending the data to the SWL. The requirements fulfilled in this version were verified through tests and inspection, and the results are described in the verification report for HWL version Alpha 1, appendix A.27. The class diagram for Alpha 1 is available in appendix A.26.

4.10.2.2.4 Alpha 2 Version Alpha 2 introduced the SatStat communication protocol available in appendix A.22. This protocol defines how the data transmitted between the HWL and SWL shall be formatted, what instructions the diagnostic system supports, as well as a handshake protocol for proper connection establishment. The introduction of this protocol required the HWL to support transmission and interpretation of JSON formatted data. As a result of being able to receive and interpret JSON formatted data, the HWL was expanded with the ability to control the SADM according to instructions received from the SWL. These changes are related to the requirements stated in appendix A.28.

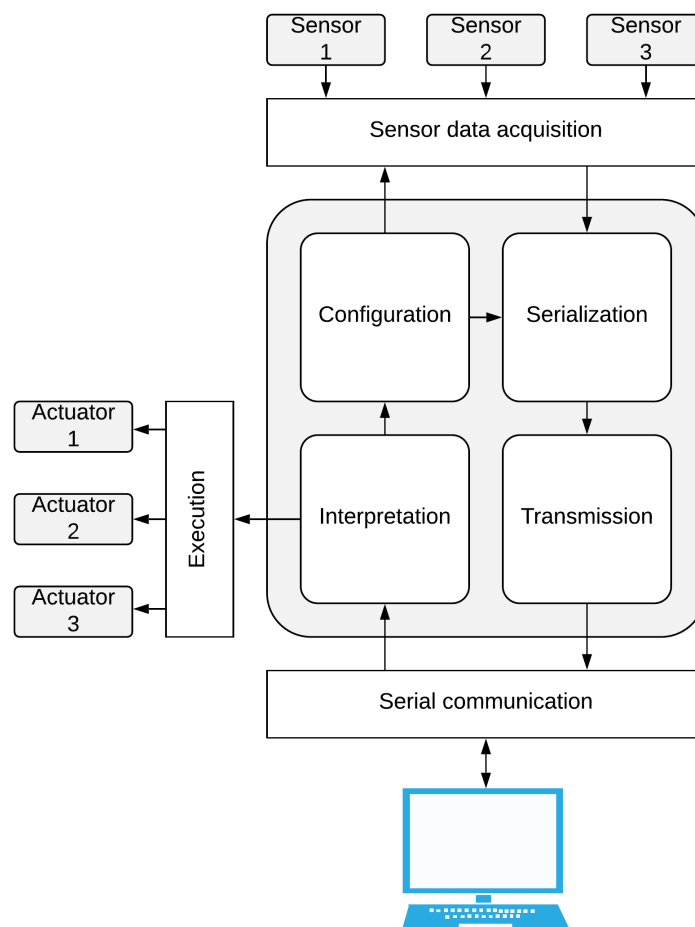


Figure 4.47: Diagnostics System HWL architecture diagram.

Figure 4.47 shows how the internal architecture of the HWL controls the external devices. External devices in the case of the HWL would be sensors, actuators and the SWL.

The SWL is represented as a desktop computer in the architecture diagram. It communicates with the HWL via a serial port connection, and is able to control the external devices and change parameters internal to the HWL by sending different instructions.

There are four main aspects to the internal structure of the HWL: interpretation, configuration, serialization and transmission. The interpretation stage prepares instructions received from the SWL either for execution on external actuators, or configuration of parameters internal to the HWL. During the handshake protocol, the configuration stage is used to set the baud rate, configuration and new line format for the serial transmission. In this version, Alpha 2, the HWL always sent all available data to the SWL, but the idea of implementing a subscriber system in order to only send requested data was planned already at this point. The serialization stage of the architecture is the stage where data to be sent gets prepared for transmission by converting it to JSON format according to the SatStat communication protocol. When the data is properly formatted, it is passed into the transmission phase which handles the actual transmission of the data.

As mentioned, the requirements fulfilled in this version were related to SADM control and transmission of JSON formatted data. The verification report for Alpha 2 can be found in A.30, and the class diagram is available in appendix A.29.

4.10.2.2.5 Alpha 3 The requirement covered in version Alpha 3 is related to implementation of error handling for received data, and can be found in appendix A.31. The first place where things might go wrong when receiving invalid data is when parsing the data to JSON format. If the data received is not properly formatted according to the JSON standard [61], an error will occur when parsing. When this happens, the SWL has to be notified that something went wrong when parsing the received data. The other areas where error handling is required, is when the data received can be parsed, but the contents deviate from the definitions stated in the SatStat communication protocol. For these kinds of errors, the SWL is notified with a more descriptive error message for easier debugging. The results of the tests performed to verify these requirements can be found in appendix A.32.

4.10.2.2.6 Beta 1 In section 4.10.2.2.4 Alpha 2, it was mentioned that the idea of a subscriber system for the HWL was planned already in Alpha 2. The initial idea was to implement a flag system marking the different sensor data as either subscribed or unsubscribed depending on the flag value. This idea was discarded because of reasons further described in the document "HWL Decisions", appendix A.24. The replacement for the flag system is the Subscriber System Template Library (SSTL). This library was implemented in version Beta 1. The requirements for Beta 1 can be found in appendix A.33.

SSTL is a library developed to simplify subscription and unsubscription of available sensor data on HWL. Upon creation of a sensor object, its corresponding data object(s) are inserted into one of the lists in the SSTL. This specific list is responsible for holding all available data. When the SWL sends a subscription request, the corresponding data object(s) are copied from the available data list to the list containing the subscribed data. When it is time to send the subscribed data, the data is fetched from this list and converted to JSON before it is sent to the SWL. Upon unsubscription of data object(s), the corresponding object(s) are removed from the subscribed data list, but they remain in the available data list. For more information on subscription or unsubscription requests, see SatStat communication protocol, appendix A.22. The class diagram for SSTL can be found in appendix A.40, and the verification report for Beta 1 is available in appendix A.35. See appendix A.34 for the class diagram covering the entire HWL for Beta 1.

4.10.2.2.7 Beta 2 Beta 2 is the final pre-release version of the diagnostics system. The HWL requirements for this version are related to efficiency and automated testing. These requirements can be found in appendix A.36.

When it comes to efficiency, the improvements done are related to how JSON data is handled internal to the HWL, as well as separating data received into two categories: instructions and requests.

Before Beta 2, the HWL had four classes for handling JSON objects: `Json_handler`, `Json_container`, `Json_object_container` and `Json_array_container`. In Beta 2, all these classes except the `Json_container` were removed. This reduction was possible as the classes complementary to the `Json_container` proved to be redundant. For more information about this change, see section 7 of the "HWL Decisions" document, appendix A.24.

In versions prior to Beta 2, all data received was considered as instructions. Some of these instructions were related to controlling the SADM, and some were related to internal configuration. The difference between those kinds of instructions is that most of the instructions concerning the SADM have to be protothreaded [84] to allow SADM operations and communication to take place somewhat simultaneously. The instructions that requires protothreading are still considered instructions, whilst the ones that don't are now considered requests. For a more detailed description on how this was implemented, see section 8 of the "HWL Decisions" document, appendix A.24.

Most of the aspects exclusively related to automated testing are handled at SWL. The only change required at HWL in order to support this feature is to provide a list of available instructions after a successful handshake. The fact that instructions and requests were separated when implementing this functionality made things a lot easier. This allowed available requests to be defined in the communication protocol, and instructions to be automatically updated on SWL when added at HWL. The identifiers of the available instructions are kept in a list which is formatted and sent to the SWL after a successful handshake.

Additionally, the `SADM.functions` class encountered a complete overhaul in Beta 2. This was due to the change of stepper motor further described in section 5.4 of the "HWL Decisions" document, appendix A.24. The HWL is no longer compatible with the old stepper motor configuration [85] [86] after this change, but it now supports the motor and driver that will be used in the final version of the SADM prototype [82] [81].

4.10.3 Virtual Prototype

A virtual prototype of the SADM was created for both testing and validation purposes. By using Microsoft HoloLens [28], the current progress in the CAD design of the SADM prototype could be validated with stakeholders. Engineers working on the design of the mechanical prototype could also inspect this virtual model in more detail to see if there were any errors or limitations to the design. All functional virtual prototypes were created using the Unity game engine [37]. The code-base for the HoloLens MR prototype can be accessed on github: <https://github.com/thmundal/SatLight>.

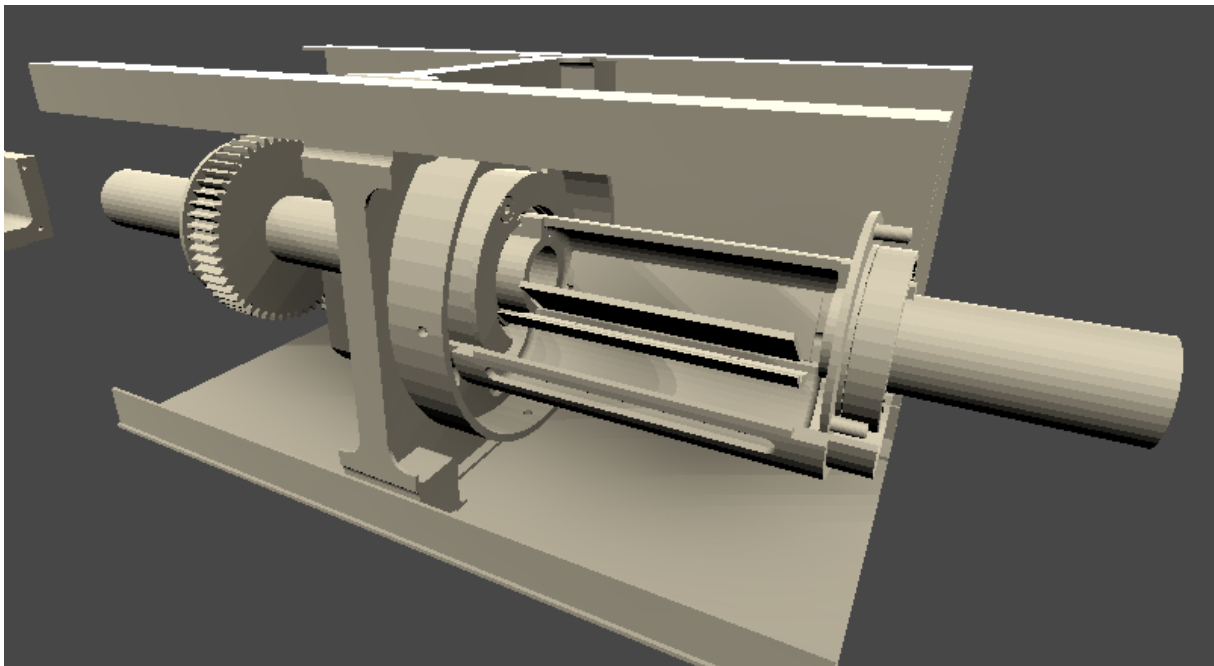


Figure 4.48: Mark 3 virtual prototype for HoloLens, exploded and expanded view

For rapid prototyping of software capabilities towards the hardware layer, and an actual SADM unit, a functional virtual prototype of the SADM attached to a 16u satellite was created separate from the HoloLens prototype. This was used in conjunction with testing and validating the functionality of the software layer when access to the hardware layer was not available. The code-base for this virtual prototype can be accessed on github: <https://github.com/thmundal/SatSim/tree/ray-system>.

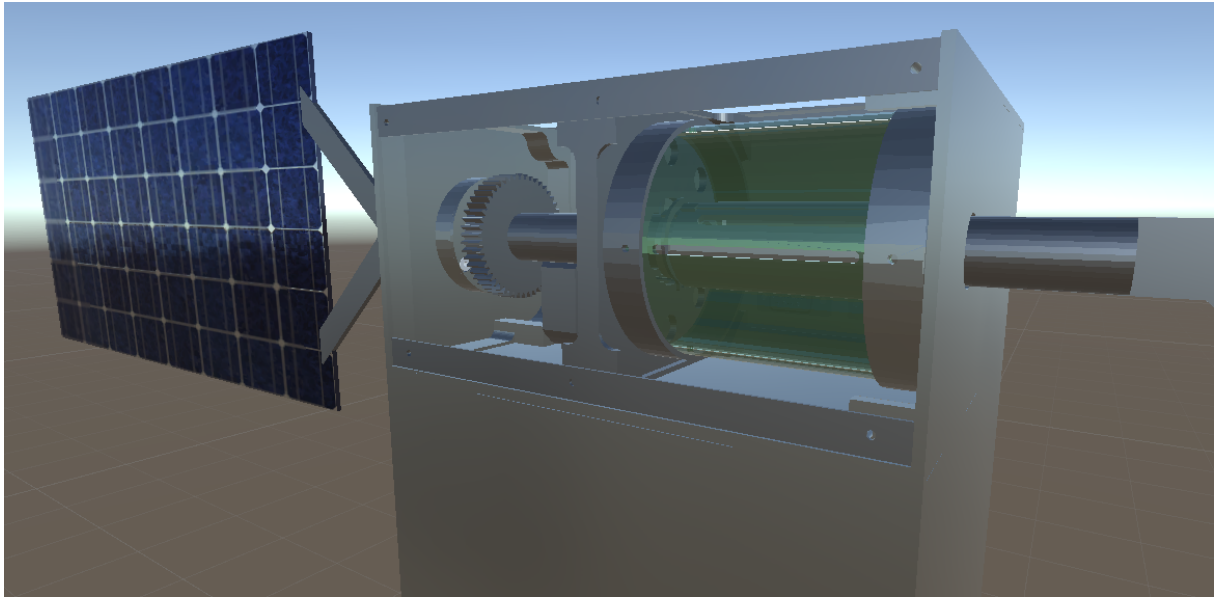


Figure 4.49: Virtual SADM prototype on 16U cubesat

The virtual prototype seen in figure 4.49 can be connected to the diagnostics system software layer and act as a virtual version of the hardware layer. With the diagnostics system, the virtual SADM can be controlled, and data can be collected about the rotary position of the SADM shaft.

Chapter 5

The Result/Final Prototype

At the end of the project, a test run of the SADM and DS was performed, as one system. A dummy of solar arrays was made specific for this test. The dummy had a weight and width similar to real ones to create the same torque required from the motor. The SADM together with the test station and rest of the DS successfully managed to be controlled and the position and torque performed by the shaft was displayed in the software.

5.1 Physical Prototype

5.1.1 Mechanical assembly

The final design of the physical prototype is shown in Figure 5.1, it is represented without the panel walls to give an impression of the design of the inside. For a detailed description of each part of Mark3, see the Design Description Appendix A.4

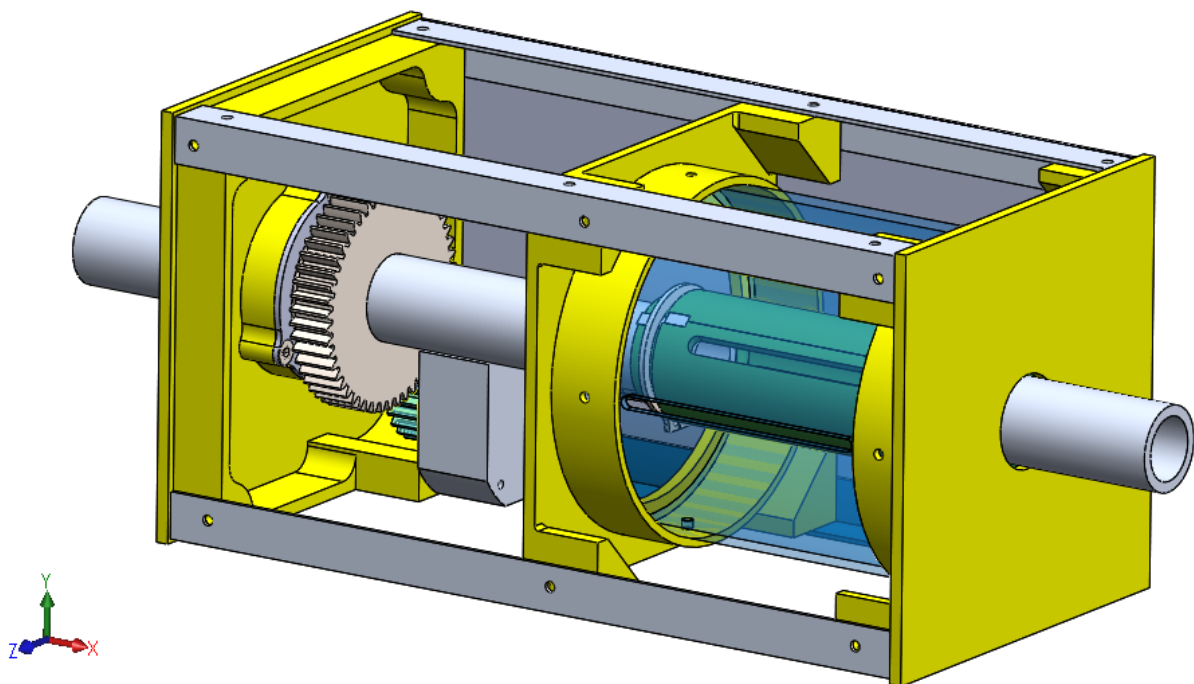


Figure 5.1: Mark3 - Space Final CAD Design

As shown in Figure 5.2 below, the final design of the prototype satisfies the requirement for weight. Additionally the outside measurements are within range of what was required. The assembling of Mark3 is relatively easy as explained in the User Manual Appendix A.5.

```

Mass properties of Mark3 Space
Configuration: Default
Coordinate system: -- default --

Mass = 1002.90 grams

Volume = 431307.93 cubic millimeters

Surface area = 435413.63 square millimeters

Center of mass: ( millimeters )
X = 97.73
Y = 46.65
Z = 75.46

Principal axes of inertia and principal moments of inertia: ( grams * square millimeters )
Taken at the center of mass.
lx = ( 1.00, 0.05, 0.05)      Px = 1751232.97
ly = (-0.07, 0.71, 0.70)    Py = 6090700.13
lz = ( 0.00, -0.71, 0.71)   Pz = 6244744.97

Moments of inertia: ( grams * square millimeters )
Taken at the center of mass and aligned with the output coordinate system.
Lxx = 1772957.13             Lxy = 210283.66             Lxz = 222673.01
Lyx = 210283.66             Lyy = 6157069.32          Lyz = 87866.35
Lzx = 222673.01             Lzy = 87866.35           Lzz = 6156651.62

Moments of inertia: ( grams * square millimeters )
Taken at the output coordinate system.
lxx = 9665942.95            lxy = 4782439.31          lxx = 7619216.62
lyx = 4782439.31            lyy = 21447737.46         lyy = 3618007.04
lzx = 7619216.62            lzy = 3618007.04         lzz = 17918629.90

```

Figure 5.2: The Mass properties of Mark3 CAD

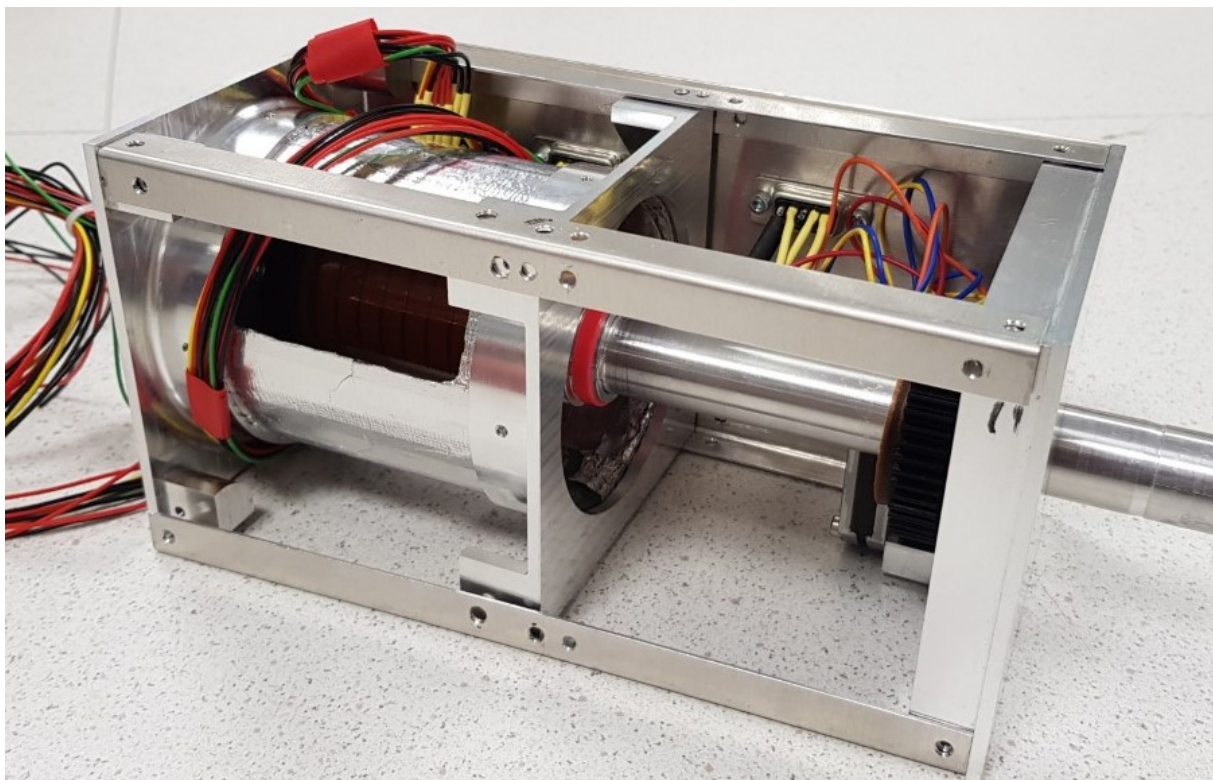


Figure 5.3: Mark3 Prototype

5.1.2 Power Transfers



Figure 5.4: Photograph of the Flex, without leads



Figure 5.5: Photograph of the Flex, with leads

One of two power transfers, including one D-sub connectors, leads between D-sub connector and twist capsule, flex and leads through shaft are completed and functional. All twenty solder joints of the flex show no significant resistance, when measured with an ordinary multi-meter and no solder joints on either side is shorted together. In figure 5.4 and figure 5.5 above, one can see the final flex with and without the power transfers leads soldered to it.

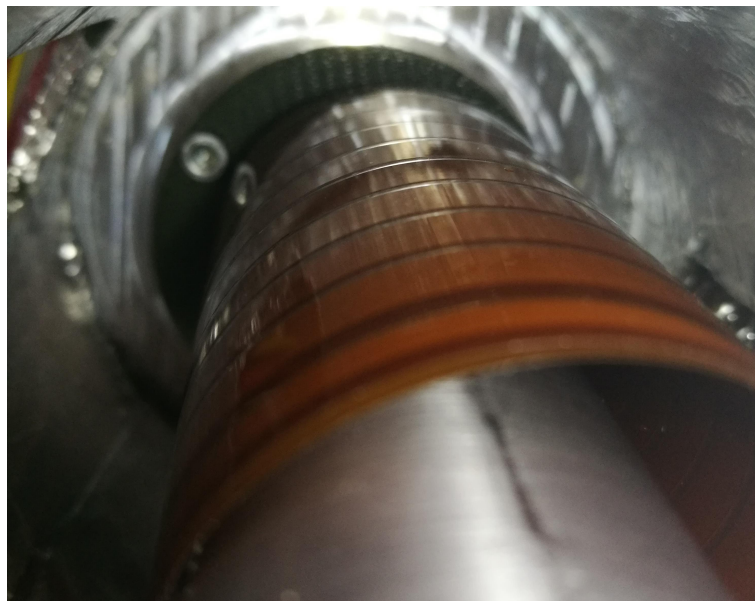


Figure 5.6: Photograph of the Flex, with leads

Figure 5.6 illustrates the flex mounted inside the Twist capsule. The mechanical behavior of the flex connected in its designated position, inside the twist capsule, is not exactly as predicted. However, it is fairly similar and shows no significant unpredictable behavior. It is able to rotate a full revolution (from -180° to $+180^\circ$), without becoming too tight or too loose. One of the two power transfers are completed and it is considered to be successful, in regards to the requirements.

5.2 Diagnostic System

The diagnostics system is made up of two main parts, the hardware layer and the software layer. The electrical components, motor driver and MCU are mounted in a control cabinet which is considered the physical part of the HWL. The firmware running on the MCU is also considered a part of the HWL. The SWL is a HMI running on a computer. It connects to the hardware layer via a USB connection, and are able to control the SADM, run automated tests and visualize feedback data for observation in graphs. When combined, the diagnostics system as a whole can be used to verify various requirements related to the SADM prototype.

5.2.1 Hardware layer

The final version of the HWL works as an interface between the SWL and the SADM testing station. The SWL and the sensors mounted on the testing station, simply connects to the MCU mounted inside the control cabinet using a few cables. All the interconnections are handled inside the cabinet. Doing so beautifies and hides the "dirty work". The different sensors mounted on the testing station are a load cell and a position sensor. A sensor for measuring temperature and humidity is also present as a part of the hardware layer, but this sensor has to be mounted inside the SADM for accurate readings.

5.2.1.1 Control Cabinet

The control cabinet has a main power supply converting 230V AC to 24V DC. This power supply provides power to the motor driver and the complementary 5V power supply. The reason for having a 5V power supply in addition to the 24V, is that the MCU, as well as some of the sensors on the testing station, requires 5V input voltage. For safety reasons, there has also been mounted a couple of fuses in the cabinet.

There is a switch mounted on the side of the control cabinet for manually controlling the SADM. This switch is wired through relays that sends an interrupt signal to the MCU when the switch is turned to either side. The interrupt signal is used to notify the firmware that the SADM is currently being manually controlled. When the MCU controls the SADM or reads sensors, the related signals are transmitted through latches before they reach the destination unit. These latches are present to provide neat wiring inside the cabinet.

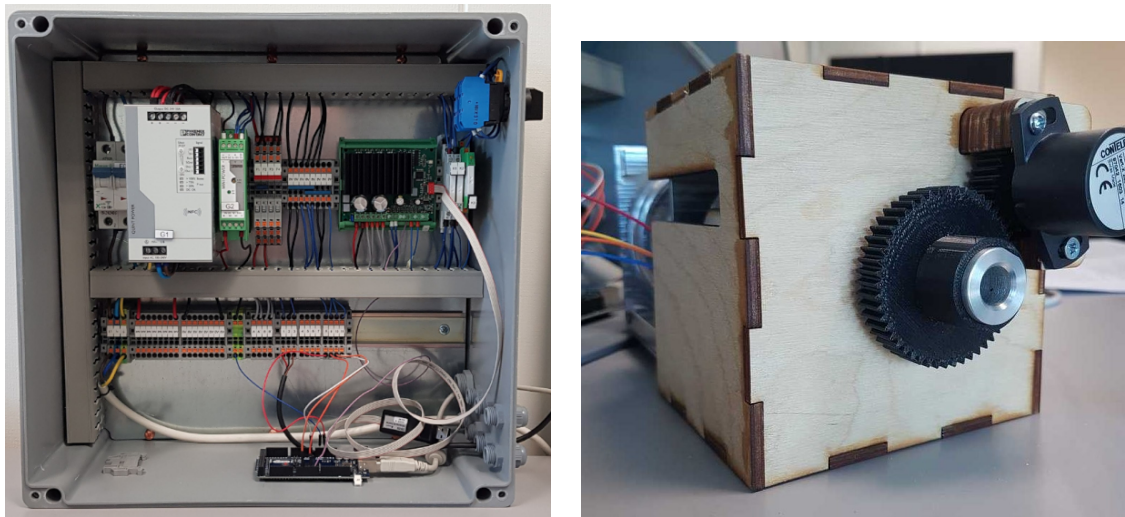


Figure 5.7: The control cabinet and the test station.

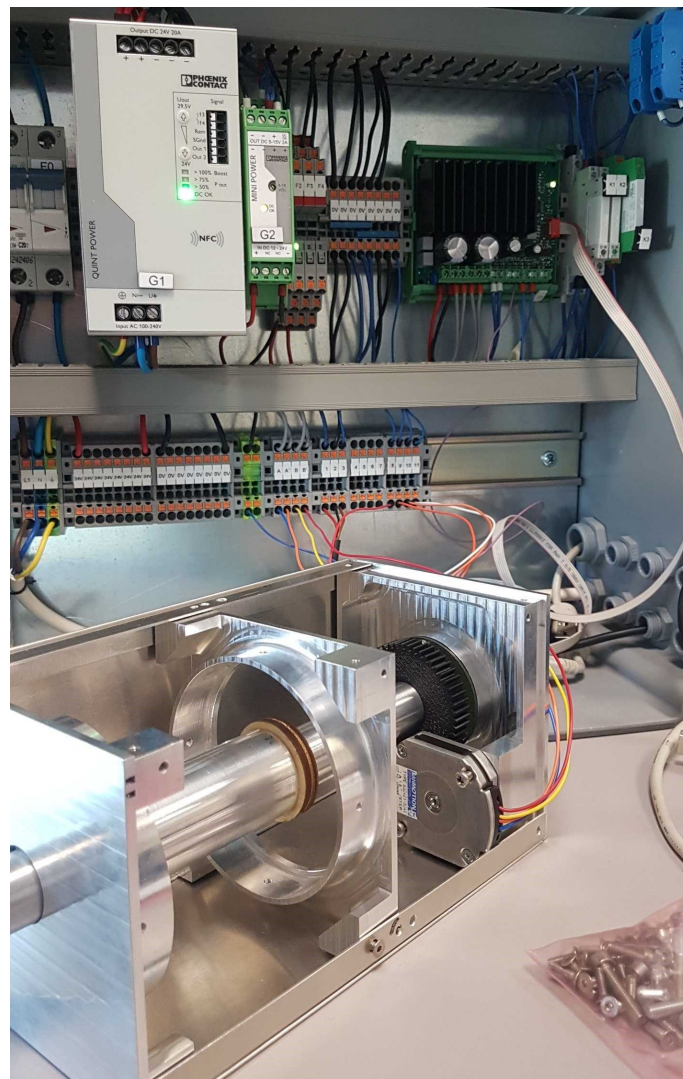


Figure 5.8: Mark3 and control cabinet

5.2.1.2 Firmware

The final version of the Firmware support real-time transmission of subscribed sensor data. The sensors available for reading are as mentioned in section 5.2.1, a load cell, a position sensor and a temperature and humidity sensor. This data is, when recorded, transmitted to the SWL following the format specified in the communication protocol, appendix A.22. The same protocol is used the other way around for transmitting instructions from the SWL to the HWL. These instructions are interpreted and executed in the same order as they are received. Some of the instructions available for SADM testing are: speed control, direction selection and destination angle. When connecting to the HWL through the SWL, a handshake protocol is carried out in order to properly establish a connection. As soon as the connection routine is finished, the user can define and execute different tests using the diagnostics system as if it the only thing involved were the SWL.

5.2.2 Software layer

The final version of the software layer acts as an HMI for the entire diagnostics system. With the software, test personell can perform tests on the SADM unit, and measure results gathered from sensor data during a test run.

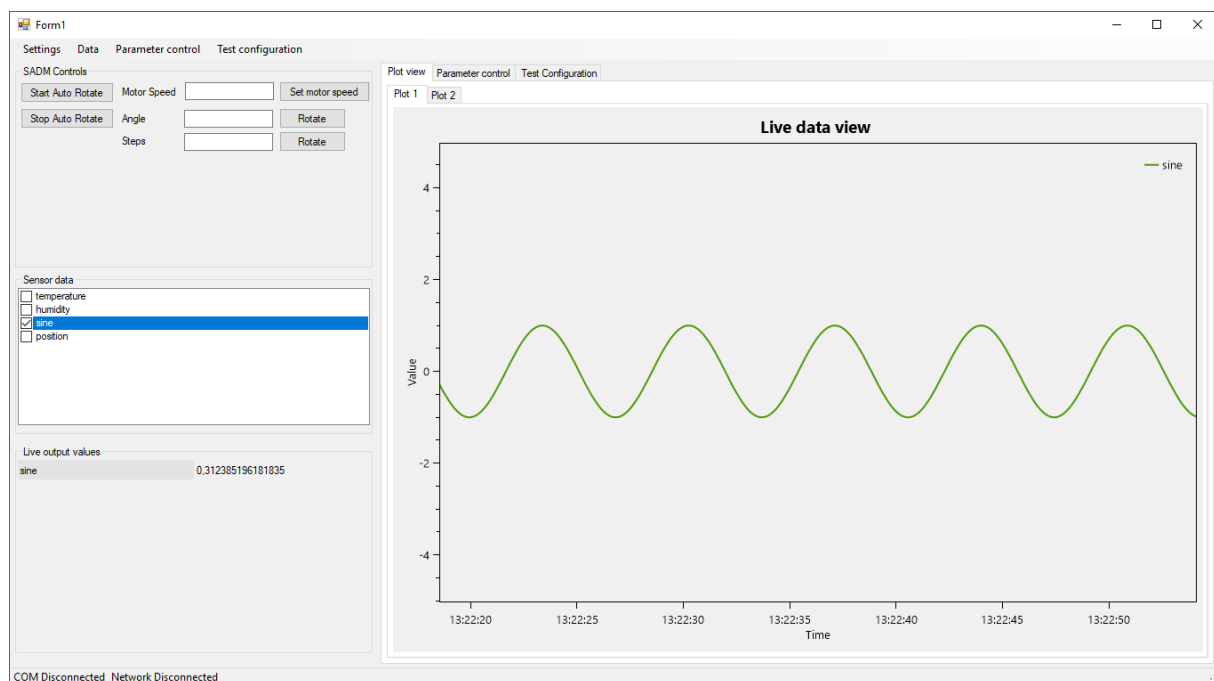


Figure 5.9: Diagnostics system software layer user interface

The software supports gathering and plotting sensor data in real-time and saving gathered data to a local database. Previously saved data can be viewed in a simple plot database viewer. The software communicates with external data sources, such as an arduino micro-controller, using a standardized communication protocol as defined in appendix A.22.

Automated testing is another feature of the software. This includes setting up a queue of instructions to be run on the SADM unit, and selecting specific data to gather while these instructions are executed.

The software can be connected to a data source either with a USB connection, or through ethernet to an external simulation.

Communication towards the software uses an established communication protocol, appendix A.22, and only support data that is formatted according to this. But as long as this protocol is followed, anyone can make a device or simulation that can interface with the software.

A user manual for the HMI software can be viewed in appendix A.23. This user manual describes how to download and install the software, and how to operate it in the way it is intended to be used.

5.2.3 The finance

As expected, the main expenses were purchasing parts and services for building prototypes and the control cabinet. Other expenses were foods for meetings/presentations and for subscription of Jira and Confluence, which was both covered by the team. The biggest expenses was covered by KDA, which was for production of mechanical parts, by Wayken RM (448 USD), and electrical components, by Elfa (2571 SEK). The total expenses for this project was approximately 8000 NOK. Mek. Lab. spent approximately 20 hours working to produce mechanical parts for this project, but this expense is not counted because Mek. Lab. is a department within KDA.

Invoices are available in appendix A.1.

5.3 Project Challenges

Challenges usually come to every team and their project. If there were no challenges then there would be lesser opportunity for learning.

At the very beginning of the project, when we got the first glimpse of what the team was supposed to be doing the next six months, there was a bit of a challenge including the software students into the given task. DSS wanted a design and prototype of a SADM that was supposed to transfer 20A current both ways through the system and rotate solar arrays. In addition to this we were given size and weight restrictions. This task was a mechanical and electrical heavy task, so the team needed to find a way of including the expertise of the software students into the project. After numerous meetings and planning, it was decided to add the diagnostic system as a part of the project. The requirements for the diagnostics system are self-defined, and was form over the course of this project.

As mentioned in section 3.4.2 some of the team members experienced some issues with the way the user stories were formed initially. Each requirement given by DSS was given one unique user story by the team members, which proved to be too generalized for describing our tasks. This problem occurred mainly for team members dealing with electrical and mechanical tasks. The reason for this is that the software students made their own requirements and the mechanical and electrical students got their requirements from DSS.

ID	Description	Importance
R3-01	The mass of the SADM shall not exceed 2kg (Harness not included)	MUST

Figure 5.10: Requirement R3-01

Figure 5.10 show an example of how the different user stories were listed in the backlog. Requirement R3-01 is a physical requirement, and is one of the most important requirements related to the physical prototype. To satisfy this specific requirement we had to decide what electrical components we needed to satisfy requirements for electrical power transfer line, we needed to design the parts of the prototype after this power transfer line and many other tasks. It proved to be difficult to satisfy a huge requirement such as R3-01, so there was need for change in the layout of the backlog and its user stories. The way we did this is further described in section 3.4.2.

When working on a tight time budget things have to be strictly planned and executed for a smooth sailing throughout a project. Numerous risks are related to this type of project, and one of these risks is related to the fact that ordered items is not always arriving in time. This was also a case in our project. Therefore it was cramped with time towards the end, considering that the ordered equipment needed testing and documentation in time for the documentation hand-in.

One of the main components in the power transfer line of the SADM is the flexible printed circuit board, or more commonly expressed as the Flex. A lot of work was put into this particular component, and when it was time to place an order, it proved to be astronomically expensive. It exceeded our budget by far, so there was a need for a plan B, which ones responsible managed to do in record time. The flex arrived two days before documentation hand-in, and the team managed to do an test assemble of the SADM just in time.

Considering the mechanical aspects, there were challenges related to making fantastic designs in SolidWorks that were not as wonderful to produce. As a major part of the verification of the design the mechanical engineering students was through mechanical workshop at KDA a number of times. The visits ended several times with the design needing changes as a result of the parts being time consuming and difficult to make. This was not ideal as a large part of the mechanical department project was to create a product that can be mass produced.

The development of the software layer of the diagnostics system was done with .NET C#. Even though this was in part familiar territory, the way the language and framework was used in this project introduced new ways of thinking about architecture and design. The codebase quickly grew large and messy, since all of the UI components required their own event handler, and all UI components was based on one class. It seemed impractical to have everything in one file, which at one point was over 1000 lines long. It was discovered that indeed .NET has solutions for this. A class can be split into partial classes and these parts can be kept in separate files. Furthermore, custom user controls can be created, encompassing UI functionality into a totally separate class that only handles one specific part of the UI. This was discovered and utilized towards the end of the project, and are techniques that would be used extensively when continuing work on this software.

Challenges in developing the software system and hardware components of the diagnostics system includes how to communicate between the software layer and hardware layer. It was observed early on that a unified communication protocol needed to be established. This protocol has seen some changes throughout the development.

Near the documentation hand-in a problem was addressed that the team had completely overlooked. A task that the team was supposed to be doing every week: make a follow-up document for the internal supervisor. The information was actually provided weekly, just not in written form. This was a critical mistake that we immediately needed to address and fix. A document of each team member's contributions was made instead in addition to describing comments on the time sheet where the team tracked all working hours.

Chapter 6

Conclusion

6.1 Summary

Maintaining structure has been really important over the course of this project. Having half the group working on the physical prototype, and the other half developing the diagnostics system, required frequent insight in each others work. The project management tool used, Jira, made it easy to keep track of what everyone at the team was working on at any given time. This ties closely in to the process model used for this project.

An iterative development process proved to be useful, but in some cases challenging. Translating heavy requirements into smaller user stories, and fragmenting them in such a way that they could be worked on iteratively was poorly understood in the beginning. The decision to rethink this structure was a good move, and helped the team make progress in a way that made more sense with respect to the agile process model chosen. Ultimately the iterative process opened up for learning from mistakes on the way, and led to rapid prototype development for inspecting the solutions before going to production stage.

The use of project management tools such as Jira and Confluence helped the team keep track of what tasks remained to be done. Other structured documents was made internally to keep track of the traceability between requirements and verification. Learning how to utilize these tools took some time, but quickly became an integral part of the daily routine.

A prototype of a Solar Array Drive Mechanism was produced with an associated testing hardware and software - the Diagnostic System release version. The prototype has the same geometrical measurements as the space graded design for an easy implementation of space grade components. The diagnostics system can be used to verify requirements that can be measured with sensors installed in the SADM, and it can be used to control the rotary position of the SADM shaft.

6.2 Reflections

The process includes sprint evaluation meetings every Monday. These meetings were done, but could have been done better. The team should have been more focused on making sure the user stories were put in a version and epic in the Jira interface. A report could have been generated after each sprint and handed over to supervisors. The team is completely new to using a model like this, as well as using tools like Jira and Confluence. Without having any data in the beginning, reports generated from Jira made little to no sense, and it was not until near the end of the project that the team could understand how the reporting worked. Learning about this gave a better perspective on how to gather information during development, and in hindsight this feature of the process tools could have been utilized a lot better.

6.3 Future Work

Considering the whole system, DS and SADM together, an idea for future work could be a control system that automatically rotates the solar arrays towards the sun, or identifies the angle that provides best possible energy absorption. For example, this can be solved with several light-sensitive sensors for identifying where the SADM (satellite) is most exposed to sunlight, rotating the solar arrays to the point at which the amount of power transferred through the SADM is largest, or by simulating the satellite's orbit around Earth and calculating its position relative to the sun. Either way, the output can be fed to a controller - PID-controller, that determines which output angle the shaft and solar arrays should have.

Solar array interface requirements were not considered in this project even though there were a fair amount of them. Therefore it would be a possibility to develop this technology further beyond the time scope of this project. The Mark2 structure would be a interesting subject to investigate further in terms of cost, design and possibility to make assembling of the SADM easier and therefore reducing assembling time and the risk of improper assembling.

More testing is also required to have finish product ready for use, both physical tests and simulations with CAD-tool. Tests that should be done are structural testing and more thorough testing of vibration and thermal stresses. The design did not take thermal expansions/compression in consideration due to lack of time for simulations.

To prevent the shaft from rotating to angles exceeding the specification, a mechanical stop should be introduced. There was made a simple drawing of this solution in SolidWorks, but it were never implemented in the prototype as other things were prioritized instead. To physically prevent the diagnostics system from ripping apart the flex, this could be a potential technology to develop further.

The final version of the software layer that came out of this project, is still very much at prototyping stage. Improvements that could be made is to overhaul the user interface design, making it easier and more intuitive to use. The code-base could be more optimized, and compartmentalized into smaller segments for better readability for anyone who would continue work

on the project.

The software should include functionality to generate test reports from automated tests that has been performed. This feature is not yet implemented due to the time available at the end of the project. This is a feature that would be beneficial to implement in future versions of the software.

As of now, adding additional sensors to the testing station, requires a firmware update at hardware layer. This means that the HWL code-base itself has to be altered in order to support reading of the new sensors. A possible solution to simplify integration of new sensors could be to add functionality for configuring additional sensors through the software layer. Additionally, there is no way to share the unit of measurement associated with the data collected. This could simply be incorporated into the stage where available sensor data is sent.

The final version of the HWL sends JSON formatted data by utilizing the `printTo` function of the `ArduinoJson` library [87]. This function delay consecutive execution by a fair amount of time, which could be reduced by sending one character at a time for each iteration of the loop function [88].

Chapter 7

Bibliography

- [1] E. Kulu, "What is nanosat?" <https://www.nanosats.eu/cubesat> [Accessed: May 05, 2019].
- [2] P. Acarnley, "Stepping motors: A guide to theory and practice 4th edition," Online, 2007, ISBN (10 digit): 0 85296 417 X ISBN (13 digit): 978-0-85296-417-0
https://doc.lagout.org/science/0_ComputerScience/8_Electronics&Robotics/SteppingMotors-AGuidetoTheoryandPractice.pdf [Accessed: May 19th, 2019].
- [3] P. Rambabu, N. E. Prasad, V. V. Kutumbarao, and R. J. H. Wanhill, "Aluminium alloys for aerospace applications," https://link.springer.com/chapter/10.1007/978-981-10-2134-3_2 [Accessed: February 21, 2019].
- [4] D. Messier, "Tiny 'cubesats' gaining bigger role in space," <https://www.space.com/29464-cubesats-space-science-missions.html> [Accessed: April 23, 2019].
- [5] Alèn, "A basic guide to nanosatellites," <https://alen.space/basic-guide-nanosatellites/> [Accessed: May 01, 2019].
- [6] NASA, "Cubesat101, basic concept and processes for first-time cubesat developers," https://www.nasa.gov/sites/default/files/atoms/files/nasa_csl_i_cubesat_101_508.pdf [Accessed: May 01, 2019].
- [7] California Polytechnic State University, "About cubesat," <http://www.cubesat.org/about> [Accessed: May 01, 2019].
- [8] —, "Cubesat design specification, rev 13," https://static1.squarespace.com/static/5418c831e4b0fa4ecac1bacd/t/56e9b62337013b6c063a655a/1458157095454/cds_rev13_final2.pdf [Accessed: April 23, 2019].
- [9] Planetary Systems Corporation, "Canisterized satellite dispenser (csd) data sheet," <https://www.planetarysystemscorp.com/wp-content/uploads/2018/08/2002337F-CSD-Data-Sheet.pdf> [Accessed: May 05, 2019].
- [10] JPL California Institute of Technology NASA, "Mars cube one (marco)," <https://www.jpl.nasa.gov/cubesat/missions/marco.php> [Accessed: May 01, 2019].
- [11] Overleaf, "Nasas technical reports server," Online, available: <https://ntrs.nasa.gov/search.jsp?R=20100021951> [Accessed May 20, 2019].
- [12] M. Passaretti and R. Hayes, "Development of a solar array drive assembly for cubesat," Online, available: <https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20100021951.pdf> [Accessed May 20, 2019].

- [13] University of South-Eastern Norway, “About usn,” <https://www.usn.no/english/about/> [Accessed: May 06, 2019].
- [14] KONGSBERG Gruppen, “Space systems,” <https://www.kongsberg.com/kda/what-we-do/space-and-surveillance/kongsberg-space/> [Accessed: May 06, 2019].
- [15] Atlassian, “Jira — issue & product tracking software — atlassian,” <https://www.atlassian.com/software/jira> [Accessed May 20, 2019].
- [16] —, “Confluence — team collaboration software — atlassian,” <https://www.atlassian.com/software/confluence> [Accessed May 20, 2019].
- [17] Slack technologies, “Where work happens — slack,” <https://slack.com/intl/en-no/> [Accessed May 20, 2019].
- [18] K. Schwaber and J. Sutherland, “The scrum guide™,” 2019, <https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf#zoom=100> [Accessed: March 04, 2019].
- [19] Mountain Goat Software, “Product owner,” 2019, <https://www.mountaingoatsoftware.com/agile/scrum/roles/product-owner> [Accessed: January 14, 2019].
- [20] G. M. Bonnema, K. T. Veenvilet, and J. F. Broenink, *Systems Design and Engineering, Facilitating Multidisciplinary Development Projects*. CRC Press, Taylor & Francis Group, 2015, 978-1-4987-5127-8 (eBook - PDF).
- [21] ECSS, “Ecscs-s-st-00-01c,” Online, available: <https://ecss.nl/standard/ecscs-s-st-00-01c-glossary-of-terms-1-october-2012/> [Accessed May 21, 2019].
- [22] J. Alblas, <https://www.scrum.org/resources/blog/managing-risk> [Accessed May 21, 2019].
- [23] M. Tomanek and J. Juricek, <https://arxiv.org/ftp/arxiv/papers/1502/1502.03595.pdf> [Accessed May 21, 2019].
- [24] Intaver institute, <http://intaver.com/risk-scores/> [Accessed May 21, 2019].
- [25] OrCAD Systems Corporation, “Cadence Release 17.2-2016,” 2016, <https://www.nordcad.dk/dk/student-forum/download-orcad.htm> [Accessed: 24.08.2018].
- [26] The GIMP Team, “GNU image manipulation program,” <http://www.gimp.org/> [Accessed: October 27, 2017].
- [27] Google Inc., “Google drive: Free cloud storage for personal use,” <https://drive.google.com> [Accessed May 20, 2019].
- [28] Microsoft, “Microsoft hololens — mixed reality technology for business,” <https://www.microsoft.com/en-us/hololens> [Accessed: May 15, 2019].

- [29] Overleaf, "About us," Online, available: <https://www.overleaf.com/about> [Accessed May 20, 2019].
- [30] Dassault Systèmes, "3d cad design software," <https://plmgroun.no/produkter/programvare/solidworks-3d-cad/> [Accessed May 20, 2019].
- [31] Visual Micro, "Arduino ide for visual studio," Online, available: <https://www.visualmicro.com/> [Accessed: May 09, 2019].
- [32] Arduino, "Arduino introduction," Online, available: Arduino, <https://www.arduino.cc/en/Guide/Introduction> [Accessed: April 21, 2019].
- [33] P. Alley, "Introductory microcontroller programming," May 2011, available: PDF, <https://web.wpi.edu/Pubs/ETD/Available/etd-042811-095908/unrestricted/alley.pdf> [Accessed: April 21, 2019].
- [34] Arduino, "Arduino - software," Online, available: <https://www.arduino.cc/en/Main/Software> [Accessed: May 09, 2019].
- [35] Microsoft, "Visual studio community," Online, available: <https://visualstudio.microsoft.com/vs/community/> [Accessed: May 09, 2019].
- [36] Microsoft Docs, "Solutions and projects in visual studio," Online, available: <https://docs.microsoft.com/en-us/visualstudio/ide/solutions-and-projects-in-visual-studio?view=vs-2019> [Accessed: May 14, 2019].
- [37] Unity Technologies, "Unity," <https://unity.com/> [Accessed: May 21, 2019].
- [38] NASA, "Small spacecraft technology state of the art," https://www.nasa.gov/sites/default/files/atoms/files/small_spacecraft_technology_state_of_the_art_2015_tagged.pdf [Accessed: May 07, 2019].
- [39] Tronrud Engineering, "Additive manufacturing," <https://www.tronrud.no/en/manufacturing/manufacturing/additive-manufacturing> [Accessed: May 07, 2019].
- [40] Innovative Solutions In Space, "2-unit cubesat structure," <https://www.isispace.nl/product/2-unit-cubesat-structure/> [Accessed: May 07, 2019].
- [41] Wikipedia, "Flexible electronics," https://en.wikipedia.org/wiki/Flexible_electronics [Accessed: March 28, 2019].
- [42] J. Sanchez, *Embedded Systems: Circuits and programming*. CRC Press, 2012, iSBN : 1-315-21687-6, 1-4398-7931-1.
- [43] NEMA, "About nema standards," Online, available: <https://www.nema.org/Standards/About-Standards/pages/default.aspx> [Accessed: May 09, 2019].
- [44] Nema Standards Publication, "Industrial control and systems," available: <https://www.nema.org/Standards/SecureDocuments/ICS16.pdf> [Accessed: May 17, 2019].

- [45] Kohara Gear Industry Co. Ltd., "Characteristics of gears," https://khkgears.net/new/gear_knowledge/introduction_to_gears/characteristics_of_gears.html [Accessed: May 1, 2019].
- [46] Kulelagerhuset AS, "Kulelagerhuset as," Online, <https://kule-as.no/> [Accessed: May 16, 2019].
- [47] R. Khurmi and J. Gupta, "A textbook of machine design, new dehli: S. chand, 2016, p. 996-997."
- [48] ECSS, "Ecscs-q-30-11c rev1 derating - eee components," <https://ecss.nl/standard/ecss-q-st-30-11c-rev-1-derating-eee-components-4-october-2011/> [Accessed: 15.05.2019].
- [49] PCB-3d, "Ipc2221," <https://tinyurl.com/y84ovchn> [Accessed: May 21, 2019].
- [50] M. M. Finckenor and K. K. de Groh, "Space environmental effects," <https://www.nasa.gov/sites/default/files/files/NP-2015-03-015-JSC.Space.Environment-ISS-Mini-Book-2015-508.pdf> [Accessed: March 15, 2019].
- [51] T. Woods, "Nasa - out of thin air," https://www.nasa.gov/topics/technology/features/atomic_oxygen.html [Accessed: March 15, 2019].
- [52] Mich from 3D printing for beginners, "What material should i use for 3d printing?" <https://3dprintingforbeginners.com/filamentprimer/> [Accessed: May 1, 2019].
- [53] A. Hutputtanasin and A. Toorian, "Cubesat design specification (cgs) revision 9," http://org.ntnu.no/studsat/docs/proposal_1/A8%20-%20Cubesat%20Design%20Specification.pdf [Accessed: May 09, 2019].
- [54] Aidan from AAA Air Support, "What are 7075, 2024 and 6061 aluminum alloys?" <https://www.aaaairsupport.com/what-are-7075-2024-and-6061-aluminum-alloys/> [Accessed: May 08, 2019].
- [55] Rocket Lab USA, "Payload user's guide," <https://www.rocketlabusa.com/assets/Uploads/Rocket-Lab-Payload-Users-Guide-7.3.pdf> [Accessed: February 15, 2019].
- [56] T. Irvine, "Damping properties of materials revision c," <https://syont.files.wordpress.com/2007/05/damping-properties-of-materials.pdf> [Accessed: May 08, 2019].
- [57] SolidWorks Help, "Mode shape/amplitude plot propertymanager," http://help.solidworks.com/2014/english/solidworks/cworks/hidd_mode_shape_amplitude.htm [Accessed: May 13, 2019].
- [58] Microsoft, ".net framework api reference," <https://docs.microsoft.com/en-us/dotnet/api/?view=netframework-4.6.1> [Accessed: March 15, 2019].

- [59] —, “C# guide,” <https://docs.microsoft.com/en-us/dotnet/csharp/> [Accessed: March 15, 2019].
- [60] D. Crockford, “Introducing json,” <http://www.json.org/> [Accessed: March 15, 2019].
- [61] ECMA International, “The json data interchange syntax,” <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf> [Accessed: March 15, 2019].
- [62] Creative Commons, “Bson (binary json) serialization,” <http://bsonspec.org/> [Accessed: March 15, 2019].
- [63] Wikipedia, “Nosql,” <https://en.wikipedia.org/wiki/NoSQL> [Accessed: March 15, 2019].
- [64] —, “Document-oriented database,” https://en.wikipedia.org/wiki/Document-oriented_database [Accessed: March 15, 2019].
- [65] Newtonsoft, “Popular high-performance json framework for .net,” 2019, <https://www.newtonsoft.com/json> [Accessed: January 14, 2019].
- [66] A. Rodríguez, “Simple, flexible, interactive & powerful data visualization for .net,” 2019, <https://lvcharts.net/> [Accessed: March 04, 2019].
- [67] M. David, “Litedb:: A .net embedded nosql database,” <http://www.litedb.org/> [Accessed: March 08, 2019].
- [68] G. V. Horrik and Øystein Bjørke, “Oxyplot: A cross-platform plotting library for .NET,” https://en.wikipedia.org/wiki/Document-oriented_database [Accessed: March 15, 2019].
- [69] Arduino, “Arduino mega 2560 rev3,” Online, available: Arduino, <https://store.arduino.cc/mega-2560-r3> [Accessed: April 21, 2019].
- [70] Contelec, “Vert-x datasheet,” https://www.elfadistelec.no/Web/Downloads/_t/ds/Vert-X_28_5V_10-90_Ub_eng_tds.pdf [Accessed: May 09, 2019].
- [71] The C++ Resources Network, “A brief description,” Online, available: <http://www.cplusplus.com/info/description/> [Accessed: May 14, 2019].
- [72] Doxygen, “Generate documentation from source code,” Online, available: <http://www.doxygen.nl/index.html> [Accessed: May 14, 2019].
- [73] J. Kew, S. Löffler, and C. Sharpsteen, “lowering the entry barrier to the tex world,” Online, available: <http://www.tug.org/texworks/> [Accessed: May 14, 2019].
- [74] B. Blanchon, “Arduinjson: Efficient json serialization for embedded c++,” Online, 2018, available: ArduinoJson, <https://arduinojson.org/> [Accessed: April 21, 2019].
- [75] R. Tillaart, “Dht library,” Online, September 2018, available: Github, <https://github.com/RobTillaart/Arduino/tree/master/libraries/DHTlib> [Accessed: April 21, 2019].

- [76] Adafruit Industries, “Adafruit dht humidity & temperature sensor library,” Online, April 2019, available: Github, <https://github.com/adafruit/DHT-sensor-library> [Accessed: April 21, 2019].
- [77] —, “Adafruit unified sensor driver,” Online, March 2019, available: Github, https://github.com/adafruit/Adafruit_Sensor [Accessed: April 21, 2019].
- [78] E. Chatzikyriakidis, “Queuearray library for arduino,” Online, available: <https://playground.arduino.cc/Code/QueueArray/> [Accessed: May 09, 2019].
- [79] Arduino, “Stepper library,” Online, available: <https://www.arduino.cc/en/reference/stepper> [Accessed: May 09, 2019].
- [80] T. Mundal and J. Skjelsbæk, “Satstat,” Online, available: <https://github.com/thmundal/SatStat> [Accessed: May 14, 2019].
- [81] Electromen, “Em-314 stepper motor driver 6a 12-24v,” Online, available: <http://www.zilvertron.com/media/electromen/EM-314.pdf> [Accessed: May 09, 2019].
- [82] SANYO DENKI, “2-phase stepping motor 42 mm square by 11.6 thin,” Online, available: http://www.farnell.com/datasheets/356617.pdf?_ga=2.66716649.692565864.1556612087-2062706234.1556612087 [Accessed: May 09, 2019].
- [83] Electromen, “User interface for em-products,” Online, available: https://electromen.com/files/8213/7811/0101/EN_em-ementool_lite.pdf [Accessed: May 14, 2019].
- [84] A. Dunkels, “Protothreads,” Online, available: <http://dunkels.com/adam/pt/> [Accessed: May 14, 2019].
- [85] Kiatronics, “28byj-48 – 5v stepper motor,” Online, available: <http://robocraft.ru/files/datasheet/28BYJ-48.pdf> [Accessed: May 09, 2019].
- [86] —, “4 phase uln2003 stepper motor driver pcb,” Online, available: <https://www.electronicoscaldas.com/datasheet/ULN2003A-PCB.pdf> [Accessed: May 09, 2019].
- [87] B. Blanchon, “JsonObject::printto(),” Online, 2018, available: ArduinoJson, <https://arduinojson.org/v5/api/jsonobject/printto/> [Accessed: May 22, 2019].
- [88] Arduino, “loop(),” Online, available: <https://www.arduino.cc/en/Reference/Loop?setlang=it> [Accessed: May 22, 2019].
- [89] S. Kalpakjian and S. Schmid, “Manufacturing engineering and technology, pearson prentice hall, 2006, p. 410.”
- [90] Wikipedia, “Small spacecraft technology state of the art,” https://en.wikipedia.org/wiki/6061_aluminium_alloy [Accessed: May 09, 2019].
- [91] PCB Universe, “Copper thickness FAQ,” <https://www.pcbuniverse.com/pcbu-tech-tips.php?a=4> [Accessed: February 25, 2019].

- [92] W. Machine, "Dip-8 package," <https://web.archive.org/web/20110813133816/http://chippackage.tecnoface.com/?d=DIP-Dual-in-line&show=pk&pk=dip8> [Accessed: May 10, 2019].

Appendix A

Appendices



Bachelor of Engineering

Project appendix

Winter semester 2019

Expenses and invoices

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Ming Kit Wong
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

Invoices for the purchases made during the project

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

Dato	Supplier	Product	Price	Currency	Purchased by
Jan-May	Atlassian	Jira Software	62.50	USD	Thomas Mundal
24.01.2019	Biltema	2 x kulelagere	80	NOK	Ming Kit Wong
09.03.2019	Kjell & Company	Kabler til skap	511	NOK	Erica Fegri
11.03.2019	Kulelagerhuset	Kulelagere til M2	545	NOK	HS News
23.04.2019	OEM Electronics AB	Motor and more	2571	SEK	KDA
23.04.2019	ELFA Distrelec	Vribryter and more	849	NOK	KDA
01.05.2019	Wayken RM	Mechanical parts	448	USD	KDA

Summary**Invoice Number: AT-69454028**

Date Issued: 14 Jan 2019

USN Bachelor group 5Krona
Kongsberg 06 3611
Norway**Billing Contact:**Thomas Mundal
USN Bachelor group 5
thmundal@gmail.com**Technical Contact:**Thomas Mundal
USN Bachelor group 5
thmundal@gmail.com**Total Paid: USD12.50****Date Paid: 14 Jan 2019****OFFICIAL RECEIPT**

Invoice Total:	USD12.50
Payment Received:	USD-12.50
Amount Now Due:	USD0.00
PayPal Account:	thmundal@gmail.com

Thank you for your payment!

For information on our refund policy and other purchasing FAQs, see
<https://www.atlassian.com/licensing/purchase-licensing>

Organization Number: 2012073

DetailsInvoice Number: **AT-69454028**

Date Issued: 14 Jan 2019

Qty	Product	Unit Price	Adjustment	Total
1	Jira Software (Cloud) 10 Users (Monthly Payments) Renewal - Site Address: usnbachelor.atlassian.net - Support Entitlement Number: SEN-13115264 - Licensed To: USN Bachelor group 5 - Billing Period: 14 Jan 2019 - 14 Feb 2019	USD10.00		USD10.00
Total Ex. Tax (USD)				USD10.00
25% VAT (USD)				USD2.50
Total Amount Paid (USD)				USD12.50

Additional Notes

The VAT exclusive total on this invoice is € 8.67 (EUR). The amount of VAT on this invoice is € 2.17 (EUR) at 25%

Licensing & SupportInvoice Number: **AT-69454028**Date Issued: 14 Jan 2019

Support Requests related to licensing or Atlassian software can be initiated at <https://www.atlassian.com/resources/support>

For support requests related to third party software please contact the third party vendor.

Usage of Atlassian products and services are subject to the Atlassian [Cloud Terms of Service](#) and [Privacy Policy](#)

Usage of third party add-ons purchased through the Atlassian Marketplace is subject to the Atlassian Marketplace [Terms of Use](#)

Specific details on Atlassian's support policy are available at <https://confluence.atlassian.com/support/atlassian-support-offerings-193299636.html>

Next steps for JIRA, Confluence, and Marketplace add-on legacy license holders are available in the [Atlassian licensing FAQ](#)

Software maintenance covers access to any support* and software product updates for your software license.

After your software maintenance period expires, you will no longer be able to access support or software updates, including security patches. Renewing your software maintenance is done purely at your discretion, and can be renewed in advance of your maintenance period expiration to ensure uninterrupted access to the support services and software and security updates.

You can continue to use your software after the active maintenance period expires. However, do keep in mind that software maintenance renewals commence from the expiration of the last active software maintenance period.

* Support covers technical service requests for implementation and configuration assistance, upgrade assistance, post-implementation product issues.

A technical service request is defined as assistance with one issue, problem, or question relating to the use or installation of a Atlassian product, regardless of the number of communications required.

Support does not cover the following:

- Development requests, including custom code development or support for third party plugins
- Database integrity or database performance issues, including tuning and optimisation of the database
- Network topology or environment issues
- Application server issues not directly related to the Atlassian product implementation, configuration or operation
- Service requests or issues referred via Atlassian forums

Summary**Invoice Number: AT-71404958**

Date Issued: 16 Feb 2019

USN Bachelor group 5Krona
Kongsberg 06 3611
Norway**Billing Contact:**Thomas Mundal
USN Bachelor group 5
thmundal@gmail.com**Technical Contact:**Thomas Mundal
USN Bachelor group 5
thmundal@gmail.com

Total Paid: USD25.00**Date Paid: 16 Feb 2019**

OFFICIAL RECEIPT

Invoice Total:	USD25.00
Payment Received:	USD-25.00
Amount Now Due:	USD0.00
PayPal Account:	thmundal@gmail.com

Thank you for your payment!

For information on our refund policy and other purchasing FAQs, see
<https://www.atlassian.com/licensing/purchase-licensing>

DetailsInvoice Number: **AT-71404958**

Date Issued: 16 Feb 2019

Qty	Product	Unit Price	Adjustment	Total
1	Confluence (Cloud) 10 Users (Monthly Payments) Renewal - Site Address: usnbachelor.atlassian.net - Support Entitlement Number: SEN-13115264 - Licensed To: USN Bachelor group 5 - Billing Period: 14 Feb 2019 - 14 Mar 2019	USD10.00		USD10.00
1	Jira Software (Cloud) 10 Users (Monthly Payments) Renewal - Site Address: usnbachelor.atlassian.net - Support Entitlement Number: SEN-13115264 - Licensed To: USN Bachelor group 5 - Billing Period: 14 Feb 2019 - 14 Mar 2019	USD10.00		USD10.00
Total Ex. Tax (USD)				USD20.00
25% VAT (USD)				USD5.00
Total Amount Paid (USD)				USD25.00

Additional Notes

The VAT exclusive total on this invoice is € 17.76 (EUR). The amount of VAT on this invoice is € 4.44 (EUR) at 25%

Licensing & SupportInvoice Number: **AT-71404958**Date Issued: 16 Feb 2019

Support Requests related to licensing or Atlassian software can be initiated at <https://www.atlassian.com/resources/support>

For support requests related to third party software please contact the third party vendor.

Usage of Atlassian products and services are subject to the Atlassian [Cloud Terms of Service](#) and [Privacy Policy](#)

Usage of third party add-ons purchased through the Atlassian Marketplace is subject to the Atlassian Marketplace [Terms of Use](#)

Specific details on Atlassian's support policy are available at <https://confluence.atlassian.com/support/atlassian-support-offerings-193299636.html>

Next steps for JIRA, Confluence, and Marketplace add-on legacy license holders are available in the [Atlassian licensing FAQ](#)

Software maintenance covers access to any support* and software product updates for your software license.

After your software maintenance period expires, you will no longer be able to access support or software updates, including security patches. Renewing your software maintenance is done purely at your discretion, and can be renewed in advance of your maintenance period expiration to ensure uninterrupted access to the support services and software and security updates.

You can continue to use your software after the active maintenance period expires. However, do keep in mind that software maintenance renewals commence from the expiration of the last active software maintenance period.

* Support covers technical service requests for implementation and configuration assistance, upgrade assistance, post-implementation product issues.

A technical service request is defined as assistance with one issue, problem, or question relating to the use or installation of a Atlassian product, regardless of the number of communications required.

Support does not cover the following:

- Development requests, including custom code development or support for third party plugins
- Database integrity or database performance issues, including tuning and optimisation of the database
- Network topology or environment issues
- Application server issues not directly related to the Atlassian product implementation, configuration or operation
- Service requests or issues referred via Atlassian forums

Summary**Invoice Number:** AT-73326728

Date Issued: 18 Mar 2019

USN Bachelor group 5Krona
Kongsberg 06 3611
Norway**Billing Contact:**Thomas Mundal
USN Bachelor group 5
thmundal@gmail.com**Technical Contact:**Thomas Mundal
USN Bachelor group 5
thmundal@gmail.com**Total Paid: USD25.00****Date Paid: 18 Mar 2019****OFFICIAL RECEIPT**

Invoice Total:	USD25.00
Payment Received:	USD-25.00
Amount Now Due:	USD0.00
PayPal Account:	thmundal@gmail.com

Thank you for your payment!

For information on our refund policy and other purchasing FAQs, see
<https://www.atlassian.com/licensing/purchase-licensing>

DetailsInvoice Number: **AT-73326728**

Date Issued: 18 Mar 2019

Qty	Product	Unit Price	Adjustment	Total
1	Confluence (Cloud) 10 Users (Monthly Payments) Renewal - Site Address: usnbachelor.atlassian.net - Support Entitlement Number: SEN-13115264 - Licensed To: USN Bachelor group 5 - Billing Period: 14 Mar 2019 - 14 Apr 2019	USD10.00		USD10.00
1	Jira Software (Cloud) 10 Users (Monthly Payments) Renewal - Site Address: usnbachelor.atlassian.net - Support Entitlement Number: SEN-13115264 - Licensed To: USN Bachelor group 5 - Billing Period: 14 Mar 2019 - 14 Apr 2019	USD10.00		USD10.00
Total Ex. Tax (USD)				USD20.00
25% VAT (USD)				USD5.00
Total Amount Paid (USD)				USD25.00

Additional Notes

The VAT exclusive total on this invoice is EUR 17.69. The amount of VAT on this invoice is EUR 4.42 at 25%

Licensing & Support

Invoice Number: AT-73326728

Date Issued: 18 Mar 2019

Support Requests related to licensing or Atlassian software can be initiated at <https://www.atlassian.com/resources/support>

For support requests related to third party software please contact the third party vendor.

Usage of Atlassian products and services are subject to the Atlassian [Cloud Terms of Service](#) and [Privacy Policy](#)

Usage of third party add-ons purchased through the Atlassian Marketplace is subject to the Atlassian Marketplace [Terms of Use](#)

Specific details on Atlassian's support policy are available at <https://confluence.atlassian.com/support/atlassian-support-offerings-193299636.html>

Next steps for JIRA, Confluence, and Marketplace add-on legacy license holders are available in the [Atlassian licensing FAQ](#)

Software maintenance covers access to any support* and software product updates for your software license.

After your software maintenance period expires, you will no longer be able to access support or software updates, including security patches. Renewing your software maintenance is done purely at your discretion, and can be renewed in advance of your maintenance period expiration to ensure uninterrupted access to the support services and software and security updates.

You can continue to use your software after the active maintenance period expires. However, do keep in mind that software maintenance renewals commence from the expiration of the last active software maintenance period.

* Support covers technical service requests for implementation and configuration assistance, upgrade assistance, post-implementation product issues.

A technical service request is defined as assistance with one issue, problem, or question relating to the use or installation of a Atlassian product, regardless of the number of communications required.

Support does not cover the following:

- Development requests, including custom code development or support for third party plugins
- Database integrity or database performance issues, including tuning and optimisation of the database
- Network topology or environment issues
- Application server issues not directly related to the Atlassian product implementation, configuration or operation
- Service requests or issues referred via Atlassian forums

Summary**Invoice Number: AT-74966186**

Date Issued: 14 Apr 2019

USN Bachelor group 5Krona
Kongsberg 06 3611
Norway**Billing Contact:**Thomas Mundal
USN Bachelor group 5
thmundal@gmail.com**Technical Contact:**Thomas Mundal
USN Bachelor group 5
thmundal@gmail.com**Total Paid: USD25.00****Date Paid: 14 Apr 2019****OFFICIAL RECEIPT**

Invoice Total:	USD25.00
Payment Received:	USD-25.00
Amount Now Due:	USD0.00
PayPal Account:	thmundal@gmail.com

Thank you for your payment!

For information on our refund policy and other purchasing FAQs, see
<https://www.atlassian.com/licensing/purchase-licensing>

DetailsInvoice Number: **AT-74966186**

Date Issued: 14 Apr 2019

Qty	Product	Unit Price	Adjustment	Total
1	Confluence (Cloud) 10 Users (Monthly Payments) Renewal - Site Address: usnbachelor.atlassian.net - Support Entitlement Number: SEN-13115264 - Licensed To: USN Bachelor group 5 - Billing Period: 14 Apr 2019 - 14 May 2019	USD10.00		USD10.00
1	Jira Software (Cloud) 10 Users (Monthly Payments) Renewal - Site Address: usnbachelor.atlassian.net - Support Entitlement Number: SEN-13115264 - Licensed To: USN Bachelor group 5 - Billing Period: 14 Apr 2019 - 14 May 2019	USD10.00		USD10.00
Total Ex. Tax (USD)				USD20.00
25% VAT (USD)				USD5.00
Total Amount Paid (USD)				USD25.00

Additional Notes

The VAT exclusive total on this invoice is EUR 17.67. The amount of VAT on this invoice is EUR 4.42 at 25%

Licensing & SupportInvoice Number: **AT-74966186**Date Issued: 14 Apr 2019

Support Requests related to licensing or Atlassian software can be initiated at <https://www.atlassian.com/resources/support>

For support requests related to third party software please contact the third party vendor.

Usage of Atlassian products and services are subject to the Atlassian [Cloud Terms of Service](#) and [Privacy Policy](#)

Usage of third party add-ons purchased through the Atlassian Marketplace is subject to the Atlassian Marketplace [Terms of Use](#)

Specific details on Atlassian's support policy are available at <https://confluence.atlassian.com/support/atlassian-support-offerings-193299636.html>

Next steps for JIRA, Confluence, and Marketplace add-on legacy license holders are available in the [Atlassian licensing FAQ](#)

Software maintenance covers access to any support* and software product updates for your software license.

After your software maintenance period expires, you will no longer be able to access support or software updates, including security patches. Renewing your software maintenance is done purely at your discretion, and can be renewed in advance of your maintenance period expiration to ensure uninterrupted access to the support services and software and security updates.

You can continue to use your software after the active maintenance period expires. However, do keep in mind that software maintenance renewals commence from the expiration of the last active software maintenance period.

* Support covers technical service requests for implementation and configuration assistance, upgrade assistance, post-implementation product issues.

A technical service request is defined as assistance with one issue, problem, or question relating to the use or installation of a Atlassian product, regardless of the number of communications required.

Support does not cover the following:

- Development requests, including custom code development or support for third party plugins
- Database integrity or database performance issues, including tuning and optimisation of the database
- Network topology or environment issues
- Application server issues not directly related to the Atlassian product implementation, configuration or operation
- Service requests or issues referred via Atlassian forums

Summary**Invoice Number: AT-76841744**

Date Issued: 14 May 2019

USN Bachelor group 5Krona
Kongsberg 06 3611
Norway**Billing Contact:**Thomas Mundal
USN Bachelor group 5
thmundal@gmail.com**Technical Contact:**Thomas Mundal
USN Bachelor group 5
thmundal@gmail.com**Total Paid: USD25.00****Date Paid: 14 May 2019****OFFICIAL RECEIPT**

Invoice Total:	USD25.00
Payment Received:	USD-25.00
Amount Now Due:	USD0.00
PayPal Account:	thmundal@gmail.com

Thank you for your payment!

For information on our refund policy and other purchasing FAQs, see
<https://www.atlassian.com/licensing/purchase-licensing>

DetailsInvoice Number: **AT-76841744**

Date Issued: 14 May 2019

Qty	Product	Unit Price	Adjustment	Total
1	Confluence (Cloud) 10 Users (Monthly Payments) Renewal - Site Address: usnbachelor.atlassian.net - Support Entitlement Number: SEN-13115264 - Licensed To: USN Bachelor group 5 - Billing Period: 14 May 2019 - 14 Jun 2019	USD10.00		USD10.00
1	Jira Software (Cloud) 10 Users (Monthly Payments) Renewal - Site Address: usnbachelor.atlassian.net - Support Entitlement Number: SEN-13115264 - Licensed To: USN Bachelor group 5 - Billing Period: 14 May 2019 - 14 Jun 2019	USD10.00		USD10.00
Total Ex. Tax (USD)				USD20.00
25% VAT (USD)				USD5.00
Total Amount Paid (USD)				USD25.00

Additional Notes

The VAT exclusive total on this invoice is EUR 17.79. The amount of VAT on this invoice is EUR 4.45 at 25%

Licensing & SupportInvoice Number: **AT-76841744**Date Issued: 14 May 2019

Support Requests related to licensing or Atlassian software can be initiated at <https://www.atlassian.com/resources/support>

For support requests related to third party software please contact the third party vendor.

Usage of Atlassian products and services are subject to the Atlassian [Cloud Terms of Service](#) and [Privacy Policy](#)

Usage of third party add-ons purchased through the Atlassian Marketplace is subject to the Atlassian Marketplace [Terms of Use](#)

Specific details on Atlassian's support policy are available at <https://confluence.atlassian.com/support/atlassian-support-offerings-193299636.html>

Next steps for JIRA, Confluence, and Marketplace add-on legacy license holders are available in the [Atlassian licensing FAQ](#)

Software maintenance covers access to any support* and software product updates for your software license.

After your software maintenance period expires, you will no longer be able to access support or software updates, including security patches. Renewing your software maintenance is done purely at your discretion, and can be renewed in advance of your maintenance period expiration to ensure uninterrupted access to the support services and software and security updates.

You can continue to use your software after the active maintenance period expires. However, do keep in mind that software maintenance renewals commence from the expiration of the last active software maintenance period.

* Support covers technical service requests for implementation and configuration assistance, upgrade assistance, post-implementation product issues.

A technical service request is defined as assistance with one issue, problem, or question relating to the use or installation of a Atlassian product, regardless of the number of communications required.

Support does not cover the following:

- Development requests, including custom code development or support for third party plugins
- Database integrity or database performance issues, including tuning and optimisation of the database
- Network topology or environment issues
- Application server issues not directly related to the Atlassian product implementation, configuration or operation
- Service requests or issues referred via Atlassian forums

ELFA Distrelec
Rosenholmveien 25
1414 TROLLÅSEN
NORWAY



KONGSBERG

BESTILLING NR. 361002145

Attn.:
E-post: norge@elfa.se

Dato: 23 apr. 2019
Saksbeh: Opheim, Nils Olav
Telefon: +47 977 53 224
E-post:
nils.olav.fiskarbekk.opheim@kongsberg.com

Vi bestiller

Pos	Ant	Enh	Part nr/Betegnelse	Enh.pris NOK	Lev.dato
10	2,00	stk	301-02-804 - YW1S-33E20 Vribryter med fjærretur	104,00	30 apr. 2019
20	1,00	stk	164-90-125 - Contelec VERT-X 2831-736-221-102 Encoder	641,00	30 apr. 2019

SUM FOR BESTILLING NOK 849,00

Kontraktbetingelser: KONGSBERG General Conditions. SELGERs generelle salgsbetingelser, unntak eller forbehold av noe slag skal ikke være gjeldende med mindre det foreligger eksplisitt skriftlig aksept fra KONGSBERG.

Leveringsbet.: FCA - Free Carrier (i samsvar med ICC, Incoterms 2010)

Betalingsbet.: Net 30 days

Leveringsadr: **Kongsberg Defence & Aerospace AS**
Kirkegårdsvn. 45, Bygg 36
3616 Kongsberg
NORWAY

Pakkseddel: Pakkseddel skal alltid følge forsendelsen

Forsendelse: Post

Forsendelsesinstrukser: Kontakt DHL Express AS for å organisere forsendelse. Tlf.: +47 810 01 345 eller www.dhl.com. Bruk kundenummer: **240747547**

Fakturering: Faktura og alle vedlegg sendes i PDF-format pr. e-post til: kda.faktura@kongsberg.com

Vennligst angi vårt bestillings- /Pos- og artikkelnr. på alle fakturaer.

BP id: 100000635

Rekv: Normann, Eirik De

KONGSBERG DEFENCE & AEROSPACE AS
Kirkegårdsveien 45 P.O.Box 1003 NO-3601 Kongsberg Norway Phone +47 32 28 82 00
Enterprise no NO 978 614 582 MVA Foretaksregisteret www.kongsberg.com

Merking: Ved krav om merking av artikler med KDA-artikkelnummer og revisjon, så er det artikkelens revisjon i henhold til denne bestilling som skal benyttes. Artikkelens revisjon er vist som en strek eller som bokstav(er) umiddelbart etter artikkelnummeret.

Alle varer og leveransedokumenter merkes "Best. nr. 361002145", vårt pos. nr og lager "Byggn.36/1 el.mek og mekanisk".

Eksportlisenser:

SELGER skal før leveranse oversende kopi av alle relevante eksportlisenser utstedt av utenlandske myndigheter. Spesielt skal SELGER oversende kopi av alle relevante ITAR-lisenser, uavhengig om disse gjelder direkte for leveransen til KONGSBERG, eller om de gjelder indirekte for deler som er inkludert eller på annen måte inngår i leveransen.

Vi ser frem til å motta Deres ordrebekreftelse på vedlagt formular.

Med vennlig hilsen
for Kongsberg Defence & Aerospace AS



Opheim, Nils Olav

**GENERAL CONDITIONS FOR MINOR PURCHASES
REVISION A DECEMBER 2014**

1. GENERAL

These General Conditions for Minor Purchases (“**General Conditions**”) shall apply unless otherwise agreed In Writing between the Parties. SELLER’s general terms and conditions, exceptions, qualifications, or other terms and conditions shall not apply, unless explicitly accepted In Writing by BUYER.

2. DEFINITIONS

“**SELLER**” shall mean the company or person stated as such in the Purchase Order. “**BUYER**” shall mean the company stated as such in the Purchase Order. “**Party**” shall mean either SELLER or BUYER. “**Parties**” shall mean both SELLER and BUYER. “**Purchase Order**” shall mean a request for the performance of the Work issued In Writing. “**Order Confirmation**” shall mean a document issued by SELLER In Writing using BUYER’s form as attached to the Purchase Order, in which SELLER declares and undertakes to perform the requested Work according to the Contract. “**Contract**” shall mean the written contract between the Parties for the performance of the Work by SELLER, consisting of the Purchase Order, these General Conditions and any other Contract Documents. “**Contract Document**” shall mean any document explicitly made part of the Contract. “**Purchase Order Price**” shall have the meaning set forth in Article 4. “**Work**” shall mean all supplies and services to be performed by SELLER for BUYER under the Contract. “**Scope of Work**” shall mean the portion of the Work to be performed by SELLER. The Scope of Work may be included in the Purchase Order or in any other of the Contract Documents. “**Specifications**” shall mean the specification of the Work, including but not limited to quality, design, and construction. The Specifications may be included in the Purchase Order or in any other of the Contract Documents. “**Delivery Schedule**” shall mean the schedule which specifies the time for delivery, performance, partial performance or Completion, as applicable. The Delivery Schedule may be included in the Purchase Order or in any other Contract Document. “**Completion**” shall mean when the Work has been performed in full, together with delivery of all applicable documentation, drawings, models, instructions, descriptions, handbooks, and manuals necessary for correct installation, operation, maintenance, and use of the Work, as specified in a Contractual Documentation Requirements List (“**CDRL**”) or any other Contract Document. “**Day**” shall mean calendar day. “**In Writing**” shall mean a document signed by BUYER and/or SELLER and submitted to the other Party either by hand, courier service, letter, fax, or pdf-attachment to an e-mail. “**Force Majeure**” shall mean an occurrence beyond the control of the Party affected impeding the performance of its obligations under the Contract, provided that such occurrence could not have been reasonably foreseen at the time of entering into the Contract and that the Party affected could not reasonably have avoided or overcome it or its consequences, including but not limited to, act of God, act of public enemy, war, blockage, strike on a national level, riot, lightning, fire, storm, flood, explosion, and Government restriction. “**Open Source**” shall mean any software, which is subject to license terms and conditions currently listed at <http://opensource.org/licenses/> or meeting the criteria listed at <http://opensource.org/docs/definition.php> or which is subject to any similar free or open source license terms. “**Intellectual Property**” shall mean all work of authorship, procedures, designs, patented and unpatented inventions and discoveries, mask works, drawings, specifications, plans of operation, technical documentation, samples, models, tools, test equipment, copyrighted works, registered and unregistered trademarks, trade secrets, know-how, and proprietary information, in all formats, languages, and versions.

3. ORDER CONFIRMATION

The Purchase Order to which these General Conditions apply is BUYER’s offer and shall become a Contract only upon full and unconditional acceptance by SELLER and in strict accordance with these General Conditions. SELLER shall within seven (7) Days after receipt of a Purchase Order return the Order Confirmation to BUYER. SELLER shall also be bound by the Purchase Order upon actual adherence thereto. If the Order Confirmation returned by SELLER to BUYER does not comply with the Purchase Order, these General Conditions or what is otherwise agreed with BUYER, BUYER reserves the right to cancel the Purchase Order without cost and/or obligation.

4. PURCHASE ORDER PRICE

The Purchase Order Price shall mean the total price specified in the Purchase Order which is subject to adjustment in accordance with Article 7 only and which shall constitute full compensation to SELLER for the Work, including all costs, expenses, taxes (excluding VAT) unless otherwise is explicitly stated in a Contract Document, duties, fees or charges of any kind incurred by or levied on SELLER related to the performance of the Purchase Order and the provision by SELLER of the Work.

5. THE WORK

SELLER shall perform the Work: (i) in conformity with the Delivery Schedule; and (ii) in conformity with the Scope of Work; and (iii) in conformity with the Specifications; and (iv) in accordance with best industry practices and standards; and (v) to achieve fitness for purpose to the extent that a particular purpose is either expressly or by implication specified in any of the Contract Documents; and (vi) in accordance with SELLER’s quality assurance system, unless otherwise required by BUYER in any of the Contract Documents; and (vii) in compliance with all applicable laws and regulations pertaining to the performance and delivery of the Work; and (viii) in a safe and secure manner with active regard to and in compliance with all of

the SELLER’s national health, environmental and safety laws, regulations, and instructions.

6. REACH–REGISTRATION, EVALUATION, AUTHORISATION AND RESTRICTIONS OF CHEMICALS

The Candidate List of Substances of Very High Concern for Authorisation (the “**Candidate List**”) is defined under the Regulation EC 1907/2006 of the European Parliament. SELLER is required to monitor the Candidate List on a regular basis and provide to BUYER information on new Substances of Very High Concern as they are added, if it applies to the Work. The current Candidate List can be found on the European Chemicals Agency (ECHA) website available at: <http://echa.europa.eu/web/guest/candidate-list-table>. If the Work contain substances found in the list, SELLER agrees to provide, at no additional cost to BUYER, information regarding identified substances name, amount contained by weight, total part weight and safe usage information. BUYER has the right to share all such information with BUYER’s customer or any other relevant third party.

7. CHANGES

BUYER may at any time instruct changes to the Delivery Schedule, Scope of Work, or Specifications of the Work required by the Contract (“**Change Order**”). If any Change Order causes an increase in the cost and/or time required for SELLER’s performance of the Work, SELLER may request an equitable adjustment to the Purchase Order Price and/or Delivery Schedule. SELLER shall without undue delay implement a Change Order when it has been received, even if the Parties have not reached a final agreement on the adjustment to the Purchase Order Price and/or the Delivery Schedule.

8. BUYER’S INFORMATION AND PROPERTY

SELLER shall keep confidential and not use BUYER’s drawings, specifications, samples, software, technical documentation, or any other data or information of a proprietary or confidential nature of the BUYER for any other purposes than performing its obligations under the Contract and in strict accordance with BUYER’s instructions.

Notwithstanding the provisions of Article 18, SELLER shall be solely responsible for loss or damage to any buyer furnished property or information in SELLER’s possession or custody, and shall at BUYER’s instruction promptly replace such at its own cost and expense or refund its value.

9. DELAYED PERFORMANCE OF THE WORK

SELLER is in delay if performance of the Work is not achieved in accordance with the Delivery Schedule for reasons other than Force Majeure. In case of delay, BUYER shall be entitled to liquidated damages amounting to five-tenths of one per cent (0.5%) of the total Purchase Order Price for each Day of delay. If only part of the Work is delayed, the liquidated damages shall be calculated on the basis of the price of the Work which cannot be used as intended due to the delay. In no event shall the liquidated damages exceed twenty per cent (20%) of the total Purchase Order Price. Liquidated damages hereunder shall be BUYER’s sole monetary remedy in the event of delay on part of the SELLER except for termination for default under Article 11 and except for gross negligence or willful misconduct on part of SELLER. If the delay is caused by gross negligence or willful misconduct on part of SELLER, BUYER may claim damages for actual losses in excess of the liquidated damages.

10. TERMINATION FOR CONVENIENCE

BUYER may at any time and for any reason (whether SELLER is in default or not) terminate the unperformed parts of the Contract in whole or in part by notice In Writing to SELLER. BUYER’s sole obligation shall be to make payment for the part of the Work delivered and shall make payment of unavoidable and documented direct costs incurred on part of SELLER relating to the terminated part of the Work.

11. TERMINATION FOR DEFAULT

BUYER shall be entitled to terminate the Contract, or any part of the Work thereof, for default with immediate effect by notice In Writing to SELLER if SELLER fails to comply with any of the requirements of the Contract and fails to remedy and cure such non-compliance within thirty (30) Days after SELLER’s receipt of written notice specifying the failure. SELLER shall diligently proceed with the performance of the Work not terminated by BUYER. BUYER shall in case of termination for default be entitled to return the terminated part of the Work and to reclaim all corresponding payments made of the Purchase Order Price. In addition, BUYER shall be entitled to compensation for the documented direct costs and expenses, hereunder any excess procurement costs resulting from the termination, subject to the limitation of liability in Article 17.

12. DELIVERY

Title shall pass to BUYER upon delivery according to the agreed trade term (INCOTERMS 2010). Unless otherwise stated in the Purchase Order or any Contract Document, terms of delivery shall be FCA SELLER’s premises. If SELLER shall install, implement, integrate, or commission the Work or parts thereof, passing of risk shall however remain with SELLER until Completion.

13. INVOICES AND PAYMENT

SELLER's invoices shall be issued according to the Purchase Order. Unless otherwise specified, invoices may not be issued before actual delivery and Completion of the Work. Payment shall be made against correct invoice(s) sixty (60) Days after receipt of invoice. BUYER reserves the right to make setoff against payments due or at issue under the Contract or any other contract with SELLER.

14. AUDIT RIGHTS

BUYER, its customer, and any representative appointed by them shall at any time, at no extra cost or expense, during normal working hours have the right to for the duration of the Contract, visit SELLER's and its subcontractors' premises for the purpose of: (i) conducting technical audits, testing and inspections; or (ii) conducting quality assurance audits, testing and inspections; or (iii) verifying that the Work is compliant with the Specifications and other requirements of the Contract. No audits, inspections, or supervisions shall exempt SELLER from its performance obligations under the Contract.

15. WARRANTY

SELLER warrants that the Work conforms to the Specifications and other requirements of the Contract, and that the Work shall be free from defects in design, material, and workmanship. SELLER's design warranty shall not apply to Work performed by SELLER pursuant to detailed designs developed, furnished, or provided by BUYER to SELLER. The warranty period shall commence upon transfer of title to BUYER and remain in effect until twenty-four (24) months after Completion of the Work (the "Warranty Period"). BUYER's warranty claims shall be presented In Writing and at the latest within thirty (30) Days following the expiry of the Warranty Period. If any non-conformity or defect in the Work or parts thereof appears within the Warranty Period, SELLER shall at its own cost and risk without undue delay, repair, rectify, replace, or re-perform the Work, after consultation with BUYER or subject to BUYER's instructions. SELLER shall reimburse BUYER all reasonable direct costs and expenses in connection with remedy of defects or non-conforming Work. Return of defective or non-conforming Work and transportation of replacement Work shall be at SELLER's cost and risk. Repaired, rectified, replaced, or re-performed Work shall be subject to the same warranty obligations as for the original Work, starting from the date of successful repair, rectification, replacement, or re-performance of the Work. In case a systematic non-conformity or defect affects similar work which SELLER has already performed to BUYER under a prior contract or agreement, the same obligation to repair, rectify, replace, or re-perform, at SELLER's own costs, shall apply to such work, provided however that such work was performed by SELLER no earlier than five (5) years before BUYER's warranty claim was presented to SELLER. A systematic non-conformity or defect shall be deemed to exist where failures occurs or may occur with a frequency, pattern, or sameness to indicate a logical regularity of occurrence. SELLER's warranty for latent defects shall extend for a period of five (5) years from Completion of the Work. For the avoidance of doubt, SELLER's obligation to repair, rectify, replace or re-perform the Work under this Article 15 shall not be subject to the limitation of liability provisions of Article 17.

16. INTELLECTUAL PROPERTY

The Parties shall retain all rights, title, and interest in or to all their respective Intellectual Property owned, developed, conceived, acquired, or obtained prior to the Contract (hereinafter referred to as "Background IP"). Intellectual Property developed, conceived, acquired or obtained by SELLER as part of the Work during the performance of the Contract (hereinafter referred to as "Foreground IP") shall be regarded as the sole Intellectual Property of BUYER unless otherwise explicitly stated in the Contract. Notwithstanding the foregoing, BUYER shall always have the right to exploit the Work by way of a nonexclusive, irrevocable, worldwide, perpetual, royalty free right to use, amend, further develop and make any sale, transfer, assignment, sublicense, distribution, incorporation or other commercial disposal of the Work in the course of its business operations. All derivative work made by BUYER based on the Work provided by SELLER to BUYER shall be regarded as the sole Intellectual Property of BUYER.

17. LIMITATION OF LIABILITY

Except for gross negligent or wilful acts or omissions of either Party, their employees, subcontractors, or representatives, neither SELLER nor BUYER shall be liable to the other for any loss of profit, loss of use, loss of production, loss of contracts, attorney's fees, or for any indirect, consequential or special damages whatsoever that may be suffered by the other. Except for gross negligent or wilful acts or omissions of either Party, their employees, subcontractors, or representatives, the total cumulative liability to the other Party whether in contract or tort shall be limited to the amount of the total Purchase Order Price. For avoidance of doubt, the limitation provisions of this Article 17 shall not apply to the indemnity provisions of Articles 8, 18 and 19.

18. INDEMNITIES

Except for gross negligent or wilful acts or omissions of the other Party, each Party shall indemnify and hold harmless the other Party, its affiliated entities, its subcontractors, their respective agents, and employees thereof from and against all claims, damages, losses, and expenses in respect of: (i) bodily injury, sickness, diseases, or death of any of its employees; and (ii) loss of or damage to its property; and (iii) bodily injury, sickness, diseases, or death, and loss of or damage to the property of any third party, caused by itself; arising from or related to the performance of the Contract.

19. THIRD PARTIES' RIGHTS

SELLER shall hold harmless, defend, and indemnify BUYER against any claim alleging that any part of the Work infringes any third party Intellectual Property Rights. SELLER warrants that the Work is free from any liens, attachments, charges, encumbrances, claims, or the like, and undertakes to hold harmless, defend, and indemnify BUYER from and against any claims related thereto.

20. OPEN SOURCE

SELLER warrants that no part of the Work include, is integrated, bundled or linked with any software that is based upon Open Source. No deviation from this warranty shall be regarded as validly accepted by BUYER; unless and to the extent SELLER: (i) explicitly and conspicuously has listed any and all Open Source based software with a brief description of their function separately; and (ii) duly provided BUYER with this information In Writing together with correct versions of all relevant license terms and conditions; and (iii) thereafter obtained an explicit complete and corresponding acceptance for the deviation In Writing from an authorised BUYER representative, included as part of each relevant Purchase Order from BUYER where such a deviation is regarded as made. SELLER shall hold harmless, defend and indemnify BUYER from and against any claims, costs, losses and damages resulting from a breach of this warranty.

21. SUPPLIER CONDUCT PRINCIPLES

SELLER commits itself to conduct its business activities in a fair, honest, responsible, ethical, and lawful manner and in strict adherence to all applicable laws and regulations governing the ethical and legal conduct of business organizations. SELLER has been provided a copy of the Supplier Conduct Principles of Kongsberg Gruppen ASA (KOG-DIR-0038) or has been informed that these Supplier Conduct Principles are available at www.kongsberg.com. The Supplier Conduct Principles shall form an integral part of the Contract, and SELLER is expected to comply with or actively pursue compliance with these principles. SELLER shall upon written request from BUYER always be obliged to: (i) document compliance with the requirements set forth above; and (ii) allow BUYER, BUYER's customer, or a third party appointed by BUYER or BUYER's customer the right to conduct such audits as it finds necessary to verify compliance with the requirements of this Article 21. For the avoidance of doubt the audit rights shall include: (a) unrestricted access to all production sites and premises; and (b) the right to communicate with and interview employees and other personnel; and (c) the right to review pertinent documentation or any other relevant material. SELLER shall ensure that any of SELLER's lower tier suppliers may also be subject to such audits as described above. The Parties shall carry their own costs incurred in relation to performance of such documentation and audit.

22. DISPUTES AND APPLICABLE LAW

The Contract shall be governed and construed by the substantive laws of Norway, excluding any choice of law rule. Any dispute, controversy or claim arising out of or relating to the Contract, or the breach thereof, shall be finally and exclusively settled by arbitration pursuant to the provisions of this Article 22, and judgment on the award rendered by the arbitrators may be entered in any court having jurisdiction thereof. Before arbitration proceedings are commenced, the Parties shall endeavour to resolve the dispute amicably through negotiations between high-level executives of the Parties. If such negotiations are not successful after a period of sixty (60) Days from a claim In Writing for such negotiations from either Party, either Party has the right to refer the dispute to final settlement through arbitration pursuant to the applicable Arbitration Rules of the United Nations Commission on International Trade Law (UNCITRAL Arbitration Rules). The International Bar Association (IBA) Rules on the Taking of Evidence in International Arbitration shall apply. The arbitration tribunal shall consist of three (3) arbitrators. The arbitration shall be conducted in the English language in Oslo, Norway. The Parties will enter into a separate written non-disclosure undertaking or agreement covering a dispute that is subject to arbitration. Notwithstanding the foregoing, each Party acknowledges that breach of the Contract may cause irreparable damage and agrees that the other Party shall be entitled to seek injunctive relief under the Contract by a competent court in any jurisdiction relevant to a breach of the Contract.

Kongsberg Defence & Aerospace AS
P.O. Box 1003
3601 Kongsberg
NORWAY

Bestilling Nr: 361002145
Dato: 23 apr. 2019

ELFA Distrelec

Attn.: Opheim, Nils Olav
E-post: nils.olav.fiskarbekk.opheim@kongsberg.co
m

Ordrebekreftelse

Vennligst signer og returner denne ordrebekreftelsen innen 7 dager til Kongsberg Defence & Aerospace AS.

Vi herved bekrefter aksept av ovennevnte bestilling, og at alle krav i bestillingen vil bli oppfylt.

Dato.....: _____

Navn(Blokkbokstaver).....: _____

Signatur.....: _____

Leverandørens ordreferanse....: _____

Leveringsdato.....: _____
Hvis ikke spesifisert på bestilling

Opprinnelsesland for varen.....: _____

Wayken RM
No. 11 Dafu Industrial Area
Guanlan Street, Longhua District
Shenzen 518110
CHINA



KONGSBERG

PURCHASE ORDER 361002151

Attn.: Ida Xing
Email: ida@waykenrm.com

Date: 01 May. 2019
Purchaser: Opheim, Nils Olav
Phone: +47 977 53 224
Email: nils.olav.fiskarbekk.opheim@kongsberg.com

We order

Pos	Qty	Unit	Part no./Description	Unit Price USD	Del.date
10	1.00	pcs	Mechanical Prototype Parts – Prototypes According to Quotation number: 19Z04011	448.00	10 May. 2019

SUM ORDER LINES USD 448.00

General Conditions: KONGSBERG General Conditions. SELLER's general terms and conditions, exceptions, qualifications, or other terms and conditions shall not apply unless explicitly accepted in writing by KONGSBERG

Delivery Terms: DAP - Delivered at Place (in accordance with Incoterms 2010)

Payment Terms: Immediately

Delivery Addr: **Kongsberg Defence & Aerospace AS**
Kirkegårdsvn. 45, Bygg 36
3616 Kongsberg
NORWAY

Packing list: A Packing list is to be included in the parcel

Shipping: Air transport

Shipping Instructions: Please contact DHL Express to arrange shipment. Use account number: **955904275**

Invoicing: Invoice and all enclosures is to be sent in PDF-format via e-mail to:
kda.faktura@kongsberg.com

In order to ensure safe transfer of payment, Bank Account Number and Wiring Instructions has to be stated on the Invoice.

BP id: 100119360

Rekv: Normann, Eirik De

KONGSBERG DEFENCE & AEROSPACE AS
Kirkegårdsveien 45 P.O.Box 1003 NO-3601 Kongsberg Norway Phone +47 32 28 82 00
Enterprise no NO 978 614 582 MVA Foretaksregisteret www.kongsberg.com

Please state our Order-, Pos- and Item- number on all invoices. Invoices with signed EUR-1 certificate or statement of EUR origin, is to be attached to the package.

Marking:

When the marking of parts with a KDA part number and revision is a requirement, the article revision in accordance with this order shall be used. The part revision is shown as a hyphen or letter(s) appearing immediately after the part number.

All packages and delivery documents are to be marked "P.O. No. 361002151", the P.O. pos. number and warehouse "Byggn.36/1 el.mek og mekanisk".

Export Licenses:

SELLER shall prior to delivery of the Work provide BUYER with copies of all applicable export licenses (both from SELLER's own country, as well as any other relevant countries) for the Work. In particular, SELLER shall provide copies of all ITAR licenses applicable for the Work, regardless if such ITAR license applies to the Work itself or to any part embedded, incorporated, or otherwise being part thereof.

We look forward to receiving your order confirmation by attached form.

Best Regards
for Kongsberg Defence & Aerospace AS



Opheim, Nils Olav

**GENERAL CONDITIONS FOR MINOR PURCHASES
REVISION A DECEMBER 2014**

1. GENERAL

These General Conditions for Minor Purchases (“**General Conditions**”) shall apply unless otherwise agreed In Writing between the Parties. SELLER’s general terms and conditions, exceptions, qualifications, or other terms and conditions shall not apply, unless explicitly accepted In Writing by BUYER.

2. DEFINITIONS

“**SELLER**” shall mean the company or person stated as such in the Purchase Order. “**BUYER**” shall mean the company stated as such in the Purchase Order. “**Party**” shall mean either SELLER or BUYER. “**Parties**” shall mean both SELLER and BUYER. “**Purchase Order**” shall mean a request for the performance of the Work issued In Writing. “**Order Confirmation**” shall mean a document issued by SELLER In Writing using BUYER’s form as attached to the Purchase Order, in which SELLER declares and undertakes to perform the requested Work according to the Contract. “**Contract**” shall mean the written contract between the Parties for the performance of the Work by SELLER, consisting of the Purchase Order, these General Conditions and any other Contract Documents. “**Contract Document**” shall mean any document explicitly made part of the Contract. “**Purchase Order Price**” shall have the meaning set forth in Article 4. “**Work**” shall mean all supplies and services to be performed by SELLER for BUYER under the Contract. “**Scope of Work**” shall mean the portion of the Work to be performed by SELLER. The Scope of Work may be included in the Purchase Order or in any other of the Contract Documents. “**Specifications**” shall mean the specification of the Work, including but not limited to quality, design, and construction. The Specifications may be included in the Purchase Order or in any other of the Contract Documents. “**Delivery Schedule**” shall mean the schedule which specifies the time for delivery, performance, partial performance or Completion, as applicable. The Delivery Schedule may be included in the Purchase Order or in any other Contract Document. “**Completion**” shall mean when the Work has been performed in full, together with delivery of all applicable documentation, drawings, models, instructions, descriptions, handbooks, and manuals necessary for correct installation, operation, maintenance, and use of the Work, as specified in a Contractual Documentation Requirements List (“**CDRL**”) or any other Contract Document. “**Day**” shall mean calendar day. “**In Writing**” shall mean a document signed by BUYER and/or SELLER and submitted to the other Party either by hand, courier service, letter, fax, or pdf-attachment to an e-mail. “**Force Majeure**” shall mean an occurrence beyond the control of the Party affected impeding the performance of its obligations under the Contract, provided that such occurrence could not have been reasonably foreseen at the time of entering into the Contract and that the Party affected could not reasonably have avoided or overcome it or its consequences, including but not limited to, act of God, act of public enemy, war, blockage, strike on a national level, riot, lightning, fire, storm, flood, explosion, and Government restriction. “**Open Source**” shall mean any software, which is subject to license terms and conditions currently listed at <http://opensource.org/licenses/> or meeting the criteria listed at <http://opensource.org/docs/definition.php> or which is subject to any similar free or open source license terms. “**Intellectual Property**” shall mean all work of authorship, procedures, designs, patented and unpatented inventions and discoveries, mask works, drawings, specifications, plans of operation, technical documentation, samples, models, tools, test equipment, copyrighted works, registered and unregistered trademarks, trade secrets, know-how, and proprietary information, in all formats, languages, and versions.

3. ORDER CONFIRMATION

The Purchase Order to which these General Conditions apply is BUYER’s offer and shall become a Contract only upon full and unconditional acceptance by SELLER and in strict accordance with these General Conditions. SELLER shall within seven (7) Days after receipt of a Purchase Order return the Order Confirmation to BUYER. SELLER shall also be bound by the Purchase Order upon actual adherence thereto. If the Order Confirmation returned by SELLER to BUYER does not comply with the Purchase Order, these General Conditions or what is otherwise agreed with BUYER, BUYER reserves the right to cancel the Purchase Order without cost and/or obligation.

4. PURCHASE ORDER PRICE

The Purchase Order Price shall mean the total price specified in the Purchase Order which is subject to adjustment in accordance with Article 7 only and which shall constitute full compensation to SELLER for the Work, including all costs, expenses, taxes (excluding VAT) unless otherwise is explicitly stated in a Contract Document, duties, fees or charges of any kind incurred by or levied on SELLER related to the performance of the Purchase Order and the provision by SELLER of the Work.

5. THE WORK

SELLER shall perform the Work: (i) in conformity with the Delivery Schedule; and (ii) in conformity with the Scope of Work; and (iii) in conformity with the Specifications; and (iv) in accordance with best industry practices and standards; and (v) to achieve fitness for purpose to the extent that a particular purpose is either expressly or by implication specified in any of the Contract Documents; and (vi) in accordance with SELLER’s quality assurance system, unless otherwise required by BUYER in any of the Contract Documents; and (vii) in compliance with all applicable laws and regulations pertaining to the performance and delivery of the Work; and

(viii) in a safe and secure manner with active regard to and in compliance with all of the SELLER’s national health, environmental and safety laws, regulations, and instructions.

6. REACH-REGISTRATION, EVALUATION, AUTHORISATION AND RESTRICTIONS OF CHEMICALS

The Candidate List of Substances of Very High Concern for Authorisation (the “**Candidate List**”) is defined under the Regulation EC 1907/2006 of the European Parliament. SELLER is required to monitor the Candidate List on a regular basis and provide to BUYER information on new Substances of Very High Concern as they are added, if it applies to the Work. The current Candidate List can be found on the European Chemicals Agency (ECHA) website available at: <http://echa.europa.eu/web/guest/candidate-list-table>. If the Work contain substances found in the list, SELLER agrees to provide, at no additional cost to BUYER, information regarding identified substances name, amount contained by weight, total part weight and safe usage information. BUYER has the right to share all such information with BUYER’s customer or any other relevant third party.

7. CHANGES

BUYER may at any time instruct changes to the Delivery Schedule, Scope of Work, or Specifications of the Work required by the Contract (“**Change Order**”). If any Change Order causes an increase in the cost and/or time required for SELLER’s performance of the Work, SELLER may request an equitable adjustment to the Purchase Order Price and/or Delivery Schedule. SELLER shall without undue delay implement a Change Order when it has been received, even if the Parties have not reached a final agreement on the adjustment to the Purchase Order Price and/or the Delivery Schedule.

8. BUYER’S INFORMATION AND PROPERTY

SELLER shall keep confidential and not use BUYER’s drawings, specifications, samples, software, technical documentation, or any other data or information of a proprietary or confidential nature of the BUYER for any other purposes than performing its obligations under the Contract and in strict accordance with BUYER’s instructions.

Notwithstanding the provisions of Article 18, SELLER shall be solely responsible for loss or damage to any buyer furnished property or information in SELLER’s possession or custody, and shall at BUYER’s instruction promptly replace such at its own cost and expense or refund its value.

9. DELAYED PERFORMANCE OF THE WORK

SELLER is in delay if performance of the Work is not achieved in accordance with the Delivery Schedule for reasons other than Force Majeure. In case of delay, BUYER shall be entitled to liquidated damages amounting to five-tenths of one per cent (0.5%) of the total Purchase Order Price for each Day of delay. If only part of the Work is delayed, the liquidated damages shall be calculated on the basis of the price of the Work which cannot be used as intended due to the delay. In no event shall the liquidated damages exceed twenty per cent (20%) of the total Purchase Order Price. Liquidated damages hereunder shall be BUYER’s sole monetary remedy in the event of delay on part of the SELLER except for termination for default under Article 11 and except for gross negligence or willful misconduct on part of SELLER. If the delay is caused by gross negligence or willful misconduct on part of SELLER, BUYER may claim damages for actual losses in excess of the liquidated damages.

10. TERMINATION FOR CONVENIENCE

BUYER may at any time and for any reason (whether SELLER is in default or not) terminate the unperformed parts of the Contract in whole or in part by notice In Writing to SELLER. BUYER’s sole obligation shall be to make payment for the part of the Work delivered and shall make payment of unavoidable and documented direct costs incurred on part of SELLER relating to the terminated part of the Work.

11. TERMINATION FOR DEFAULT

BUYER shall be entitled to terminate the Contract, or any part of the Work thereof, for default with immediate effect by notice In Writing to SELLER if SELLER fails to comply with any of the requirements of the Contract and fails to remedy and cure such non-compliance within thirty (30) Days after SELLER’s receipt of written notice specifying the failure. SELLER shall diligently proceed with the performance of the Work not terminated by BUYER. BUYER shall in case of termination for default be entitled to return the terminated part of the Work and to reclaim all corresponding payments made of the Purchase Order Price. In addition, BUYER shall be entitled to compensation for the documented direct costs and expenses, hereunder any excess reprocurement costs resulting from the termination, subject to the limitation of liability in Article 17.

12. DELIVERY

Title shall pass to BUYER upon delivery according to the agreed trade term (INCOTERMS 2010). Unless otherwise stated in the Purchase Order or any Contract Document, terms of delivery shall be FCA SELLER’s premises. If SELLER shall install, implement, integrate, or commission the Work or parts thereof, passing of risk shall however remain with SELLER until Completion.

13. INVOICES AND PAYMENT

SELLER's invoices shall be issued according to the Purchase Order. Unless otherwise specified, invoices may not be issued before actual delivery and Completion of the Work. Payment shall be made against correct invoice(s) sixty (60) Days after receipt of invoice. BUYER reserves the right to make setoff against payments due or at issue under the Contract or any other contract with SELLER.

14. AUDIT RIGHTS

BUYER, its customer, and any representative appointed by them shall at any time, at no extra cost or expense, during normal working hours have the right to for the duration of the Contract, visit SELLER's and its subcontractors' premises for the purpose of: (i) conducting technical audits, testing and inspections; or (ii) conducting quality assurance audits, testing and inspections; or (iii) verifying that the Work is compliant with the Specifications and other requirements of the Contract. No audits, inspections, or supervisions shall exempt SELLER from its performance obligations under the Contract.

15. WARRANTY

SELLER warrants that the Work conforms to the Specifications and other requirements of the Contract, and that the Work shall be free from defects in design, material, and workmanship. SELLER's design warranty shall not apply to Work performed by SELLER pursuant to detailed designs developed, furnished, or provided by BUYER to SELLER. The warranty period shall commence upon transfer of title to BUYER and remain in effect until twenty-four (24) months after Completion of the Work (the "Warranty Period"). BUYER's warranty claims shall be presented In Writing and at the latest within thirty (30) Days following the expiry of the Warranty Period. If any non-conformity or defect in the Work or parts thereof appears within the Warranty Period, SELLER shall at its own cost and risk without undue delay, repair, rectify, replace, or re-perform the Work, after consultation with BUYER or subject to BUYER's instructions. SELLER shall reimburse BUYER all reasonable direct costs and expenses in connection with remedy of defects or non-conforming Work. Return of defective or non-conforming Work and transportation of replacement Work shall be at SELLER's cost and risk. Repaired, rectified, replaced, or re-performed Work shall be subject to the same warranty obligations as for the original Work, starting from the date of successful repair, rectification, replacement, or re-performance of the Work. In case a systematic non-conformity or defect affects similar work which SELLER has already performed to BUYER under a prior contract or agreement, the same obligation to repair, rectify, replace, or re-perform, at SELLER's own costs, shall apply to such work, provided however that such work was performed by SELLER no earlier than five (5) years before BUYER's warranty claim was presented to SELLER. A systematic non-conformity or defect shall be deemed to exist where failures occurs or may occur with a frequency, pattern, or sameness to indicate a logical regularity of occurrence. SELLER's warranty for latent defects shall extend for a period of five (5) years from Completion of the Work. For the avoidance of doubt, SELLER's obligation to repair, rectify, replace or re-perform the Work under this Article 15 shall not be subject to the limitation of liability provisions of Article 17.

16. INTELLECTUAL PROPERTY

The Parties shall retain all rights, title, and interest in or to all their respective Intellectual Property owned, developed, conceived, acquired, or obtained prior to the Contract (hereinafter referred to as "**Background IP**"). Intellectual Property developed, conceived, acquired or obtained by SELLER as part of the Work during the performance of the Contract (hereinafter referred to as "**Foreground IP**") shall be regarded as the sole Intellectual Property of BUYER unless otherwise explicitly stated in the Contract. Notwithstanding the foregoing, BUYER shall always have the right to exploit the Work by way of a nonexclusive, irrevocable, worldwide, perpetual, royalty free right to use, amend, further develop and make any sale, transfer, assignment, sublicense, distribution, incorporation or other commercial disposal of the Work in the course of its business operations. All derivative work made by BUYER based on the Work provided by SELLER to BUYER shall be regarded as the sole Intellectual Property of BUYER.

17. LIMITATION OF LIABILITY

Except for gross negligent or wilful acts or omissions of either Party, their employees, subcontractors, or representatives, neither SELLER nor BUYER shall be liable to the other for any loss of profit, loss of use, loss of production, loss of contracts, attorney's fees, or for any indirect, consequential or special damages whatsoever that may be suffered by the other. Except for gross negligent or wilful acts or omissions of either Party, their employees, subcontractors, or representatives, the total cumulative liability to the other Party whether in contract or tort shall be limited to the amount of the total Purchase Order Price. For avoidance of doubt, the limitation provisions of this Article 17 shall not apply to the indemnity provisions of Articles 8, 18 and 19.

18. INDEMNITIES

Except for gross negligent or wilful acts or omissions of the other Party, each Party shall indemnify and hold harmless the other Party, its affiliated entities, its subcontractors, their respective agents, and employees thereof from and against all claims, damages, losses, and expenses in respect of: (i) bodily injury, sickness, diseases, or death of any of its employees; and (ii) loss of or damage to its property; and (iii) bodily injury, sickness, diseases, or death, and loss of or damage to the property of any third party, caused by itself; arising from or related to the performance of the Contract.

19. THIRD PARTIES' RIGHTS

SELLER shall hold harmless, defend, and indemnify BUYER against any claim alleging that any part of the Work infringes any third party Intellectual Property Rights. SELLER warrants that the Work is free from any liens, attachments, charges, encumbrances, claims, or the like, and undertakes to hold harmless, defend, and indemnify BUYER from and against any claims related thereto.

20. OPEN SOURCE

SELLER warrants that no part of the Work include, is integrated, bundled or linked with any software that is based upon Open Source. No deviation from this warranty shall be regarded as validly accepted by BUYER; unless and to the extent SELLER: (i) explicitly and conspicuously has listed any and all Open Source based software with a brief description of their function separately; and (ii) duly provided BUYER with this information In Writing together with correct versions of all relevant license terms and conditions; and (iii) thereafter obtained an explicit complete and corresponding acceptance for the deviation In Writing from an authorised BUYER representative, included as part of each relevant Purchase Order from BUYER where such a deviation is regarded as made. SELLER shall hold harmless, defend and indemnify BUYER from and against any claims, costs, losses and damages resulting from a breach of this warranty.

21. SUPPLIER CONDUCT PRINCIPLES

SELLER commits itself to conduct its business activities in a fair, honest, responsible, ethical, and lawful manner and in strict adherence to all applicable laws and regulations governing the ethical and legal conduct of business organizations. SELLER has been provided a copy of the Supplier Conduct Principles of Kongsberg Gruppen ASA (KOG-DIR-0038) or has been informed that these Supplier Conduct Principles are available at www.kongsberg.com. The Supplier Conduct Principles shall form an integral part of the Contract, and SELLER is expected to comply with or actively pursue compliance with these principles. SELLER shall upon written request from BUYER always be obliged to: (i) document compliance with the requirements set forth above; and (ii) allow BUYER, BUYER's customer, or a third party appointed by BUYER or BUYER's customer the right to conduct such audits as it finds necessary to verify compliance with the requirements of this Article 21. For the avoidance of doubt the audit rights shall include: (a) unrestricted access to all production sites and premises; and (b) the right to communicate with and interview employees and other personnel; and (c) the right to review pertinent documentation or any other relevant material. SELLER shall ensure that any of SELLER's lower tier suppliers may also be subject to such audits as described above. The Parties shall carry their own costs incurred in relation to performance of such documentation and audit.

22. DISPUTES AND APPLICABLE LAW

The Contract shall be governed and construed by the substantive laws of Norway, excluding any choice of law rule. Any dispute, controversy or claim arising out of or relating to the Contract, or the breach thereof, shall be finally and exclusively settled by arbitration pursuant to the provisions of this Article 22, and judgment on the award rendered by the arbitrators may be entered in any court having jurisdiction thereof. Before arbitration proceedings are commenced, the Parties shall endeavour to resolve the dispute amicably through negotiations between high-level executives of the Parties. If such negotiations are not successful after a period of sixty (60) Days from a claim In Writing for such negotiations from either Party, either Party has the right to refer the dispute to final settlement through arbitration pursuant to the applicable Arbitration Rules of the United Nations Commission on International Trade Law (UNCITRAL Arbitration Rules). The International Bar Association (IBA) Rules on the Taking of Evidence in International Arbitration shall apply. The arbitration tribunal shall consist of three (3) arbitrators. The arbitration shall be conducted in the English language in Oslo, Norway. The Parties will enter into a separate written non-disclosure undertaking or agreement covering a dispute that is subject to arbitration. Notwithstanding the foregoing, each Party acknowledges that breach of the Contract may cause irreparable damage and agrees that the other Party shall be entitled to seek injunctive relief under the Contract by a competent court in any jurisdiction relevant to a breach of the Contract.

Kongsberg Defence & Aerospace AS
P.O. Box 1003
3601 Kongsberg
NORWAY

Purchase Order: 361002151
Order date: 01 May. 2019

Wayken RM

Attn.: Opheim, Nils Olav
Email: nils.olav.fiskarbekk.opheim@kongsberg.com

Order Confirmation

Please Return this Order Confirmation within 7 days to Kongsberg Defence & Aerospace AS.

We hereby confirm that we accept the Purchase Order referenced above, and that we will fully comply with the requirements of the Purchase Order.

Date.....: _____

Name(Block Letters).....: _____

Signature.....: _____

Supplier Order Reference.....: _____

Delivery Date
If not specified in Purchase Order _____

Country of origin.....: _____

OEM Electronics AB
Förrådsvägen 2
SE-573 29 Tranås
SWEDEN



KONGSBERG

PURCHASE ORDER 361002144

Attn.: Matz Holmstedt
Email: info@oemelectronics.se

Date: 23 Apr. 2019
Purchaser: Opheim, Nils Olav
Phone: +47 977 53 224
Email: nils.olav.fiskarbekk.opheim@kongsberg.com

We order

Pos	Qty	Unit	Part no./Description	Unit Price SEK	Del.date
10	2.00	pcs	EM-KP72-87.5 - DIN-skinnefeste	112.00	30 Apr. 2019
20	1.00	pcs	7404201060 - Motor	508.00	30 Apr. 2019
30	1.00	pcs	EM-328 - Parametersetter	460.00	30 Apr. 2019
40	1.00	pcs	EM-314-12-24V - Driver	1,379.00	30 Apr. 2019
SUM ORDER LINES				SEK <u>2,571.00</u>	

General Conditions: KONGSBERG General Conditions. SELLER's general terms and conditions, exceptions, qualifications, or other terms and conditions shall not apply unless explicitly accepted in writing by KONGSBERG

Delivery Terms: FCA - Free Carrier (in accordance with Incoterms 2010)

Payment Terms: Net 30 days

Delivery Addr: **Kongsberg Defence & Aerospace AS**
Kirkegårdsvn. 45, Bygg 36
3616 Kongsberg
NORWAY

Packing list: A Packing list is to be included in the parcel

Shipping: Air transport

BP id: 100111763

Rekv: Normann, Eirik De

KONGSBERG DEFENCE & AEROSPACE AS
Kirkegårdsveien 45 P.O.Box 1003 NO-3601 Kongsberg Norway Phone +47 32 28 82 00
Enterprise no NO 978 614 582 MVA Foretaksregisteret www.kongsberg.com

Shipping Instructions: Please contact DHL Express to arrange shipment. Use account number: 955904275

Invoicing: Invoice and all enclosures is to be sent in PDF-format via e-mail to:
kda.faktura@kongsberg.com

In order to ensure safe transfer of payment, Bank Account Number and Wiring Instructions has to be stated on the Invoice.

Please state our Order-, Pos- and Item- number on all invoices. Invoices with signed EUR-1 certificate or statement of EUR origin, is to be attached to the package.

Marking: When the marking of parts with a KDA part number and revision is a requirement, the article revision in accordance with this order shall be used. The part revision is shown as a hyphen or letter(s) appearing immediately after the part number.

All packages and delivery documents are to be marked "P.O. No. 361002144", the P.O. pos. number and warehouse "Byggn.36/1 el.mek og mekanisk".

Export Licenses:

SELLER shall prior to delivery of the Work provide BUYER with copies of all applicable export licenses (both from SELLER's own country, as well as any other relevant countries) for the Work. In particular, SELLER shall provide copies of all ITAR licenses applicable for the Work, regardless if such ITAR license applies to the Work itself or to any part embedded, incorporated, or otherwise being part thereof.

We look forward to receiving your order confirmation by attached form.

Best Regards
for Kongsberg Defence & Aerospace AS



Opheim, Nils Olav

**GENERAL CONDITIONS FOR MINOR PURCHASES
REVISION A DECEMBER 2014**

1. GENERAL

These General Conditions for Minor Purchases (“**General Conditions**”) shall apply unless otherwise agreed In Writing between the Parties. SELLER’s general terms and conditions, exceptions, qualifications, or other terms and conditions shall not apply, unless explicitly accepted In Writing by BUYER.

2. DEFINITIONS

“**SELLER**” shall mean the company or person stated as such in the Purchase Order. “**BUYER**” shall mean the company stated as such in the Purchase Order. “**Party**” shall mean either SELLER or BUYER. “**Parties**” shall mean both SELLER and BUYER. “**Purchase Order**” shall mean a request for the performance of the Work issued In Writing. “**Order Confirmation**” shall mean a document issued by SELLER In Writing using BUYER’s form as attached to the Purchase Order, in which SELLER declares and undertakes to perform the requested Work according to the Contract. “**Contract**” shall mean the written contract between the Parties for the performance of the Work by SELLER, consisting of the Purchase Order, these General Conditions and any other Contract Documents. “**Contract Document**” shall mean any document explicitly made part of the Contract. “**Purchase Order Price**” shall have the meaning set forth in Article 4. “**Work**” shall mean all supplies and services to be performed by SELLER for BUYER under the Contract. “**Scope of Work**” shall mean the portion of the Work to be performed by SELLER. The Scope of Work may be included in the Purchase Order or in any other of the Contract Documents. “**Specifications**” shall mean the specification of the Work, including but not limited to quality, design, and construction. The Specifications may be included in the Purchase Order or in any other of the Contract Documents. “**Delivery Schedule**” shall mean the schedule which specifies the time for delivery, performance, partial performance or Completion, as applicable. The Delivery Schedule may be included in the Purchase Order or in any other Contract Document. “**Completion**” shall mean when the Work has been performed in full, together with delivery of all applicable documentation, drawings, models, instructions, descriptions, handbooks, and manuals necessary for correct installation, operation, maintenance, and use of the Work, as specified in a Contractual Documentation Requirements List (“**CDRL**”) or any other Contract Document. “**Day**” shall mean calendar day. “**In Writing**” shall mean a document signed by BUYER and/or SELLER and submitted to the other Party either by hand, courier service, letter, fax, or pdf-attachment to an e-mail. “**Force Majeure**” shall mean an occurrence beyond the control of the Party affected impeding the performance of its obligations under the Contract, provided that such occurrence could not have been reasonably foreseen at the time of entering into the Contract and that the Party affected could not reasonably have avoided or overcome it or its consequences, including but not limited to, act of God, act of public enemy, war, blockage, strike on a national level, riot, lightning, fire, storm, flood, explosion, and Government restriction. “**Open Source**” shall mean any software, which is subject to license terms and conditions currently listed at <http://opensource.org/licenses/> or meeting the criteria listed at <http://opensource.org/docs/definition.php> or which is subject to any similar free or open source license terms. “**Intellectual Property**” shall mean all work of authorship, procedures, designs, patented and unpatented inventions and discoveries, mask works, drawings, specifications, plans of operation, technical documentation, samples, models, tools, test equipment, copyrighted works, registered and unregistered trademarks, trade secrets, know-how, and proprietary information, in all formats, languages, and versions.

3. ORDER CONFIRMATION

The Purchase Order to which these General Conditions apply is BUYER’s offer and shall become a Contract only upon full and unconditional acceptance by SELLER and in strict accordance with these General Conditions. SELLER shall within seven (7) Days after receipt of a Purchase Order return the Order Confirmation to BUYER. SELLER shall also be bound by the Purchase Order upon actual adherence thereto. If the Order Confirmation returned by SELLER to BUYER does not comply with the Purchase Order, these General Conditions or what is otherwise agreed with BUYER, BUYER reserves the right to cancel the Purchase Order without cost and/or obligation.

4. PURCHASE ORDER PRICE

The Purchase Order Price shall mean the total price specified in the Purchase Order which is subject to adjustment in accordance with Article 7 only and which shall constitute full compensation to SELLER for the Work, including all costs, expenses, taxes (excluding VAT) unless otherwise is explicitly stated in a Contract Document, duties, fees or charges of any kind incurred by or levied on SELLER related to the performance of the Purchase Order and the provision by SELLER of the Work.

5. THE WORK

SELLER shall perform the Work: (i) in conformity with the Delivery Schedule; and (ii) in conformity with the Scope of Work; and (iii) in conformity with the Specifications; and (iv) in accordance with best industry practices and standards; and (v) to achieve fitness for purpose to the extent that a particular purpose is either expressly or by implication specified in any of the Contract Documents; and (vi) in accordance with SELLER’s quality assurance system, unless otherwise required by BUYER in any of the Contract Documents; and (vii) in compliance with all applicable laws and regulations pertaining to the performance and delivery of the Work; and

(viii) in a safe and secure manner with active regard to and in compliance with all of the SELLER’s national health, environmental and safety laws, regulations, and instructions.

6. REACH-REGISTRATION, EVALUATION, AUTHORISATION AND RESTRICTIONS OF CHEMICALS

The Candidate List of Substances of Very High Concern for Authorisation (the “**Candidate List**”) is defined under the Regulation EC 1907/2006 of the European Parliament. SELLER is required to monitor the Candidate List on a regular basis and provide to BUYER information on new Substances of Very High Concern as they are added, if it applies to the Work. The current Candidate List can be found on the European Chemicals Agency (ECHA) website available at: <http://echa.europa.eu/web/guest/candidate-list-table>. If the Work contain substances found in the list, SELLER agrees to provide, at no additional cost to BUYER, information regarding identified substances name, amount contained by weight, total part weight and safe usage information. BUYER has the right to share all such information with BUYER’s customer or any other relevant third party.

7. CHANGES

BUYER may at any time instruct changes to the Delivery Schedule, Scope of Work, or Specifications of the Work required by the Contract (“**Change Order**”). If any Change Order causes an increase in the cost and/or time required for SELLER’s performance of the Work, SELLER may request an equitable adjustment to the Purchase Order Price and/or Delivery Schedule. SELLER shall without undue delay implement a Change Order when it has been received, even if the Parties have not reached a final agreement on the adjustment to the Purchase Order Price and/or the Delivery Schedule.

8. BUYER’S INFORMATION AND PROPERTY

SELLER shall keep confidential and not use BUYER’s drawings, specifications, samples, software, technical documentation, or any other data or information of a proprietary or confidential nature of the BUYER for any other purposes than performing its obligations under the Contract and in strict accordance with BUYER’s instructions.

Notwithstanding the provisions of Article 18, SELLER shall be solely responsible for loss or damage to any buyer furnished property or information in SELLER’s possession or custody, and shall at BUYER’s instruction promptly replace such at its own cost and expense or refund its value.

9. DELAYED PERFORMANCE OF THE WORK

SELLER is in delay if performance of the Work is not achieved in accordance with the Delivery Schedule for reasons other than Force Majeure. In case of delay, BUYER shall be entitled to liquidated damages amounting to five-tenths of one per cent (0.5%) of the total Purchase Order Price for each Day of delay. If only part of the Work is delayed, the liquidated damages shall be calculated on the basis of the price of the Work which cannot be used as intended due to the delay. In no event shall the liquidated damages exceed twenty per cent (20%) of the total Purchase Order Price. Liquidated damages hereunder shall be BUYER’s sole monetary remedy in the event of delay on part of the SELLER except for termination for default under Article 11 and except for gross negligence or willful misconduct on part of SELLER. If the delay is caused by gross negligence or willful misconduct on part of SELLER, BUYER may claim damages for actual losses in excess of the liquidated damages.

10. TERMINATION FOR CONVENIENCE

BUYER may at any time and for any reason (whether SELLER is in default or not) terminate the unperformed parts of the Contract in whole or in part by notice In Writing to SELLER. BUYER’s sole obligation shall be to make payment for the part of the Work delivered and shall make payment of unavoidable and documented direct costs incurred on part of SELLER relating to the terminated part of the Work.

11. TERMINATION FOR DEFAULT

BUYER shall be entitled to terminate the Contract, or any part of the Work thereof, for default with immediate effect by notice In Writing to SELLER if SELLER fails to comply with any of the requirements of the Contract and fails to remedy and cure such non-compliance within thirty (30) Days after SELLER’s receipt of written notice specifying the failure. SELLER shall diligently proceed with the performance of the Work not terminated by BUYER. BUYER shall in case of termination for default be entitled to return the terminated part of the Work and to reclaim all corresponding payments made of the Purchase Order Price. In addition, BUYER shall be entitled to compensation for the documented direct costs and expenses, hereunder any excess reprocurement costs resulting from the termination, subject to the limitation of liability in Article 17.

12. DELIVERY

Title shall pass to BUYER upon delivery according to the agreed trade term (INCOTERMS 2010). Unless otherwise stated in the Purchase Order or any Contract Document, terms of delivery shall be FCA SELLER’s premises. If SELLER shall install, implement, integrate, or commission the Work or parts thereof, passing of risk shall however remain with SELLER until Completion.

13. INVOICES AND PAYMENT

SELLER's invoices shall be issued according to the Purchase Order. Unless otherwise specified, invoices may not be issued before actual delivery and Completion of the Work. Payment shall be made against correct invoice(s) sixty (60) Days after receipt of invoice. BUYER reserves the right to make setoff against payments due or at issue under the Contract or any other contract with SELLER.

14. AUDIT RIGHTS

BUYER, its customer, and any representative appointed by them shall at any time, at no extra cost or expense, during normal working hours have the right to for the duration of the Contract, visit SELLER's and its subcontractors' premises for the purpose of: (i) conducting technical audits, testing and inspections; or (ii) conducting quality assurance audits, testing and inspections; or (iii) verifying that the Work is compliant with the Specifications and other requirements of the Contract. No audits, inspections, or supervisions shall exempt SELLER from its performance obligations under the Contract.

15. WARRANTY

SELLER warrants that the Work conforms to the Specifications and other requirements of the Contract, and that the Work shall be free from defects in design, material, and workmanship. SELLER's design warranty shall not apply to Work performed by SELLER pursuant to detailed designs developed, furnished, or provided by BUYER to SELLER. The warranty period shall commence upon transfer of title to BUYER and remain in effect until twenty-four (24) months after Completion of the Work (the "Warranty Period"). BUYER's warranty claims shall be presented In Writing and at the latest within thirty (30) Days following the expiry of the Warranty Period. If any non-conformity or defect in the Work or parts thereof appears within the Warranty Period, SELLER shall at its own cost and risk without undue delay, repair, rectify, replace, or re-perform the Work, after consultation with BUYER or subject to BUYER's instructions. SELLER shall reimburse BUYER all reasonable direct costs and expenses in connection with remedy of defects or non-conforming Work. Return of defective or non-conforming Work and transportation of replacement Work shall be at SELLER's cost and risk. Repaired, rectified, replaced, or re-performed Work shall be subject to the same warranty obligations as for the original Work, starting from the date of successful repair, rectification, replacement, or re-performance of the Work. In case a systematic non-conformity or defect affects similar work which SELLER has already performed to BUYER under a prior contract or agreement, the same obligation to repair, rectify, replace, or re-perform, at SELLER's own costs, shall apply to such work, provided however that such work was performed by SELLER no earlier than five (5) years before BUYER's warranty claim was presented to SELLER. A systematic non-conformity or defect shall be deemed to exist where failures occurs or may occur with a frequency, pattern, or sameness to indicate a logical regularity of occurrence. SELLER's warranty for latent defects shall extend for a period of five (5) years from Completion of the Work. For the avoidance of doubt, SELLER's obligation to repair, rectify, replace or re-perform the Work under this Article 15 shall not be subject to the limitation of liability provisions of Article 17.

16. INTELLECTUAL PROPERTY

The Parties shall retain all rights, title, and interest in or to all their respective Intellectual Property owned, developed, conceived, acquired, or obtained prior to the Contract (hereinafter referred to as "**Background IP**"). Intellectual Property developed, conceived, acquired or obtained by SELLER as part of the Work during the performance of the Contract (hereinafter referred to as "**Foreground IP**") shall be regarded as the sole Intellectual Property of BUYER unless otherwise explicitly stated in the Contract. Notwithstanding the foregoing, BUYER shall always have the right to exploit the Work by way of a nonexclusive, irrevocable, worldwide, perpetual, royalty free right to use, amend, further develop and make any sale, transfer, assignment, sublicense, distribution, incorporation or other commercial disposal of the Work in the course of its business operations. All derivative work made by BUYER based on the Work provided by SELLER to BUYER shall be regarded as the sole Intellectual Property of BUYER.

17. LIMITATION OF LIABILITY

Except for gross negligent or wilful acts or omissions of either Party, their employees, subcontractors, or representatives, neither SELLER nor BUYER shall be liable to the other for any loss of profit, loss of use, loss of production, loss of contracts, attorney's fees, or for any indirect, consequential or special damages whatsoever that may be suffered by the other. Except for gross negligent or wilful acts or omissions of either Party, their employees, subcontractors, or representatives, the total cumulative liability to the other Party whether in contract or tort shall be limited to the amount of the total Purchase Order Price. For avoidance of doubt, the limitation provisions of this Article 17 shall not apply to the indemnity provisions of Articles 8, 18 and 19.

18. INDEMNITIES

Except for gross negligent or wilful acts or omissions of the other Party, each Party shall indemnify and hold harmless the other Party, its affiliated entities, its subcontractors, their respective agents, and employees thereof from and against all claims, damages, losses, and expenses in respect of: (i) bodily injury, sickness, diseases, or death of any of its employees; and (ii) loss of or damage to its property; and (iii) bodily injury, sickness, diseases, or death, and loss of or damage to the property of any third party, caused by itself; arising from or related to the performance of the Contract.

19. THIRD PARTIES' RIGHTS

SELLER shall hold harmless, defend, and indemnify BUYER against any claim alleging that any part of the Work infringes any third party Intellectual Property Rights. SELLER warrants that the Work is free from any liens, attachments, charges, encumbrances, claims, or the like, and undertakes to hold harmless, defend, and indemnify BUYER from and against any claims related thereto.

20. OPEN SOURCE

SELLER warrants that no part of the Work include, is integrated, bundled or linked with any software that is based upon Open Source. No deviation from this warranty shall be regarded as validly accepted by BUYER; unless and to the extent SELLER: (i) explicitly and conspicuously has listed any and all Open Source based software with a brief description of their function separately; and (ii) duly provided BUYER with this information In Writing together with correct versions of all relevant license terms and conditions; and (iii) thereafter obtained an explicit complete and corresponding acceptance for the deviation In Writing from an authorised BUYER representative, included as part of each relevant Purchase Order from BUYER where such a deviation is regarded as made. SELLER shall hold harmless, defend and indemnify BUYER from and against any claims, costs, losses and damages resulting from a breach of this warranty.

21. SUPPLIER CONDUCT PRINCIPLES

SELLER commits itself to conduct its business activities in a fair, honest, responsible, ethical, and lawful manner and in strict adherence to all applicable laws and regulations governing the ethical and legal conduct of business organizations. SELLER has been provided a copy of the Supplier Conduct Principles of Kongsberg Gruppen ASA (KOG-DIR-0038) or has been informed that these Supplier Conduct Principles are available at www.kongsberg.com. The Supplier Conduct Principles shall form an integral part of the Contract, and SELLER is expected to comply with or actively pursue compliance with these principles. SELLER shall upon written request from BUYER always be obliged to: (i) document compliance with the requirements set forth above; and (ii) allow BUYER, BUYER's customer, or a third party appointed by BUYER or BUYER's customer the right to conduct such audits as it finds necessary to verify compliance with the requirements of this Article 21. For the avoidance of doubt the audit rights shall include: (a) unrestricted access to all production sites and premises; and (b) the right to communicate with and interview employees and other personnel; and (c) the right to review pertinent documentation or any other relevant material. SELLER shall ensure that any of SELLER's lower tier suppliers may also be subject to such audits as described above. The Parties shall carry their own costs incurred in relation to performance of such documentation and audit.

22. DISPUTES AND APPLICABLE LAW

The Contract shall be governed and construed by the substantive laws of Norway, excluding any choice of law rule. Any dispute, controversy or claim arising out of or relating to the Contract, or the breach thereof, shall be finally and exclusively settled by arbitration pursuant to the provisions of this Article 22, and judgment on the award rendered by the arbitrators may be entered in any court having jurisdiction thereof. Before arbitration proceedings are commenced, the Parties shall endeavour to resolve the dispute amicably through negotiations between high-level executives of the Parties. If such negotiations are not successful after a period of sixty (60) Days from a claim In Writing for such negotiations from either Party, either Party has the right to refer the dispute to final settlement through arbitration pursuant to the applicable Arbitration Rules of the United Nations Commission on International Trade Law (UNCITRAL Arbitration Rules). The International Bar Association (IBA) Rules on the Taking of Evidence in International Arbitration shall apply. The arbitration tribunal shall consist of three (3) arbitrators. The arbitration shall be conducted in the English language in Oslo, Norway. The Parties will enter into a separate written non-disclosure undertaking or agreement covering a dispute that is subject to arbitration. Notwithstanding the foregoing, each Party acknowledges that breach of the Contract may cause irreparable damage and agrees that the other Party shall be entitled to seek injunctive relief under the Contract by a competent court in any jurisdiction relevant to a breach of the Contract.

Kongsberg Defence & Aerospace AS
P.O. Box 1003
3601 Kongsberg
NORWAY

Purchase Order: 361002144
Order date: 23 Apr. 2019

OEM Electronics AB

Attn.: Opheim, Nils Olav
Email: nils.olav.fiskarbekk.opheim@kongsberg.com

Order Confirmation

Please Return this Order Confirmation within 7 days to Kongsberg Defence & Aerospace AS.

We hereby confirm that we accept the Purchase Order referenced above, and that we will fully comply with the requirements of the Purchase Order.

Date.....: _____

Name(Block Letters).....: _____

Signature.....: _____

Supplier Order Reference.....: _____

Delivery Date
If not specified in Purchase Order _____

Country of origin.....: _____

Kjell & Company

Triaden Lørenskog Storsenter
Tel. 69 52 09 05 Orgnr: 815 420 292

Datum: 09.03.19 18:12 NR: 0520079605
Selg.: Ole Andreas M Trans: 52070839

Nr	Beskrivelse	Belop
38388	VGA-kabel 1,8 m	189,90 D
38236	Seriekabel-hunn ti	129,90 D
39322	D-sub til lodding DB Hunn 2st x 29,90	59,80 D
39326	D-sub til lodding DB Hann 2st x 34,90	69,80 D
39327	D-sub til lodding VG Hunn 2st x 29,90	59,80 D
2	Kjell&Co Plastpose Medium	2,00 D
Total kr		511,20
Nets kort		-511,20
Teller Visa/MC		
MVA D	25%	511,20 102,24

Bax: 14932216-598145
09/03/2019 18:25

Visa

*****0056-1
AID: A0000000031010
TVR: 0080008000
TSI: F800
Sted: 7796956
Ref.: 208921 232927 IB1
Resp.: 00
Overf.: 525

KJØP
NOK 511,20
GODKJENT

Ta vare på kvitteringen for reklamasjon
Kjøpsvilkår: kjell.com/kjoepvilkar
E-post: lorsnskog@kjell.com
Adresse: Kjell & Co Norway AS
Sandviksveien 176, 1337 Sandvika
Orgnr: 815 420 292 MVA

BILTEMA

Biltema Norge avd 289 Hønefoss
Hvervenveien 2E 3511-Hønefoss
Foretaksregisteret NO 882692302 MVA
Åpningstider: 8-21 (9-18) Tlf:22 22 20 22

SALGSKVITTERING

Kvitt. 17980 24.01.2019 10:24:19
Operator Dan Remi Term.nr 0201
Herav mva 15.96 Ant. varer 2

41405 KULELAGER 6202 2RS	
1 * 39.90	39.90
41408 KULELAGER 6201 2RS	
1 * 39.90	39.90

Avrundning	0.20
TOTALT	80.00
TILBAKE	10.00

KONTANT 90.00

MVA%	Grunnlag	MVA	Totalt
25.00	63.84	15.96	79.80



Åpent kjøp 30 dager mot ubrukt forpakning.
Kvittering gjelder som garantibevis og skal medbringes ved reklamasjon og tilbakekjøp.
Takk for handelen og velkommen tilbake!



Bachelor of Engineering

Project appendix

Winter semester 2019

Calculation of Inertia

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Ming Kit Wong
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document is a explanation behind the calculation of moment of inertia for the drivetrain and the solar arrays.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 22, 2019	Signature
---------------------------------------	-----------

Contents

1	Background	2
2	The process	2
3	Summary	3
4	Inertia sheet	4
5	References	7

1 Background

Similar to how mass determines the force needed to move an object linearly, moment of inertia (also called inertia only) determines the torque needed to rotate an object. This calculation was used to determine the torque required from the motor to rotate the drivetrain and the solar arrays.

$$\tau = I\alpha \quad (1)$$

Equation 1: Torque in rotational motion, a function of inertia and angular acceleration

A spreadsheet was used to calculate the inertia during the design development. Since the design varies, so does the inertia. The inertia of the final design is considered as the inertia that determines the torque needed to rotate the drivetrain and assembly. This is verified by comparing manual calculation with calculation of the CAD-design by SolidWorks.

The same source is used through the whole document [1].

2 The process

The inertia of a object is dependent on the mass, how the mass is distributed and the position of the rotational axis. The axis is placed in the middle of the solar array for this calculation. It is reasonable to assume that this is realistic since the inertia would have been four times more if it was placed on the sides, as shown in 2 4.

$$I = \frac{1}{12}Ma^2 \quad (2)$$

Equation 2: Formula for flat plate about central axis

$$I = \frac{1}{3}Ma^2 \quad (3)$$

Equation 3: Formula for flat plate with axis along edge

$$I = \frac{1}{2}M(R_1^2 + R_2^2) \quad (4)$$

Equation 4: Formula for hollow cylinder

3 Summary

The manual calculation and the calculation done by SolidWorks had a deviation of 2,7 percent. The explanation for this is most likely because the manual calculation was simplified because it consists of various dimensions and materials.

	kgm^2	gmm^2
Manual	0.0027	2679167
CAD	0.0028	2754862

Table 1: Some CubeSat design specifications

Since the shaft consists of several parts of different dimensions, the calculation is simplified by using the total mass of the rotating parts between the wings. The parameter used as radius in the calculation are based on the main shaft.

Manual calculations

Drivetrain	
Mass	200 gram
Inner radii	5 mm
Outer radii	10 mm

Mol 12500

Per solar arrays

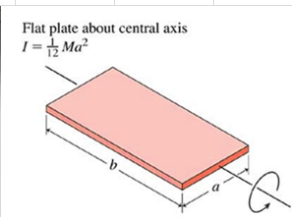
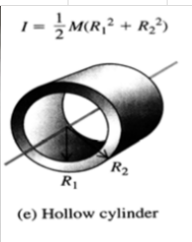
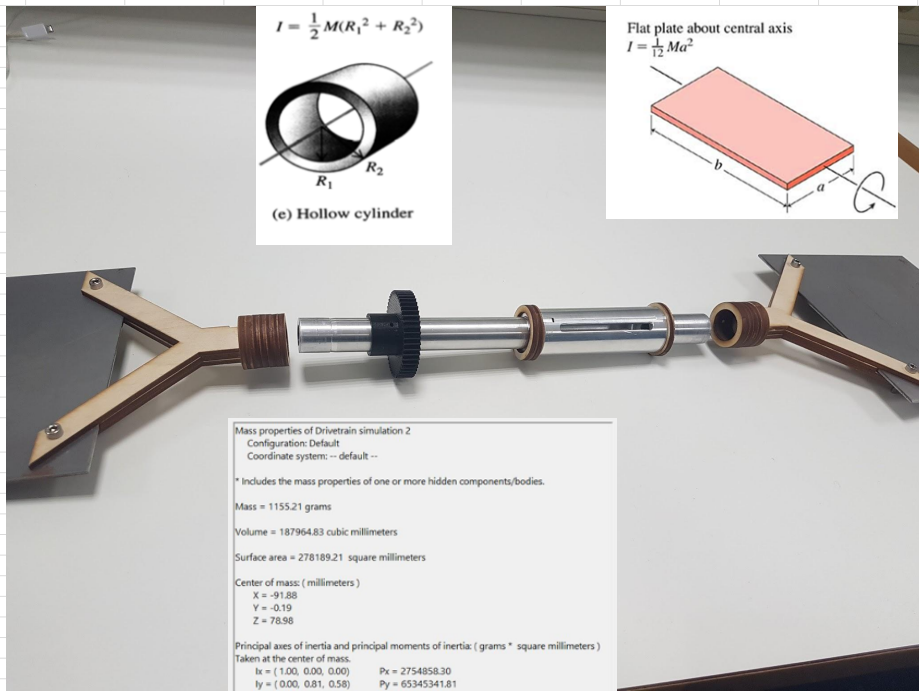
Mass 400 gram

Mol 1333333.3

The sum of drivetrain and two solar arrays 2679166.7 gmm²

Moment of Inertia comparison with simulation in

	gmm ²	Compared with simulation, %
Mark1	2260822	82,1
Mark2	1808409	65,6
Mark3, manual	2679167	97,3
Mark3, simulation	2754862	100



Mass properties of Drivetrain simulation 2
 Configuration: Default
 Coordinate system: -- default --
 * Includes the mass properties of one or more hidden components/bodies.
 Mass = 1155.21 grams
 Volume = 187964.83 cubic millimeters
 Surface area = 278189.21 square millimeters
 Center of mass: (millimeters)
 X = -91.88
 Y = -0.19
 Z = 78.98
 Principal axes of inertia and principal moments of inertia: (grams * square millimeters)
 Taken at the center of mass.
 Ix = (1.00, 0.00, 0.00) Px = 2754858.30
 Iy = (0.00, 0.81, 0.58) Py = 65345341.81
 Iz = (0.00, -0.58, 0.81) Pz = 67966672.20
 Moments of inertia: (grams * square millimeters)
 Taken at the center of mass and aligned with the output coordinate system.
 Ixx = 2754862.48 Iyy = 2924.92 Lxz = 16085.48
 Lyx = -2924.92 Lyy = 66230792.95 Lyz = 1239772.01
 Lzx = 16085.48 Lzy = 1239772.01 Lzz = 67081216.92
 Moments of inertia: (grams * square millimeters)
 Taken at the output coordinate system.
 Ixx = 9960579.92 Iyy = 16716.61 Izz = -8366390.65
 Iyx = 16716.61 Izy = 83187934.47 Iyz = 1222887.91
 Ixz = -8366390.65 Izy = 1222887.91 Izz = 76832720.07

This is a calculation of Moment of Inertia of Mark2.

Mark2 is SatLight's second prototype and is made with various materials. In this calculation, the weight of the components are based on the density of aluminium 2014 to give an indication of the moment of inertia. This is only to give a brief estimation. The final calculation will be done by a CAD-tool when the design is ready.

One can see that it is the solar arrays that creates most moment of inertia, as this is the heaviest component. The rest of the components are almost negligible compared to the SA

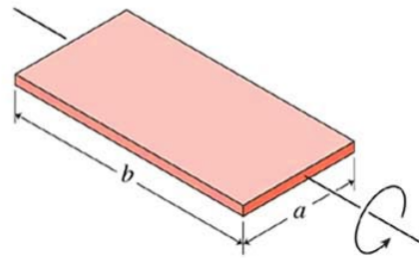
Moment of Inertia for SA when the shaft is parallel with b

Mass, M	400 g
width, a	164 mm
length, b	392 mm

MOI, SA	896533.3 gmm ²
	0.00090 kgm ²

Flat plate about central axis

$$I = \frac{1}{12} Ma^2$$

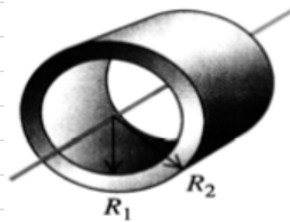


Moment of inertia for both shafts (hollow cylinder)

Mass	64.27 g
Inner radius, R1	7.5 mm
Outer radius, R2	10 mm

MOI, shaft	5021.09375 gmm ²
	0.00005021 kgm ²

$$I = \frac{1}{2} M(R_1^2 + R_2^2)$$



(e) Hollow cylinder

Moment of inertia for the gear on the shaft

Mass	33 g
Radius	24.5 mm

MOI, shaft gear	9979.2 gmm ²
	0.000009979 kgm ²

Moment of inertia for the flex shaft

Mass	5.65 g
Radius	11 mm

Moment of inertia for the gear on the motor

Mass	5.65 g	MOI, motor gear	341.83 gmm ²
Radius	11 mm		0.000003418 kgm ²

Total moment of inertia in the drivetrain

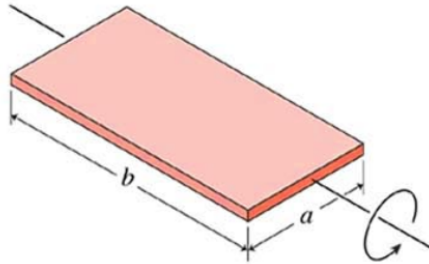
MOI, motor gear	341.83 gmm ²	When shaft is parallel with the length	1808408.74 gmm ²	0.0018 kgm ²
	0.000003418 kgm ²			

Moment of Inertia for SA when the shaft is parallel with a

Mass, M	500 g
width, a	164 mm
length, b	392 mm
MOI, SA	6402666.7 gmm ² 0.006 kgm ²

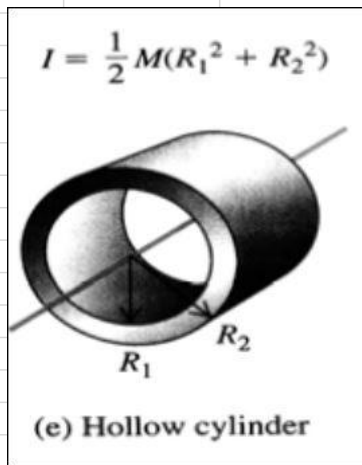
Flat plate about central axis

$$I = \frac{1}{12} Ma^2$$



Moment of Inertia for SA when the shaft is parallel with b

Mass, M	500 g
width, a	164 mm
length, b	392 mm
MOI, SA	1120666.7 gmm ² 0.00112 kgm ²



Moment of inertia for shaft (hollow cylinder)

Mass	60 g
Inner radius, R1	14.5 mm
Outer radius, R2	16.5 mm
MOI, shaft	14475 gmm ² 0.000014475 kgm ²

Moment of inertia for the gear on the shaft

Mass	25 g
Radius	20 mm

Total moment of inertia in the drivetrain

MOI, shaft gear	4978.0 gmm ²	When shaft is parallel with the width	6422155.37 gmm ²	0.0064 kgm ²
	0.000004978 kgm ²	When shaft is parallel with the length	2260822.03 gmm ²	0.0023 kgm ²

Moment of inertia for the gear on the motor

Mass	1.69 g
Radius	6.5 mm
MOI, motor gear	35.70 gmm ² 0.00000035701 kgm ²

5 References

- [1] H. Young and R. Freeman, "University physics," pearson Addison-Wesley, 2008, p. 298-299.

Bachelor of Engineering

Project appendix

Winter semester 2019

Models and sketches of configurations

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Ming Kit Wong and Rita Hogstad
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This is a document about the process of how the design was developed.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

Contents

- 1 Models and sketches of configurations** **2**
- 1.1 Independent solar arrays with two motors 2
- 1.2 Motor perpendicular to the shaft 3
- 1.3 Independent solar arrays with one motor 4
- 1.4 Synchronized and independent gear shift concept 4
- 1.5 Parallell motor and shaft 5

- 2 Conclusion** **6**

1 Models and sketches of configurations

To get ideas for solutions and concepts we used Lego to simulate our product. Lego parts were used that matched the main components of SADM to meet the requirements of components, ie shaft, motor, and twist capsule. Bearings are the only main component that was left out to simplify the model. It was considered unnecessarily to include this because bearings are always mounted on the shaft. Lego solar cells were used to illustrate the movements, although the solar cells are not part of our product.

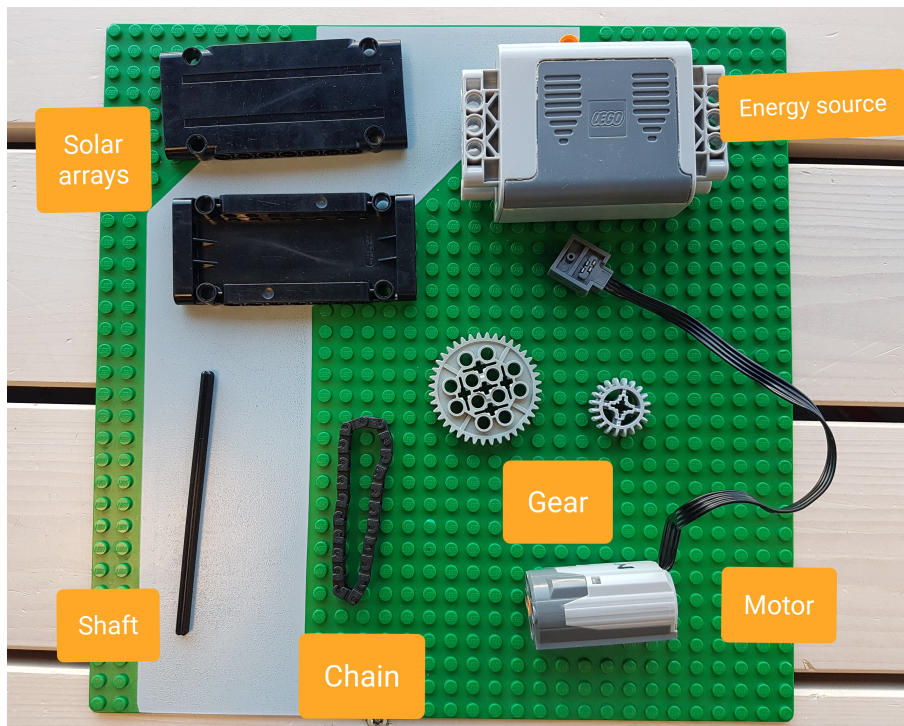


Figure 1: the main components in Lego

1.1 Independent solar arrays with two motors

Having two motors can make the solar arrays move independently to optimize the production of energy and to increase the redundancy. A satellite that is no longer energy positive will quickly be useless and compromise the mission. The drawback with this concept is that a motor is very expensive. This concept is only recommended if the customer can afford two motors and two twist capsules to reduce the risk of not producing power.

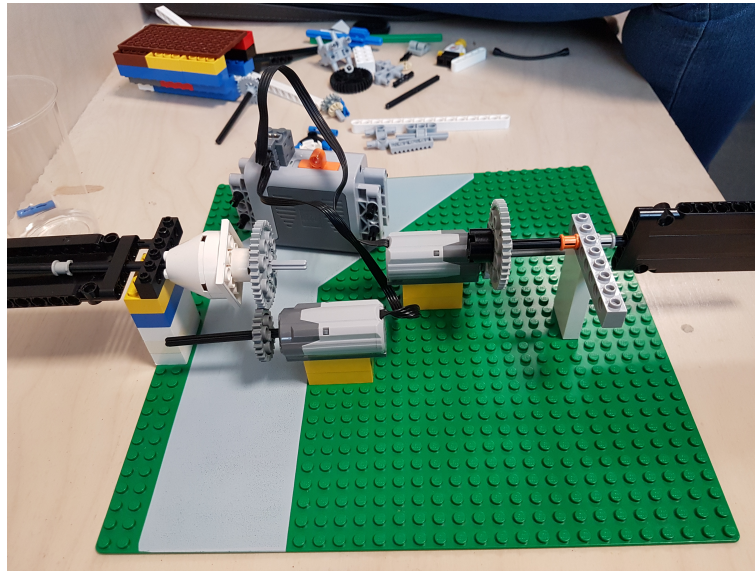


Figure 2: One motor on each solar array

Fig 2: Independent solar arrays with two motors

1.2 Motor perpendicular to the shaft

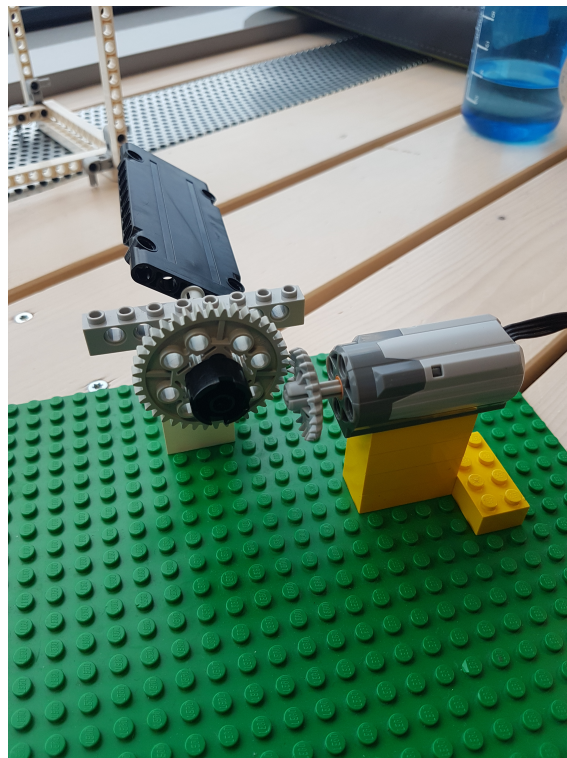


Figure 3: Motor perpendicular to the shaft

This concept uses one motor to run both solar arrays, with the motor perpendicular to the shaft. This is a cost- and energy efficient way to drive the solar arrays. Having the motor perpendicular to the shaft gives a lot of space for the twist capsule, but requires a small motor that is still strong enough to perform as expected.

1.3 Independent solar arrays with one motor

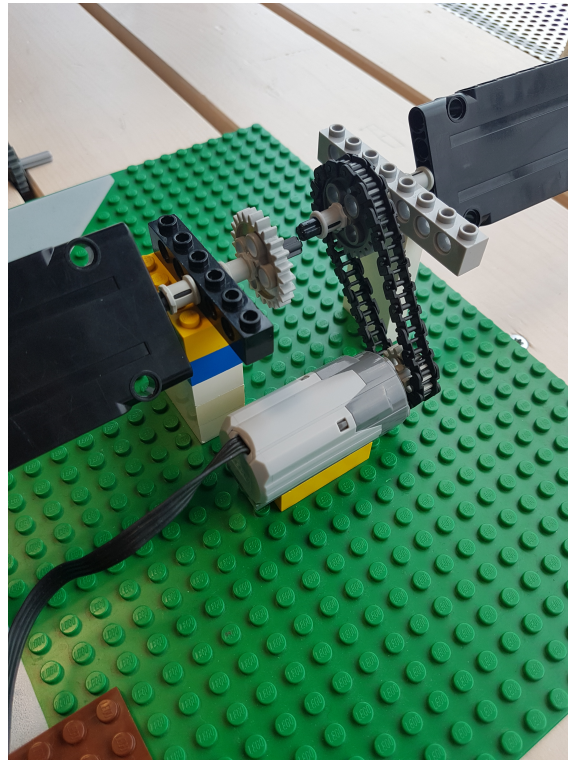


Figure 4: Independent solar arrays with one motor

The idea behind this concept is to be able to move the solar arrays independently of each other to maybe optimize the production of energy. This concept is built as two separate systems, one on each side of our product. The gears are driven by a chain from the motor, and placed close to each other in the middle of the units. The idea is to change the gears like on a bicycle when a solar array needs adjustment. The pros with having two separate systems is the redundancy in case a gear or bearing should fail to rotate the solar array. The cons are that the gears and bearing are highly reliable and the risk for failure is low. This concept also depends on a gear shifter, which will increase the weight and the risk of mechanical failure.

1.4 Synchronized and independent gear shift concept

The idea behind this concept is to be able to shift gears like a bicycle. A differential on each side of the gear in the middle makes the solar arrays able to move independently when the chain is on one of the outer gears. The solar arrays moves synchronized when driving the gear in the middle. The flexes will have to be placed between the wall and the outer gears in this configuration. The drawback with this concept is that it requires three gears, two differentials with three gears each, two separate twist capsules and a device to move the chain. Many parts means higher price, a lot of weight and reduced redundancy. This concept is not recommended if there is not required that the solar arrays to be able to move independently.

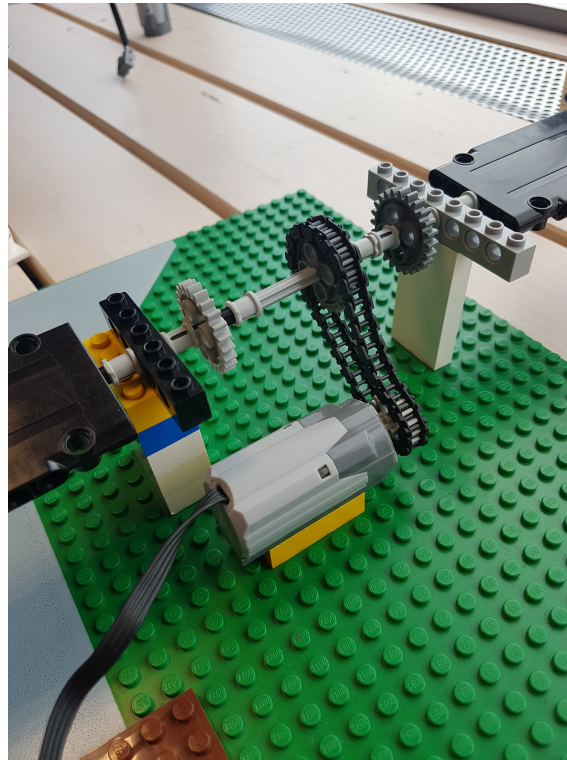


Figure 5: Synchronized and independent gear shift concept

1.5 Parallell motor and shaft

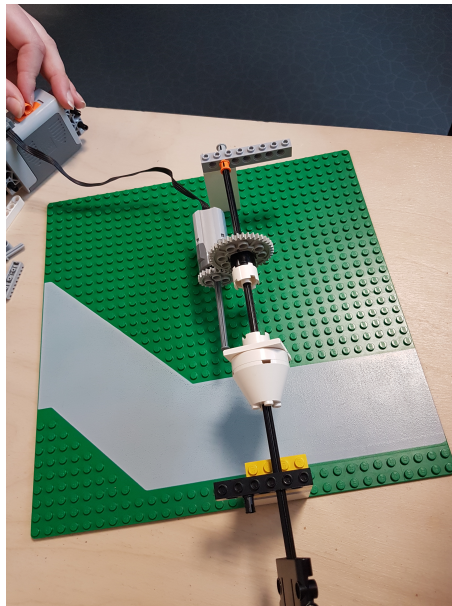


Figure 6: Parallel motor and shaft

This configuration is based on one shaft, one twist capsule and one motor to rotate the solar arrays synchronously. Having the motor parallel to the shaft allows the SADM to use a bigger motor. This might be necessarily as most stepper motors for this use are too long to be placed perpendicular to the shaft.

2 Conclusion

Our product is a solar array driving mechanism that is meant to be compatible with any other cubesats in a certain size, almost like a off the shelf-product. To make it competitive in the market, it has to be lightweight, reliable and affordable to produce. Based on these requirements, we have chosen the configuration Motor parallel to the shaft. We believe this is the configuration with the highest reliability and the lowest weight due to few components. Using Lego was a very simple way to prove the functionality of the concept, but it doesn't show if it's possible to fit all the components inside our limited space. A drawing was drawn in 1:1 ratio in addition to building Lego. This was to get an idea of how to utilize the space inside the SADM as the space of a 2U is only 200 x 100 x 100 mm. The dimensions of the components were only a rough estimation, and were slightly oversized to create a worst case scenario. The drawing became a great way of picturing where all the parts would fit in, because of the limitations on how large the parts could be. The prototype Mark2 will be built based on this drawing.

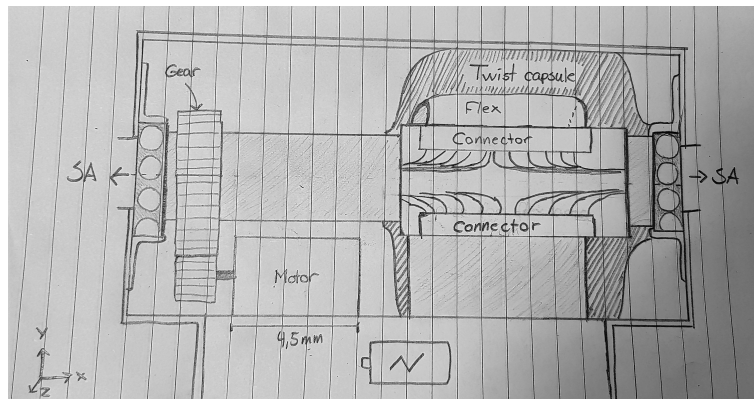


Figure 7: Drawing of the concept in 1:1 size



Bachelor of Engineering

Project appendix

Winter semester 2019

Design Description

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Erica Fegri, Håvar Østrem, Ming Kit Wong, Rita Hogstad
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document is describing the final design of the SADM.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 22, 2019	Signature
---------------------------------------	-----------

Contents

- 1 Introduction 3**
- 2 Materials 3**
- 3 The structure 4**
 - 3.1 Left and Right lids 6
 - 3.2 The Middle Wall 6
 - 3.3 The L-profiles 7
 - 3.4 The flat walls 8
- 4 Drivetrain & Twist Capsule 9**
 - 4.1 Flex Shaft 11
 - 4.2 Capsule Housing 12
 - 4.3 Flexible PCB 13
 - 4.4 Main Shafts 14
 - 4.5 Bearings 15
- 5 Motor 15**
 - 5.1 Gears 16
- 6 Electrical harness 19**
 - 6.1 Derating 19
 - 6.2 Connectors 21
 - 6.3 Ground 21
- 7 Flexible PCB 22**
 - 7.1 Scemactical Drawing 22
 - 7.2 Padstacks and footprints 22
 - 7.3 PCB Layout 23
 - 7.4 Dimensions 23
- 8 References 24**
- A 2D Drawings and Appendix 25**
 - A.1 Bearing Lid 2D 25
 - A.2 Capsule Housing 2D 27
 - A.3 Wall with Dsubs 2D 28
 - A.4 FlexShaft2D 29
 - A.5 Left Lid 2D 30
 - A.6 Long Main Shaft 2D 31
 - A.7 L Shaped Beam 2D 32
 - A.8 Midlle wall 2D 33
 - A.9 Right Lid 2D 34
 - A.10 Short Main Shaft 2D 35
 - A.11 Side Walls 2D 36

A.12 SADM powertransfer 37
A.13 Motor connection 38
A.14 HPC Gear table 39
A.15 Datasheet stepper motor 43

1 Introduction

Mark3 (M3) is the last prototype that was built and it was designed to fulfill all the requirements that is classified as "must". The purpose of M3 is to prove the functionality and performance that is required of the product and to have a prototype where the team can test the mechanical functionalities, electronics and the Diagnostic Systems (DS).

The main goal of the design was to have a product that is

- Easy to assemble
- Have a reasonable production cost
- Scalability

This document describes the design behind SatLight's solar array drive mechanism (SADM).

2 Materials

Space grade materials and components was not necessarily for testing at our level, therefore, material with similar properties in industry standard was chosen instead. M3 has a twin that is called M3-Space, which is a CAD-model used for simulations. The design of M3-Space is very similar to the original M3, but the materials defined in the CAD-model are space grade to give a realistic test results during analyzing. Some minor changes were made on M3-space after the prototype was produced. The figures in this document is showing the design of M3-Space.

The main material for M3 is aluminum 6061, in which the main alloying elements are magnesium and silicon. This gives the material very good mechanical properties and is very commonly used in extruding [1]. M3-Space is designed with alloy 7075, which is space grade and standard for space vehicle parts. The CubeSat Design Specification (CDS) is also specifying that the structure is to be made of 7075 [2].

The Flex was initially designed to be divided horizontally (along the length of the Flex) into five zones, of which the two outermost zones were meant to be manufactured with ordinary rigid PCB material, such as FR4, [] as well as four layers of copper and adhesive. The zone in the middle, which is the Flexible part, is made from polyimide, 70 μm of copper and coverlay. The zones in between the Flex and the rigid zones are called stiffeners and are simply expansions of the Flex with additional polyimide.

In the process of ordering the Flex, the team received price estimates for the initial design as well as a design somewhat modified with the rigid zones, which greatly increased the cost of production, replaced with stiffeners. The latter choice was preferable and thus, the Flex design have three zones instead of five. A Flexible zone and two stiffeners at each end. Apart from the stiffeners, the stackup remains the same.

3 The structure

This section describes the primary structure of the SADM. A primary structure is the components that is designed to transmit loads through the spacecraft, and to the interface of the deployer. It is also providing other primary functions, such as attachment points to other components and heat transmitting.

The structure consists of three walls that are 100x100mm, four L-profiles and four 100x200mm walls of sheet metal, assembled with screws. This design makes the SADM easy to scale up or down by adjusting the lengths of the L-profiles and the sheet metals. The main production methods are milling, cutting and bending.

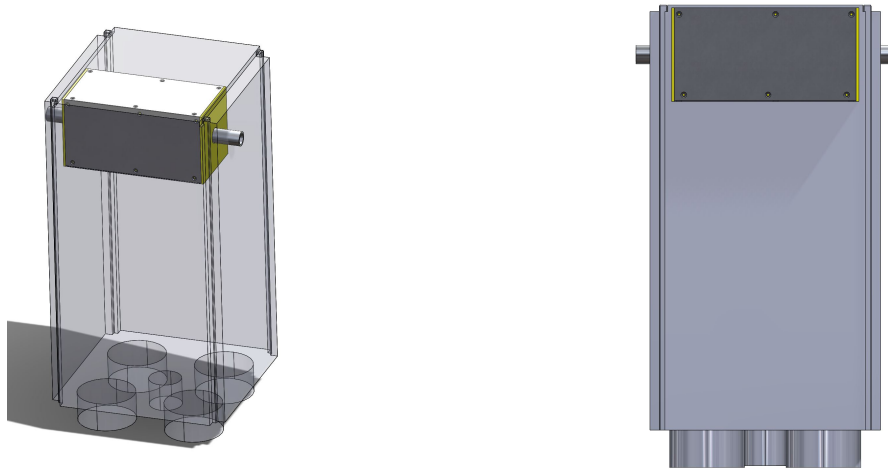
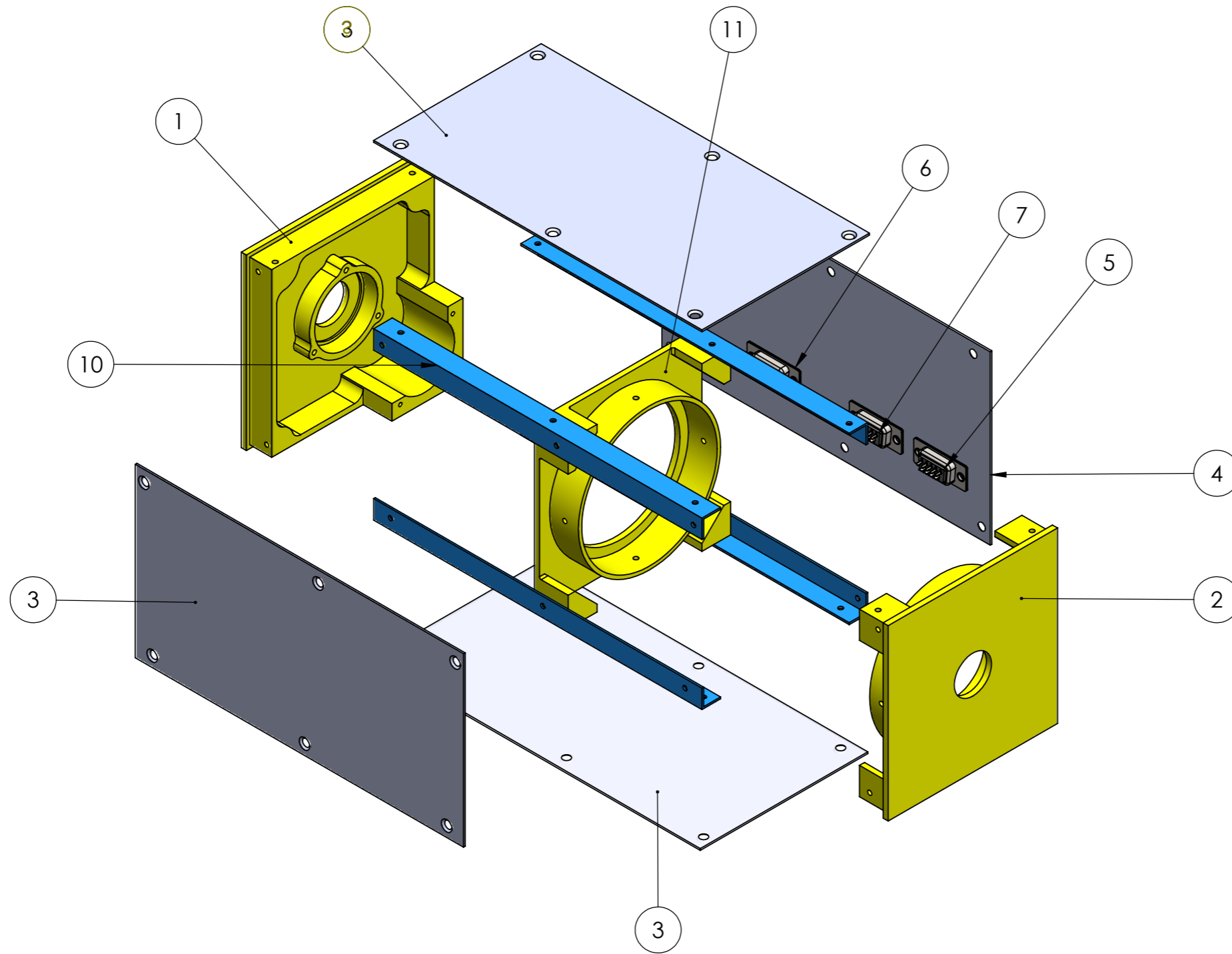


Figure 1: The SatLight SADM inside of a transparent 16U (left) and partly outside of a 16U to compare the sizes (right).

The outside dimension of the SatLight SADM is 100x100x200mm and the width of a 16U is 246mm. This makes it possible to fit our product inside of the structure of a 16U, so that the space craft (SC) can provide some protection to our product from the space environment. The interface to the SC is not specified, therefore it is not taken in consideration during design of the structure, but this should be easy to implement.



ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	Left Lid - M3	Left Lid	1
2	Right lid	Right Lid	1
3	Wall 200x100	Panel Wall	3
4	Wall 200x100 With Dsub holes	Panel Wall - Satellite Infrface, with power transfer	1
6	D-T	Dsub	3
10	L-profile	L Shaped Wall Supports	4
11	Wall B, middle 100x100	Middle Wall	1

3.1 Left and Right lids

Both the lids are 100x100mm and 3mm thick, made by milling. There are two threaded screw holes on each corners to attach the lid to the L-profiles and the flat walls. They both have a hole in the center where the shaft comes out.

The lid on the left-hand side in Figure 2 has 3mm walls along the edges to stiffen the lid, which is necessarily under production and considered as an advantage to strengthen the overall structure. The lower right corner of the Left Lid is where the motor is mounted. The extrude is designed to fit the motor(see section 5) with a hollowing for the motor gear. The lid on the right hand side in figure 2 does not have the supporting walls along the edge, because the circular flange in the middle will stiffen the structure.



Figure 2: The lids are a part of the primary structure, but also functioning as mounting for Twist Capsule, bearings and the motor.

The flange in the middle of both lids are for mounting the bearing to the lid, where it will be held in place by a bearing lid that is also functioning as a hard preload. There is also a small edge on the inside of the flange, that is contributing to preload the bearing. Both the flange and the edge inside is specifically designed for the bearing is chosen for M3, which is not space grade. Modification will be needed when space grade bearings are selected.

3.2 The Middle Wall

This wall is placed inside the unit and functions as a stiffener and is also used to attach the Twist Capsule to the structure. It is 96x96mm and 3mm thick with a hole in the middle to fit the shaft through the wall and to reduce weight. The Middle Wall is produced by milling.

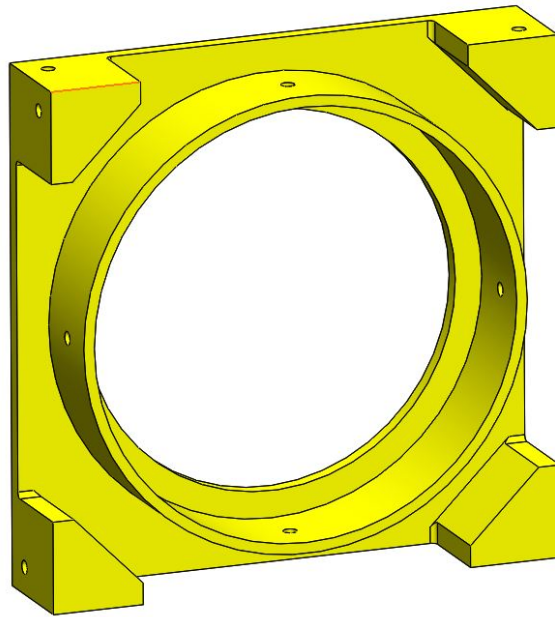


Figure 3: The Middle Wall that functions as a stiffener to the SADM and the Twist Capsule.

The Middle Wall has the same corners as the Right Lid, again the supporting walls along the edges are not included as the flange in the middle will function as stiffener. This will also allow the machinist to use a larger diameter on the tool to remove the mass around the flange when milling. This will shorten the production time and reduce cost.

3.3 The L-profiles

The SADM has four L-profiles to attach the lids and walls together. They are 1mm thick and 193mm long and have screw holes that accommodates with the other components.

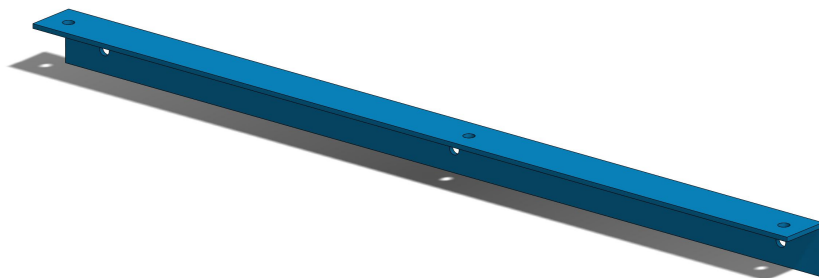


Figure 4: The Right Lid is functioning as a bearing and Twist Capsule holder.

For Mark3, the L-profiles are made of bent sheet metal as it is easily accessible and the shape is easy to produce. The drawback of a bent sheet is that the bending creates an unwanted radius on the inside of the corner and also unpredicted stresses on the material. For mass production, we recommend that the L-profiles are made by cold extrusion.

Extrusion is a very common way to manufacture long parts that have the same shape all the way, and it will also avoid the unwanted inner radius in the corner. Cold extrusion is more

expensive than hot extrusion, as it requires more energy to deform a cold work piece. Although, the finished product is much more precise as warping is avoided when there are no thermal stresses (provided that the heat generated by friction and deformation is below recrystallization temperature). The overall advantages of cold extrusion is that it gives a good control of dimensional tolerances, improved surface finish and reduces the need for finishing operations.[3] Yet, this production method is only recommended for mass production, as it requires extrusion tools. Bending might be preferable in few quantities.

3.4 The flat walls

The walls consist of four 1mm aluminum sheet metal that are 194x99mm. Their purpose is to cover the Drivetrain and Twist Capsule and to stiffen the structure. Since the SADM is so small that it can fit inside of a 16U, the walls do not need to protect the Drivetrain and Twist Capsule from the space environment.

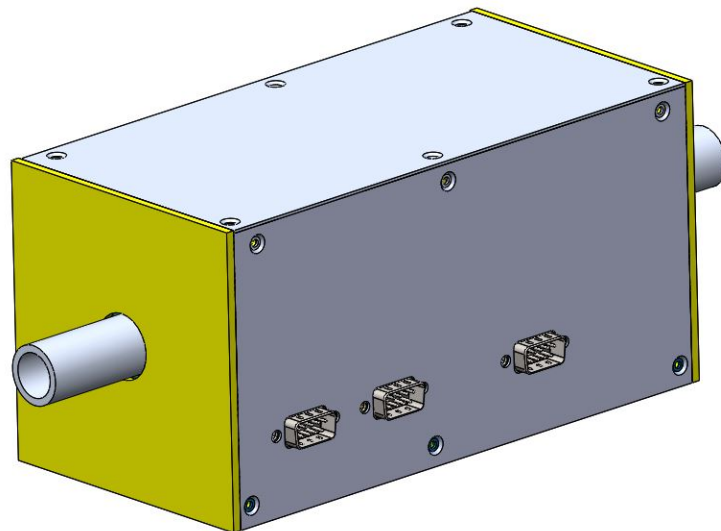
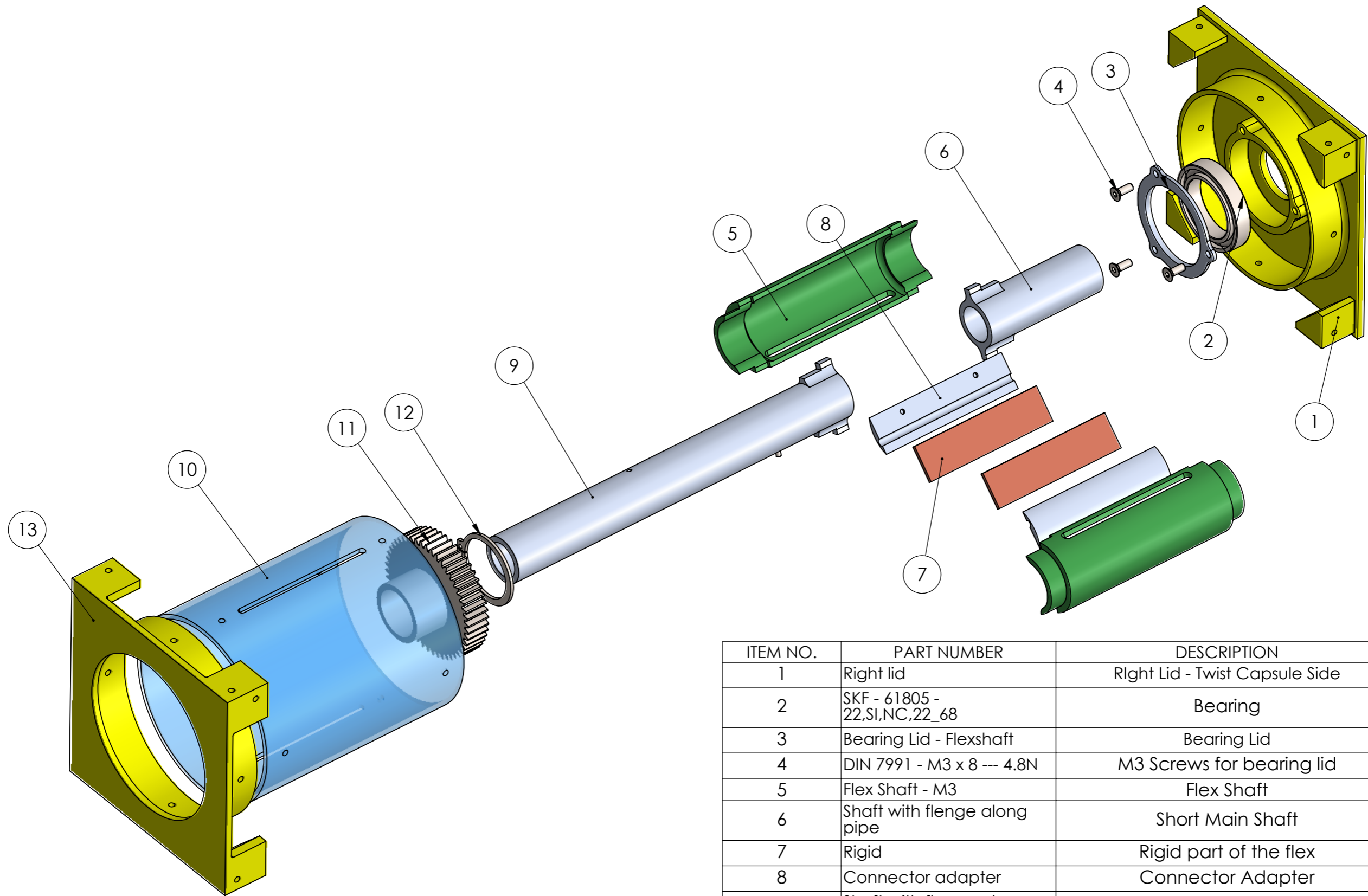


Figure 5: An isometric view of the SADM from behind.

The placement of screw holes are identical on all the walls, this makes the parts easy to produce and reduces the risk for improper installation during assembly. One of the walls is equipped with holes for D-sub connectors, two for the Flexes and one for the motor. The wall with D-sub connectors shown in Figure 5 is facing the inside of the satellite.

4 Drivetrain & Twist Capsule

The main components of the Drivetrain is one gear fastened to the motor, one gear fastened to the long Main Shaft. The shaft is also wider inside the Twist Capsule, to fit one end of each of the two Flexes. There are also secondary components and supporting elements, such as bearings, bearing lids and connector adapter, that will be described below. This section will also describe the Capsule Housing, which is one of the parts of the Twist Capsule, together with the Flex Shaft and the Flex itself. The Capsule Housing is the stationary part of the Twist Capsule and the Flex Shaft is the rotating part.



ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	Right lid	Right Lid - Twist Capsule Side	1
2	SKF - 61805 - 22,SI,NC,22_68	Bearing	1
3	Bearing Lid - Flexshaft	Bearing Lid	1
4	DIN 7991 - M3 x 8 --- 4.8N	M3 Screws for bearing lid	3
5	Flex Shaft - M3	Flex Shaft	2
6	Shaft with flenge along pipe	Short Main Shaft	1
7	Rigid	Rigid part of the flex	2
8	Connector adapter	Connector Adapter	2
9	Shaft with flenge along pipe, long	Long Main Shaft	1
10	Capsule Housing	Capsule Housing	1
11	DIN - Spur gear 1M 50T 20PA 10FW --- S50B25H25L20N	Large Gear	1
12	B27.7M - 3AM1-27	Pipe Clamp	1
13	Wall B, middle 100x100	Middle Wall	1

4.1 Flex Shaft

Item no. 5 in the assembly drawing is one half of the Flex Shaft, two pieces of this part makes the complete Flex Shaft. The Flex Shaft is connected to the Main Shafts (item no. 6 and 9) at the ends and held together with a bearing on one side and a aluminum ring on the other side. The Flex Shaft is the core of both the Drivetrain and the Twist Capsule and its main function is to house the interface between the Flex and the wiring coming from the solar arrays.

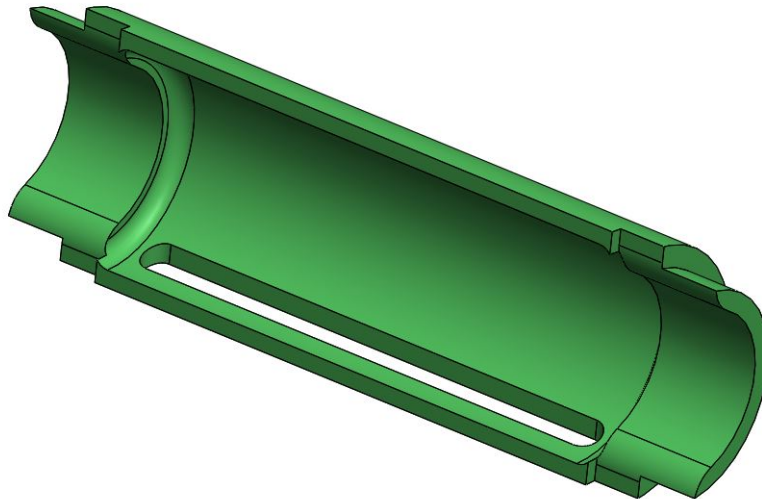


Figure 6: The Flex Shaft

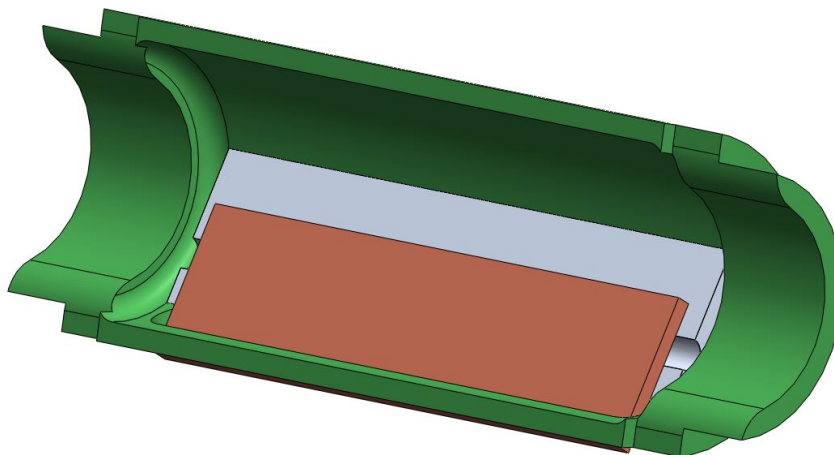


Figure 7: The Flex Shaft with the connector adapter(item No. 8) and an illustration of the rigid.

The width and diameters of the Flex Shaft is mainly determined by the following constraints and parameters:

- The lower bound of the Flex Shaft's inner diameter is constrained by the size of the harness, connectors and end of the Flex, which need to fit inside the Flex Shaft.
- The upper bound of the Flex Shaft's outer diameter is constrained by the relationship with the inner diameter of the Capsule Housing, as well as the length of the Flex.

- The required mechanical stiffness and strength determine the thickness of the Flex Shaft, which is the difference between the outer diameter and the inner diameter.
- The maximum width of the Flex Shaft is constrained by the room available inside the SADM, along the axis of the shaft. This room is shared by The Twist Capsule (including the Flex Shaft), the motor, gears, walls and bearings.
- The minimum width of the Flex Shaft is constrained by the minimum width of the Flex.

The Flex has a stiffener at each end, where one end is mechanically connected to the Flex Shaft and electrically soldered to the harness, that transfer power from the solar arrays. The stiffener is mounted on the Flex connector adapter(item no. 8) which gives the Flexes a shallow angle, ensuring that the Flexes wraps around the Flex Shaft as tight as possible as well as keeping the the bending radius as high as possible to minimize fatigue. These three parts are attached to each other with the use of Epoxy Adhesive EC-2216.

The Flex Shaft were produced by turning and milling by Mechanical Lab. at KDA and they are one of the most demanding parts to produce due to their size, shape and requirement to tolerances. Molding might be a faster and better production method for mass production, though it will still require machining afterwards to get the desired tolerances and surface finish.

4.2 Capsule Housing

Item no. 10. The Capsule Housing is the stationary part of the Twist Capsule and is mounted between the Right Lid and the Middle Wall. The main function of the housing is to protect and to limit the movement of the Flex. The inner diameter of the housing is therefore carefully calculated with regard to the length and the stiffness of the Flex. The Capsule Housing is equipped with two openings along the wall where the Flex exit the Twist Capsule. It was decided, with the use of Flexculator, that a inner diameter of 76mm was the most optimal for our Flex.

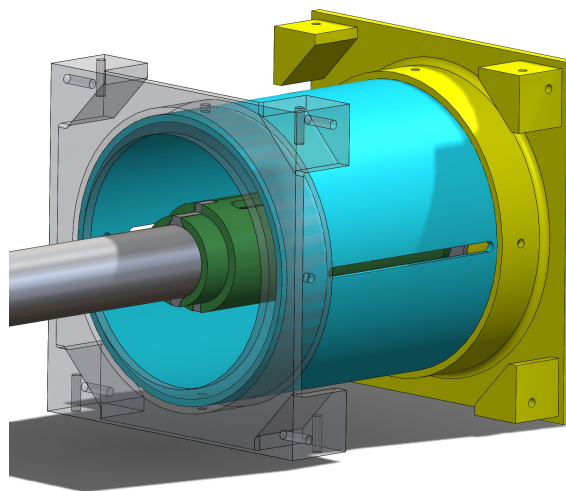


Figure 8: Illustration of how the Capsule Housing is mounted to cover the Flex and Flex Shaft.

For Mark3, the Capsule Housing was made with transparent plastic sheets to demonstrate the movement of the Flex. For operational use, we assume that this part can be made of some-

thing light weight and easy to produce, such as PEEK-plastic instead of metal, because the housing is not taking any loads. Thermal expansion must be considered if any other materials than aluminium is used.

The width and diameters of the Capsule Housing is mainly determined by the following constraints and parameters:

- The lower bound of the Capsule Housing's inner diameter is constrained by the relationship between the outer diameter of the Flex Shaft, as well as the length of the Flex.
- The upper bound of the Flex Shaft's outer diameter is constrained size of the inner dimensions of the SADM, the Flexes' connections, the leads from the Capsule Housing to the D-sub connectors and the D-sub connectors themselves.
- The required mechanical stiffness and strength determine the thickness of the Capsule Housing, which is the difference between the outer diameter and the inner diameter.
- The maximum width of the Flex Shaft is constrained by the room available inside the SADM, along the axis of the shaft. This room is shared by The Twist Capsule (including the Capsule Housing), the motor, gears, walls and bearings.
- The minimum width of the Capsule Housing is constrained by the minimum width of the Flex.

4.3 Flexible PCB

The width of the two Flexes are 60 mm respectively to provide sufficient conductive copper. The Flexes runs through slots in both the Flex Shaft and the Capsule Housing. They are mechanically connected on the inside of the Flex Shaft and on the outside of the Capsule Housing. Inside the Flex Shaft, the Flexes are soldered to the wires of the solar array harness. On the outside of the Capsule Housing they are soldered to the leads, which at the other end are connected to the D-sub. The length of the Flexes must be adequate for a full revolution of the shaft. Furthermore they should always have a positive curvature. This requires an accurate calculation of the relationship between the Flexes, the Flex Shaft and the Capsule Housing. The length of the Flexes are the the most constrictive of these. The relationship is shown in the following equation.

$$l_f = \pi \left(\frac{r_c + r_s}{2} \right) + r_c (\theta_c - \theta_{co}) + r_s (\theta_s - (\theta_c + \pi)) \quad (1)$$

Equation 1: Flex length in terms of radius-angle products

In equation 1, the minimum length of one Flex l_f , are determined by the radius of the Flex Shaft r_s , the radius of the Capsule Housing r_c , the angle of the slot in the Flex Shaft θ_s and the angle of the slot in the Capsule Housing θ_{co} . The angle θ_s must be positive and rotations of the Flex must be added to this angle. θ_c is the angle for were the Flex touch the Twist Capsule. See figure 9.

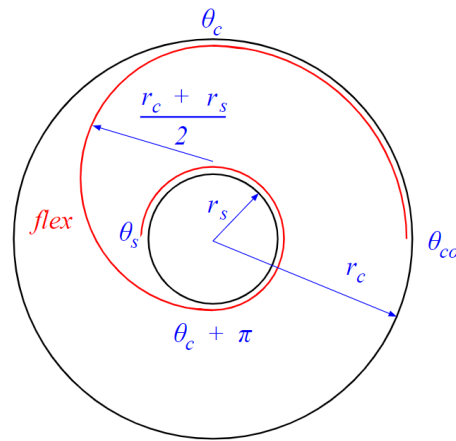
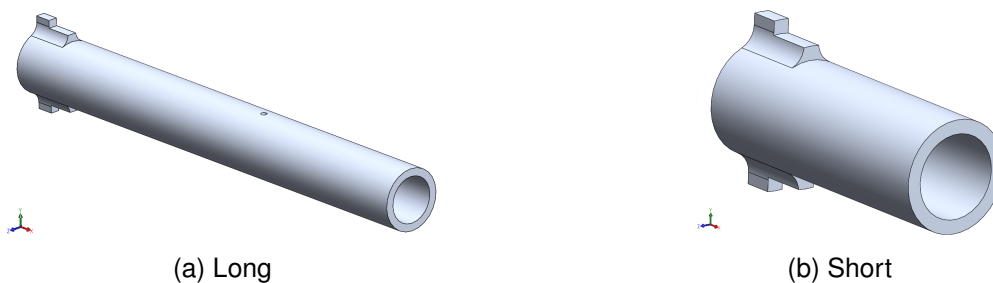


Figure 9: Relationship between minimum Flex length, shaft radius and Capsule Housing radius

4.4 Main Shafts

Item no. 6 and 9 are the Main Shafts of the SADM. Since the Twist Capsule is in one half of the SADM and the motor and gear components are in the other side, there was need for two lengths for the Main Shafts. The flanges at the end are the interface with the Flex Shaft. It is hollow to house the harness between the solar arrays and the Flex Shaft. The inner diameter of the shaft is 15mm and based on a brief estimation of the cross section of the harness needed to transfer the necessarily power from SA to the space craft. The estimation was performed in the early phase of the project. The wall thickness of the shaft is 2.5mm, which makes the outer diameter 20mm.



(a) Long

(b) Short

Figure 10: Main Shafts

Later testing revealed that the shaft is oversized for the harness, but in order to simplify the production for Mech. lab, the shaft was made with 20mm in outer diameter and 10mm in inner diameter.

The long Main Shaft was assembled to the Flex Shaft by a aluminium ring and the short Main Shaft was assembled by a bearing (item no. 2).

4.5 Bearings

Item no. 2. There are two identical bearings in the SADM. Both of the bearings are SKF 61805-2RS1, deep groove ball bearings with 25mm and 37mm diameters. They are designed to take axial and radial loads and have a rubber seal with steel enforcement. The steel used for SKF super-precision bearings are made of carbon chromium steel (100Cr6) [1]. Since the SKF-bearings are not space grade, the material used for the ball bearings on the M3-space was therefor 400C. 400C has the highest strength, hardness and wear resistance of all the stainless alloys after heat treatment and it is also used by ADR ALCEN [4].

Attribute	Value
Inside Diameter	25mm
Outside Diameter	37mm
Ball Bearing Type	Deep Groove
Race Width	7mm
End Type	Sealed
Number of Rows	1
Static Load Rating	2.6kN
Cage Material	Steel
Race Type	Plain
Dynamic Load Rating	4.36kN
Bore Type	Parallel

Figure 11: Specification of bearing 61805-2RS1, provided by SKF.

5 Motor

The motor's function is to transform electrical power and -signals to rotation and position of the shaft and solar arrays. The motor

The motor is a stepper motor from OEM Motor and was selected by analyzing related requirements and by guidance from OEM Motor's sales office.

The key parameters for selecting motor:

- **Functional:** The motor's operating sequence is rotating slow to a given angular position. The motor must be strong enough to rotate the shaft, solar arrays and additional equipment.
- **Physical:** The motor must be as small and compact as possible to not seize physical space and weight.
- The motor interface shall be according to NEMA-standard, so it easily can be replaced.

Stepper motor performance:

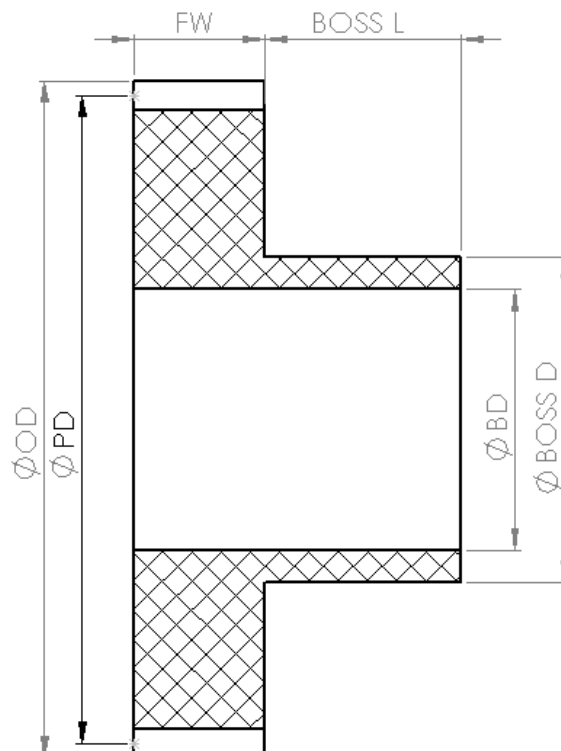
Manufacturer	Sanyo Deki
Model	SS2421 - 5041
Phases	2 phases
Full step angle	1.8 degrees
Winding resistance	3.5 Ω /phase
Winding inductance	1.2 mH/phase
Holding torque at rated current	0.083 Nm
Rated phase current	1A

Table 1: Motor characteristics.

The motor's datasheet is included in appendix A.15 and shows the motors dimensions and characteristics diagram.

5.1 Gears

Two spur gears was used to transfer movement from the motor to the drivetrain. The gear factor is 2 and the dimensions of the gears are listed in the figure below. The gear is item No. 11 in the Bill of material.



	No. of Teeth	PD (mm)	OD (mm)	BD (mm)	Boss D (mm)	Boss L (mm)	FW (mm)
Small Gear	25	25	27	5	0	0	10
Large Gear	50	50	52	20	25	20	10

Figure 12: Table of gears

The chosen vendor for the gears is HPC Gears Ltd with 40 years in making gear systems [5], and below there are two tables for the gears that were chosen and their data. Both of the tables shows torque check for module 1 gears. Figure 13 shows standard duty (steel, alternatively brass or tufnol) gears.

TORQUE CHECK 50 teeth x 1000 rpm	214M15 303S21	214M15 Cse/HD	Brass	Tufnol
variations TORQUE Nm.	2.600	12.000	1.250	1.000
at back K.W.	0.254	1.261	0.134	0.104

Figure 13: Standard duty steel gears

Figure 14 shows heavy duty gears for higher torque performance.

TORQUE CHECK 50 teeth x 1000 rpm	655M13	655M13 Cse/HD	817M40	817M40 Ind/HD
variations TORQUE Nm.	10.600	29.300	10.600	15.520
at back K.W.	1.119	3.074	1.119	1.626

Figure 14: Heavy duty

The standard duty gears' part number is G1-25 for the small gear and G1-50 for the large gear. As for heavy duty gears, the part numbers are YG1-25 for the small gear, and YG1-50 for the large gear. More information about the gears in appendix A.14

The dimensioning of the gears was decided due to the space limitations of the SADM and to make genuine dimensions so that there would be an "off the shelf" product. In Figure 15 there is an illustration of how the geometry of the gears versus Main Shaft and motor holder was designed. To relieve the motor the largest possible gears were used.

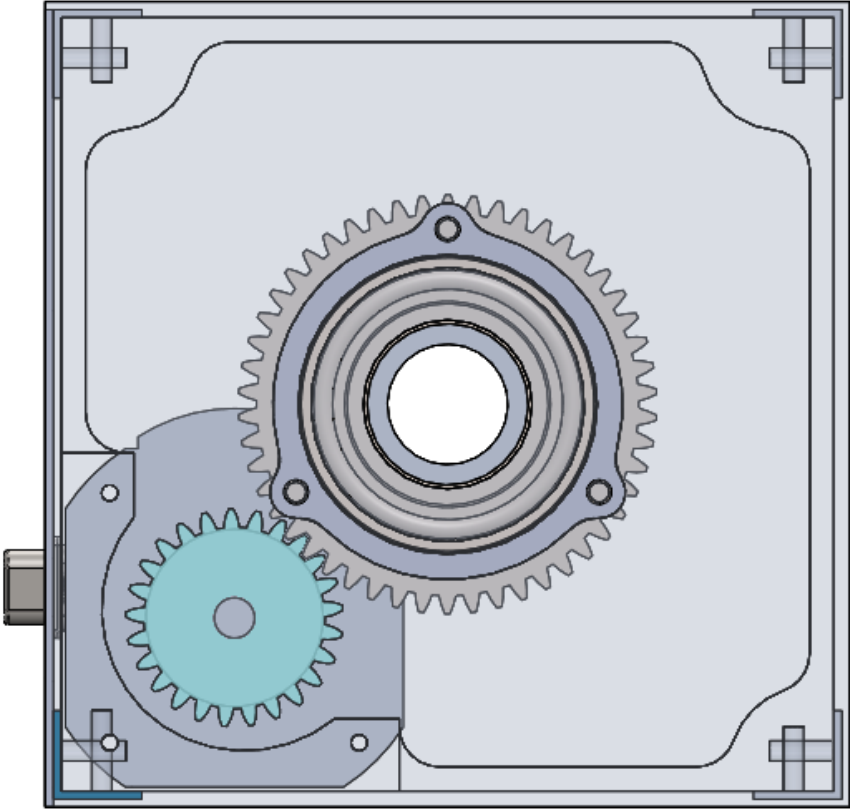


Figure 15: Left view showing space limitations

6 Electrical harness

6.1 Derating

The drive parameters for derating of power transfers are the following (table 2).

Req.-ID	Requirement
R4-05	The maximum total current transfer in the power lines shall be 20A forward and 20A return; - 10A each direction each solar array
R7-03	The SADM power transfers shall meet with the derating performance in ECSS-Q-ST-30-11C [SD8] for nominal operation.
R7-04	The SADM power transfers shall meet the derating performance in ECSS-Q-ST-30-11C [SD8] in failure case (loss of one single wire).[...]
R4-07	Solar Array grounding lines shall be insulated from SADM grounding.
R4-03	The length of the rotor harness shall be 500 mm flying leads.

Table 2: Requirements connected to derating.

The power transfers three different stages is shown in table 3. According to ECSS-Q-ST-30-11C [6], table 6-37: *Derating of parameters for Wires and cables family-group code 13-01 to 13-03*, derated current for single wire (I_{SW}), shown in column three in table 3:

Type of power transfer	Cross section	I_{SW}
Leads from D-sub to interface on Capsule Housing	$2 \cdot 2 \cdot 4 \cdot 0.35\text{mm}^2 (> \text{AWG } 22)$	5A
Flex traces	$2 \cdot 2 \cdot 4 \cdot 0.35\text{mm}^2 (> \text{AWG } 22)$	5A
Flying leads from Flex Shaft trough shaft	$2 \cdot 2 \cdot 4 \cdot 0.5\text{mm}^2 (\text{AWG } 20)$	7.5A

Table 3: Power transfers cross section.

Note: Values for I_{SW} are for ambient temperature of 40°C. Temperature factor is not taken into account.

When the power transfers are loaded to the maximum (R4-05), the current transferred in each lead is:

$$\frac{20A + 20A}{2 * 2 * 4} = 2.50A/lead \quad (2)$$

The maximum amount of current in each lead is 2.5 A. To achieve this, the power transfer is equally disturbed in the leads, the resistance in each parallel path has to be equal.

Wire in bundles:

The wires occurs in bundles inside the shaft, (see table 3). Inside the Flex Shaft, the wires enters in opposite parallel and are continued inside the shaft in separate directions to solar arrays, one each side of the SADM. Wires in bundles decreases the current capacity in each wire. The bundle inside the shaft is $2 * 2 * 4 = 16$ wires. The Flex inside Twist Capsule is not

considered as a bundle due to its geometry and physical characteristics.

Section 6.32.2, bullet point b, in ECSS-Q-ST-30-11C gives us the relation between derating on current for bundles (I_{BW}) and single wires (I_{SW}):

$$I_{BW} = I_{SW} * K \quad (3)$$

The bundle inside Flex Shaft has 16 wires transferring current, and two grounding lines, one for each solar array that according to R4-07 shall be insulated from SADM grounding lines.

Section 6.32.2, bullet point c explains how to calculate number of wires (N) in the same bundle including leads with current (N_{WC}) and leads without current (N_{GND}):

$$N = N_{WC} + 0.5 * N_{GND}$$

$$N = 16 + 0.5 * 2 \quad (4)$$

According to table 6-38: *Bundle calculator for calculation of the derated current for each individual wire in bundles of N wires*:

For N = 17 from equation 4, the calculation of factor K is:

$$K = 0.81 - [0.15 * \ln(N)]$$

$$K = 0.81 - [0.15 * \ln(17)]$$

$$K = 0.39 \quad (5)$$

Using factor K in equation (3) gives the deration on current for each lead in bundle;

$$I_{BW} = 7.5A * 0.39$$

$$I_{BW} = 2.89A/lead \quad (6)$$

Based on calculations in equations (2) (4), (5) and (6) requirement R4-05 and R7-03 is verified: The current in each lead when the load is maximized is less than the derated current in each lead. $2.50A < 2.89A$

The definition of failure cause is according to requirement R7-04 in requirement specifications as follows:

Definition of failure case: If only one wire is lost, the increase in current applies only for one wire in the bundle and the margin is considered acceptable as long as the current is below single wired de-rated capacity.

Consider maximum load on power transfers, the increase in current for one wire due to failure cause is:

$$2 * 2.5A = 5A \quad (7)$$

$5A \leq I_{SW}$ for both $0.35mm^2$ /AWG 22 and $0.50mm^2$ /AWG 20.

6.2 Connectors

The requirement taken into consideration when choosing the electrical interfaces is:

Req.-ID	Requirement
R4-04	There shall be integrated connectors on the stator side, connector size and type are to be determined by SADM supplier.

Table 4

The connectors consist of one 9-pin male D-sub connector for motor signals and two 15-pin female D-sub connectors.

ECSS-Q-ST-30-11C [6] **Ch. 6.11 Connectors [...], section 6.11.3 Additional requirements not related to derating, point a:** *For power connectors, power and return lines shall be separated by at least one unassigned contact to reduce the short-circuit risk.*

The configuration on the D-sub connectors with 15 pin for power transfers are designed to fulfill this requirement. This configuration is shown in appendix A.12.

ECSS-Q-ST-30-11C [6] **Ch. 6.11 Connectors [...], section 6.11.3 Additional requirements not related to derating, point b and point c** is also taken into account.

The 9-pin motor connector configuration is shown in appendix A.13.

6.3 Ground

Req.-ID	Requirement
R2-05	The SADM shall transfer solar array grounding lines.
R4-06	Electrical grounding between SADM shaft and housing shall be redundant.
R4-07	Solar Array grounding lines shall be insulated from SADM grounding.

Table 5: Requirements connected to derating.

See appendix A.12 for grounding configuration. The power transferring system supports R4-07, SADM grounding lines are separated from SA grounding lines.

- SA grounding lines follows the power transfers.

- SA grounding lines inside shaft is $2 * 0.35mm^2 / AWG 22$.
- SA grounding lines is transferred trough the Flexes with $2 * 2.5mm$ track width * $70 \mu m$ height. Equals $0.175mm^2 / AWG 25$.
- SA grounding lines between Flex housing ad connection is $2 * 0.35mm^2 / AWG 22$.
- Redundant grounding should be considered achieved by wiring. This is not achieved in the prototype for sake of other mechanical design parameters.
- The Flexes has $2 * 0.175mm^2 / AWG$ (one track on each Flex) reserved redundant ground-ing. The grounding tracks is not connected to the mounting holes.

7 Flexible PCB

The schematics for the Flexible PCB is produced in Orcad, Cadence CIS. The Connectors are produced in Alegro Padstack Editor. The PCB Design is produced in Alegro PCB Editor. All of these software tools are from the Cadence 17.2-2016 student software package. [7]

7.1 Scemtical Drawing

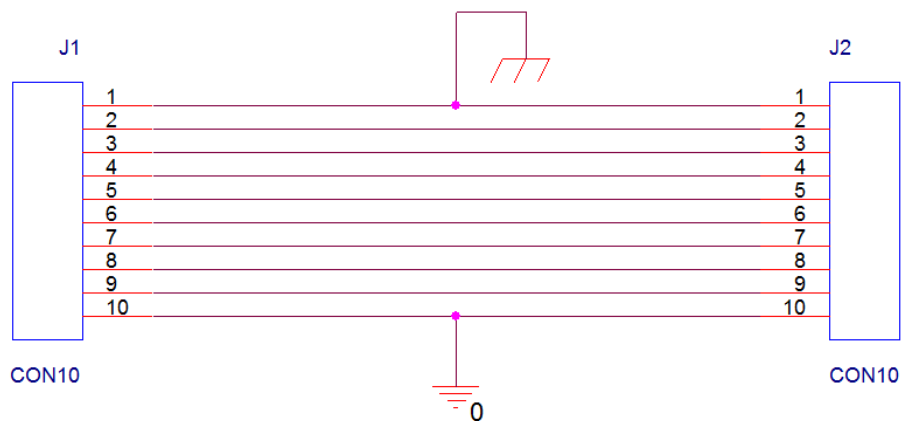


Figure 16: Flexible PCB, second iteration

Figure 16 show the schematic of the flex. It consists of two ten-pin connectors connected together with parallel wires. The schematic show two different grounds to distinguish solar array ground and protective earth in the PCB design. The schematic is mainly for creating a netlist for the PCB editor.

7.2 Padstacks and footprints

The padstacks were designed in Alegro padstack editor and the footprints were designed in Alegro PCB editor. These designs were modified by the PCB producer to simplify the production. The dimensions are shown in table 6 below.

7.3 PCB Layout

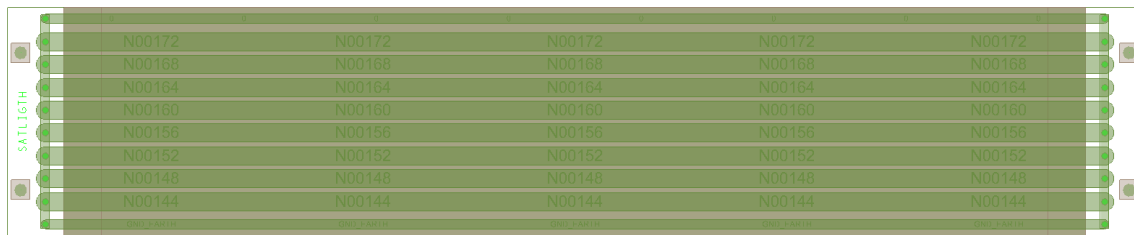


Figure 17: Flexible PCB, second iteration

The PCB layout was designed in Allegro PCB editor. Figure 17 shows the team's design, which was submitted to the producer. The producer modified the design to simplify production. However, the dimensions of the outline for the rigid zones, the flexible zones and the PCB as a whole remained as designed by the team. As did all of the copper thickness, the traces and spaces.

7.4 Dimensions

ZONES: stiffener with connectors, Flex, stiffener with connectors		
COPPER THICKNESS:	base copper + copper plating	2 · 35 μm
TOTAL LENGTH:		300 mm
STIFFENER LENGTH:	2 zones	25 mm
Flex LENGTH:	1 zone	250 mm
TOTAL WIDTH:		60 mm
STIFFENER WIDTH:		60 mm
Flex WIDTH :		60 mm
POWER TRACE WIDTH:	8 traces	5 mm
POWER TRACE SPACING:	7 spaces	1 mm
GROUND TRACE WIDTH:	2 traces	2.5 mm
GROUND TO Flex EDGE SPACING:	2 spaces	1.75 mm
GROUND TO POWER SPACING:	2 spaces	2.25 mm
CONNECTIONS: 2 connectors, one on each side of the Flex, each connector consists of 10 holes in a row with 6 mm spacing. Distance from drillhole center to the short edges of the PCB is 10 mm. Distance from drillhole center to the stiffener is 5 mm.		
DRILL HOLE DIAMETER:		1.3 mm
REGULAR PAD DIAMETER:		1.75 mm
THERMAL PAD DIAMETER:		2.15 mm
ANTIPAD DIAMETER:		2.15 mm

Table 6: Physical dimensions of the Flexible PCB

Table 6 shows the dimensions of the Flexible PCB, as designed by the team. The producer's modified dimensions are not included.

8 References

- [1] Wikipedia, "Small spacecraft technology state of the art," https://en.wikipedia.org/wiki/6061_aluminium_alloy [Accessed: May 09, 2019].
- [2] California Polytechnic State University, "Cubesat design specification, rev 13," https://static1.squarespace.com/static/5418c831e4b0fa4ecac1bacd/t/56e9b62337013b6c063a655a/1458157095454/cds_rev13_final2.pdf [Accessed: April 23, 2019].
- [3] S. K. S. Schmid, "Manufacturing engineering and technology, pearson prentice hall, 2006, p. 410."
- [4] A. ALCEN, "Catalogue - high precision ball bearings," http://www.adr.fr/sites/adr.fr/files/pdf/2.-_adr_cat.-_en.pdf [Accessed: May 12, 2019].
- [5] T. Hinchliffe and M. D. at HPC Gears, "About us," https://www.hpcgears.com/n/left_menu/about_us/about_us.php[Accessed: 15.05.2019].
- [6] E. C. for Space Standardization, "Ecss-q-30-11c rev1 derating - eee components," <https://ecss.nl/standard/ecss-q-st-30-11c-rev-1-derating-eee-components-4-october-2011/>[Accessed: 15.05.2019].
- [7] OrCAD Systems Corporation, "PCB Editor," 2016, <https://www.nordcad.dk/dk/student-forum/download-orcad.htm> [Accessed: 24.08.2018].
- [8] SKF, "Materials for bearing rings and rolling elements," <https://www.skf.com/group/products/bearings-units-housings/super-precision-bearings/principles/bearing-specifics/materials/materials-for-bearing-rings-and-rolling-elements/index.html> [Accessed: May 12, 2019].

A 2D Drawings and Appendix



Bachelor of Engineering

Project appendix

Winter semester 2019

Mark3 User Manual

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Rita Hogstad, Ming Kit Wong
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

Thorough description of how to assemble Mark3

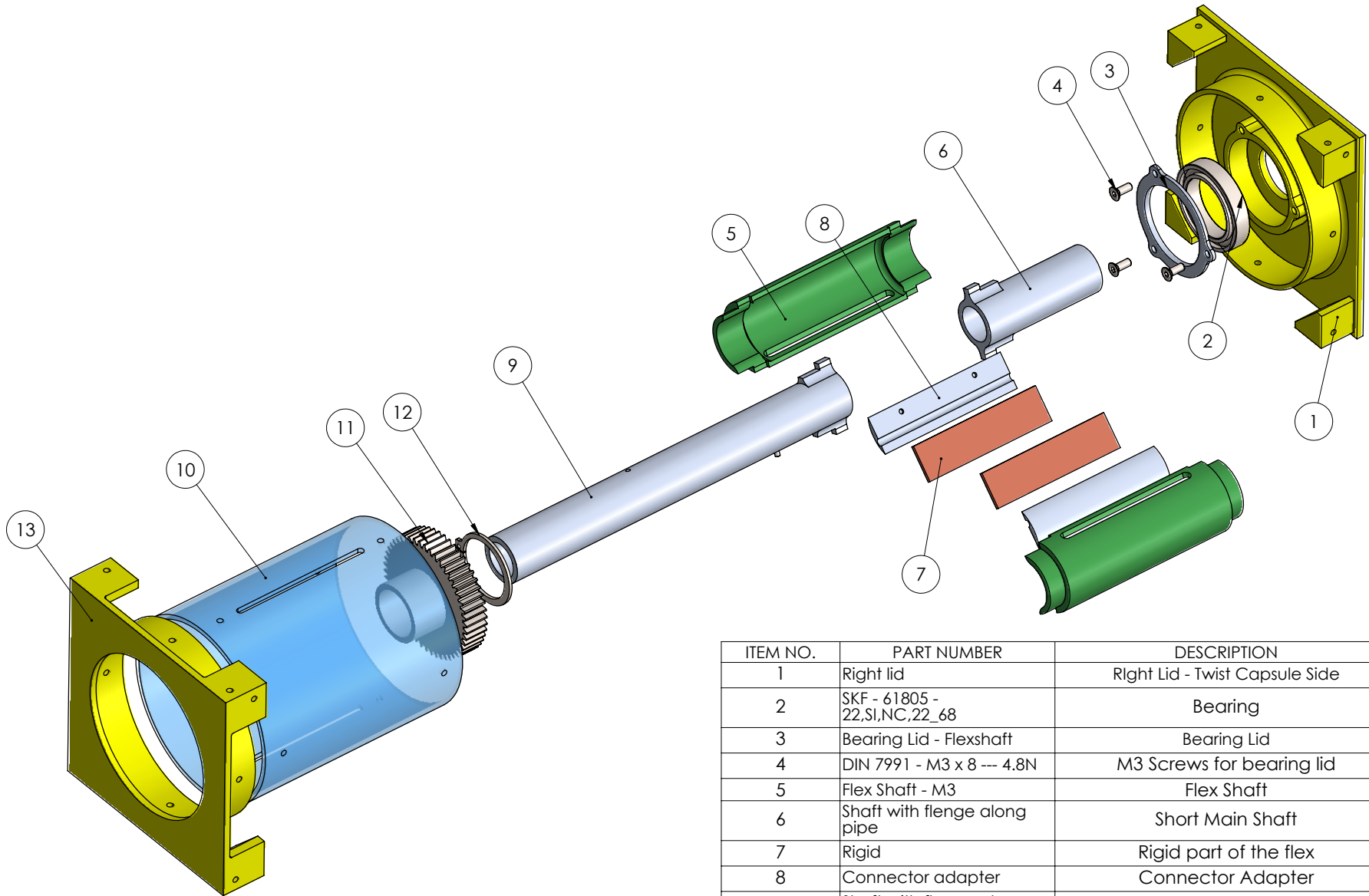
I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

Contents

- 1 Drivetrain and Twist Capsule** **3**
- 1.1 Attaching the harness 3
- 1.2 Assembling the Drivetrain 3

- 2 The Remaining Structure** **8**



ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	Right lid	Right Lid - Twist Capsule Side	1
2	SKF - 61805 - 22,SI,NC,22_68	Bearing	1
3	Bearing Lid - Flexshaft	Bearing Lid	1
4	DIN 7991 - M3 x 8 --- 4.8N	M3 Screws for bearing lid	3
5	Flex Shaft - M3	Flex Shaft	2
6	Shaft with flenge along pipe	Short Main Shaft	1
7	Rigid	Rigid part of the flex	2
8	Connector adapter	Connector Adapter	2
9	Shaft with flenge along pipe, long	Long Main Shaft	1
10	Capsule Housing	Capsule Housing	1
11	DIN - Spur gear 1M 50T 20PA 10FW --- S50B25H25L20N	Large Gear	1
12	B27.7M - 3AM1-27	Pipe Clamp	1
13	Wall B, middle 100x100	Middle Wall	1

1 Drivetrain and Twist Capsule

1.1 Attaching the harness

1. Solder the electrical harness from the solar array to the rigid part of the flex (item No. 7).
2. Stick the flex out of the slot in the flex shaft.
3. Run the wiring from the solar arrays through the main shafts(item No. 9 and 6).
4. Use Epoxy adhesive EC-2216 to bond the rigid part to the connector adapter (item No. 8) with the flex shaft(item No. 5) as shown in figure 1.

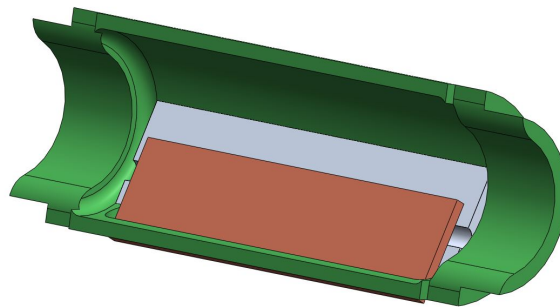


Figure 1: Illustration of flex shaft with adapter and flex. The harness are not showing.

5. Repeat the previous steps to assemble the other side of the flex shaft.
6. Allow the adhesive to cure. Follow the instructions about curing of EC-2216.

1.2 Assembling the Drivetrain

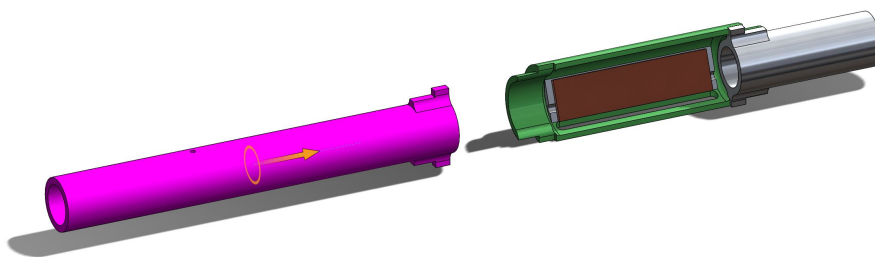


Figure 2: Place the main shafts (item 6 and 9) on each side of the flex shaft sub assembly, with the flange towards the flex.

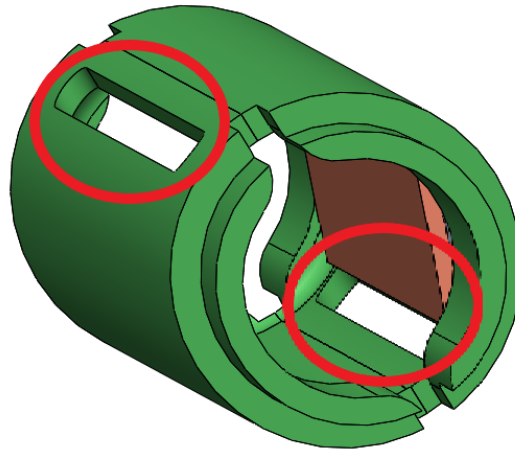


Figure 3: Place the flex shafts together as shown in figure. It is important that the slot in the flex shafts are not placed side by side, but facing opposite of each other.

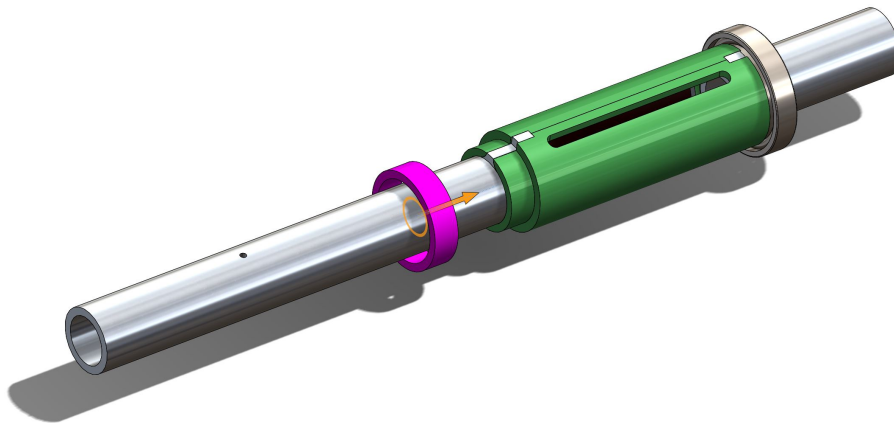


Figure 4: The retaining ring (item No. 12) and one of the bearings (item No. 2) is then mounted on the flex shaft assembly. Mount the retaining ring on the same side as the long main shaft and the bearing on the other side. Apply isopropanol on the gliding surface of the shaft to reduce friction. Preheat the rings to facilitate the assembly if necessary.



Figure 5: Place the shaft assembly into the Right Lid (item No.1) with the bearing facing the lid.

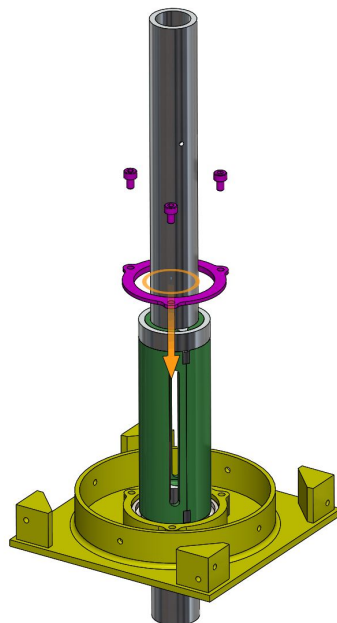


Figure 6: Use three socket head cap screws (item No. 4) to fasten the Bearing Lid (Item No. 3) to the Right Lid.

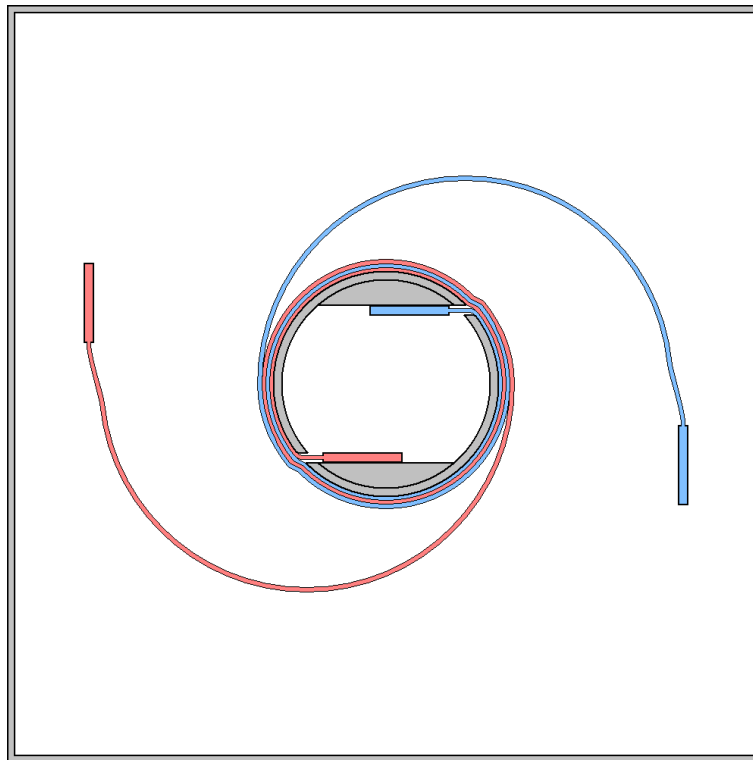


Figure 7: The flexes needs to be coiled around the flex shaft two times and laid on top of each other like shown in Figure 7. Turn the shaft counter clock wise.

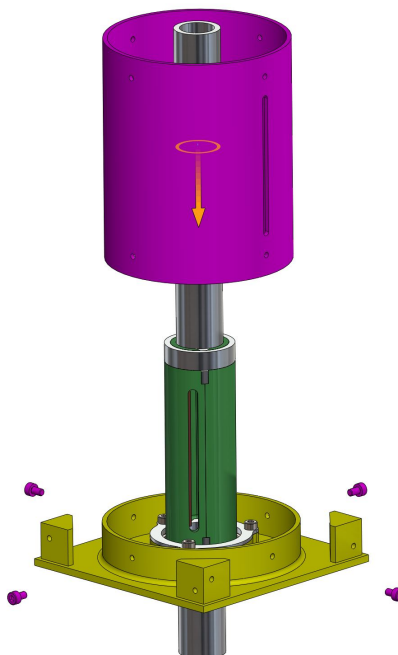


Figure 8: Use four socket head cap screws to attach the Capsule Housing (Item No. 10) to the Right Lid.

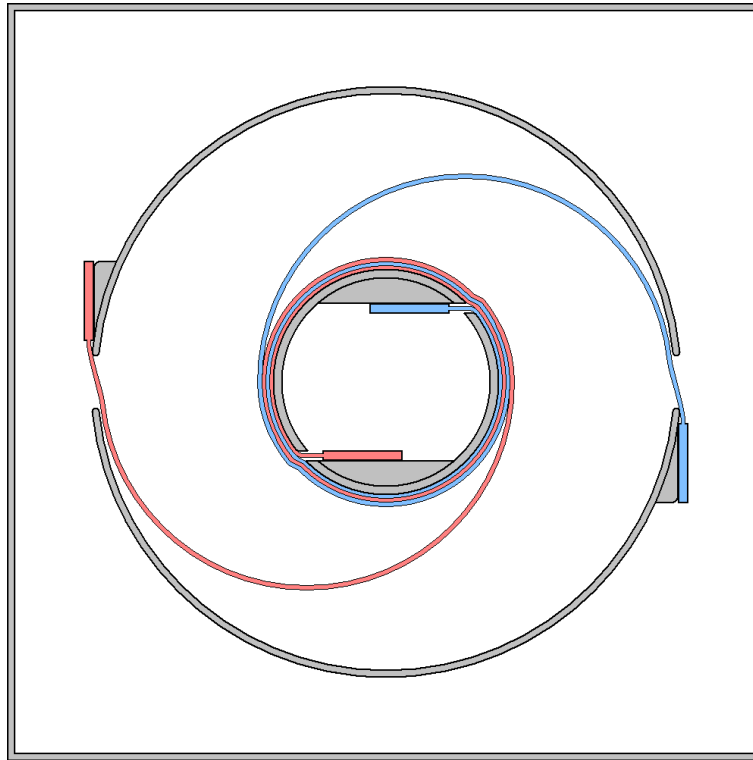


Figure 9: Now that the Capsule housing is inserted, the loose rigid parts of the flex needs to be inserted through the slots in the Capsule Housing and mounted to the Capsule Housing wall as shown in Figure 9

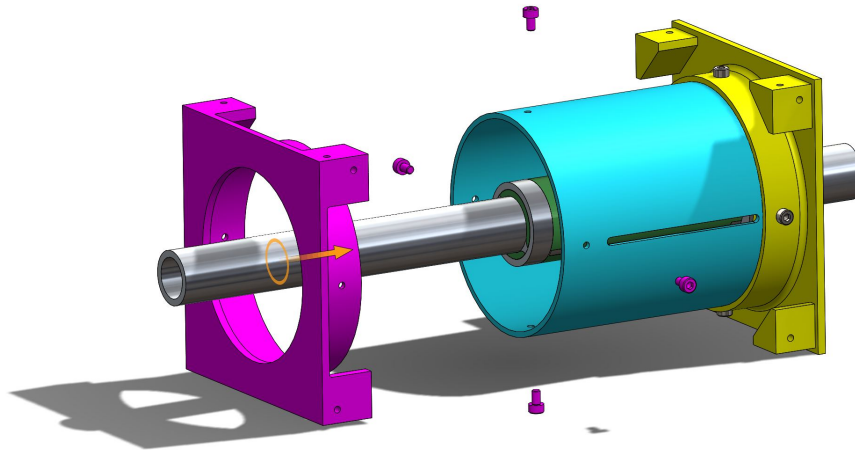


Figure 10: Close the twist capsule by fasten the middle wall (item No.13) onto the capsule housing with four socket head cap screws.

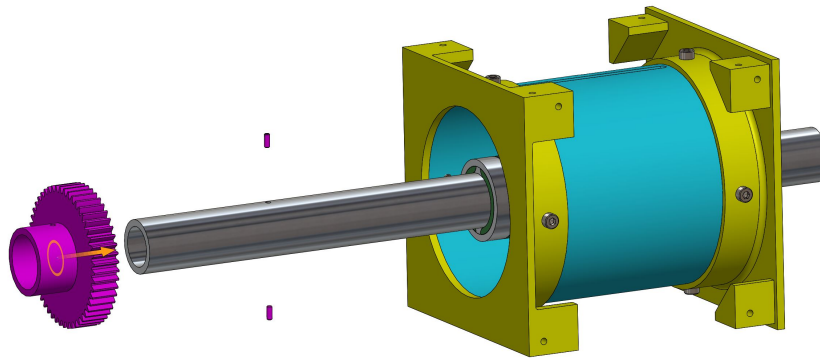


Figure 11: Attach the largest gear to the long main shaft with two set screws.

The sub assembly of drivetrain and twist capsule is done.

2 The Remaining Structure

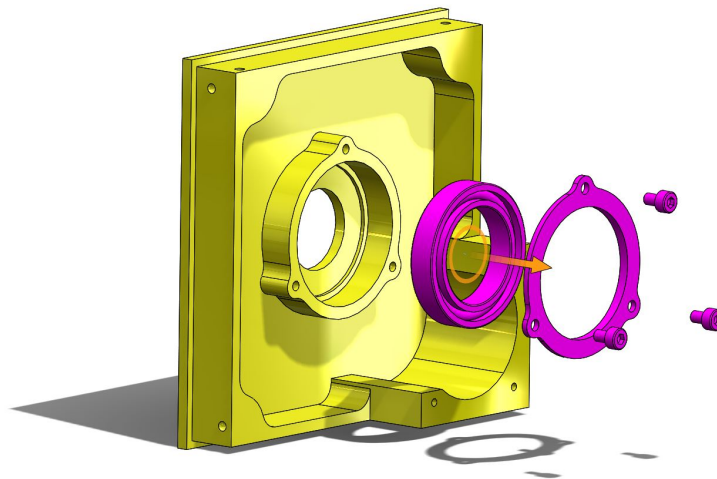


Figure 12: Place the Bearing (item No. 15) into the Left Lid (item No. 14) and fasten the Bearing Lid (item No. 16) onto the Left lid with three socket head cap screws (item No. 18)

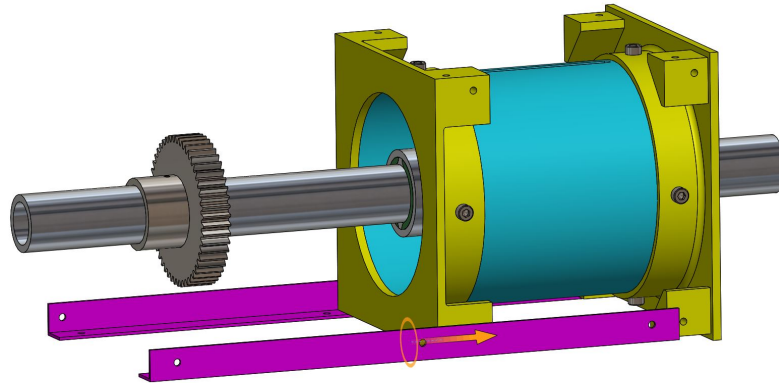


Figure 13: Place two of the L-profiles beneath the drivetrain sub assembly without fastening it. Ensure that the screw holes are coincident.

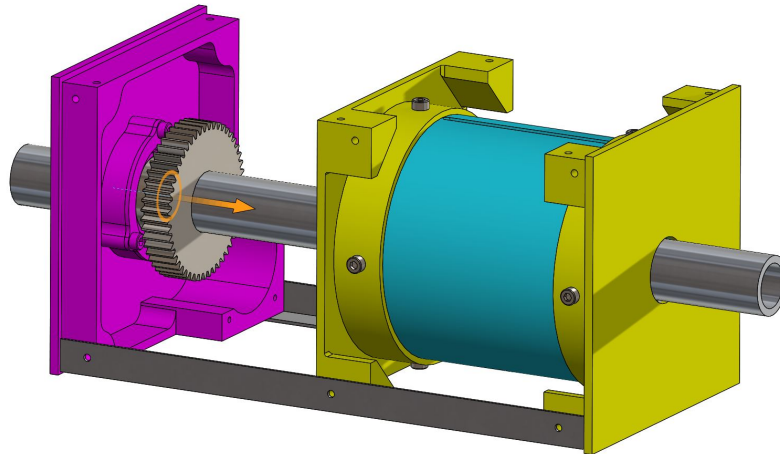


Figure 14: Thread the shaft through the left lid and push the lid towards the drivetrain sub assembly until it hits the L-profiles. Parts of the flange on the gear should be in the bearing at this point. Ensure that the screw holes are coincident and the motor holder is at the bottom.

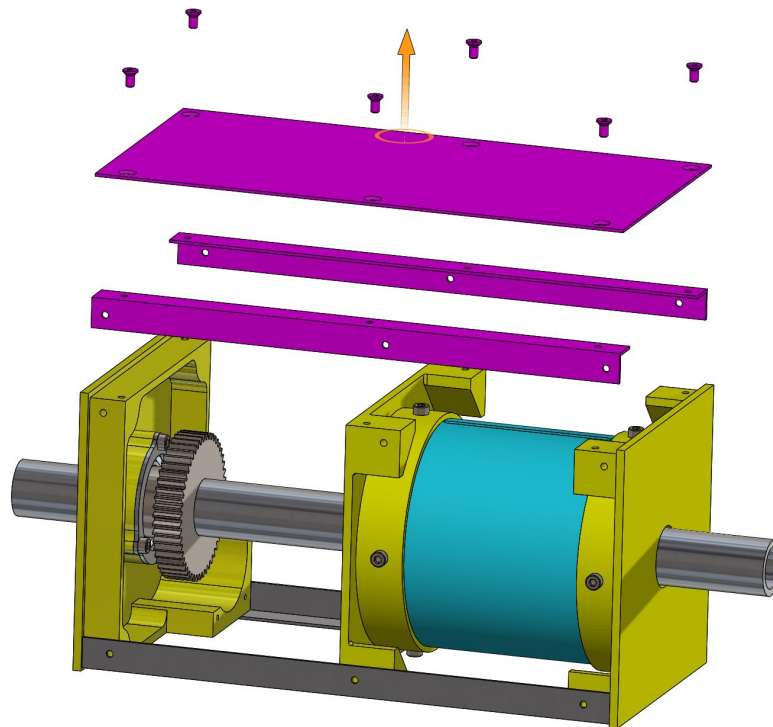


Figure 15: Fasten one of the flat walls to the top of the assembly, with two L-profiles and six socket countersunk head screws.

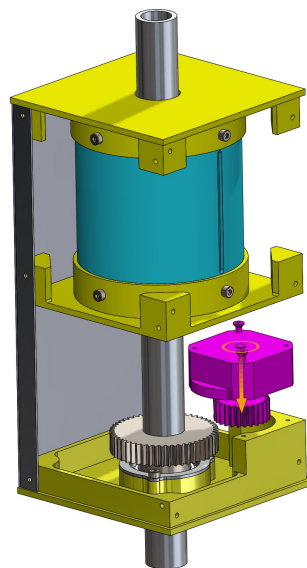


Figure 16: Mount the remaining gear to the motor and fasten the motor to the structure of left lid, with two socket countersunk head screws.

Solder the wires from the motor to the D-sub with nine pins.

The two 15 pin D-sub connectors are soldered to the wiring coming from the Flex connector. One D-sub for each of the wiring sets coming from their respective Flex.

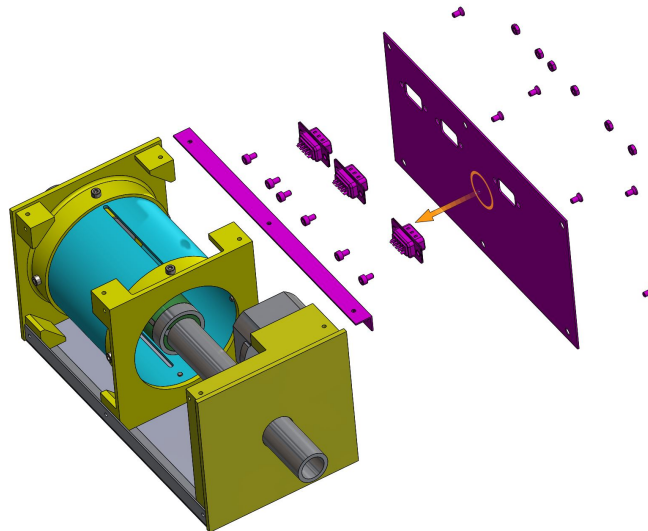


Figure 17: Mount the D-sub's that are soldered with wires to the designated flat wall, with two socket head cap screws and two nuts each. Place one of the L-profiles on the corner closest to the motor so that the screw holes are coincident with the holes on the assembly. Fasten the wall to the assembly with six socket countersunk head screws.

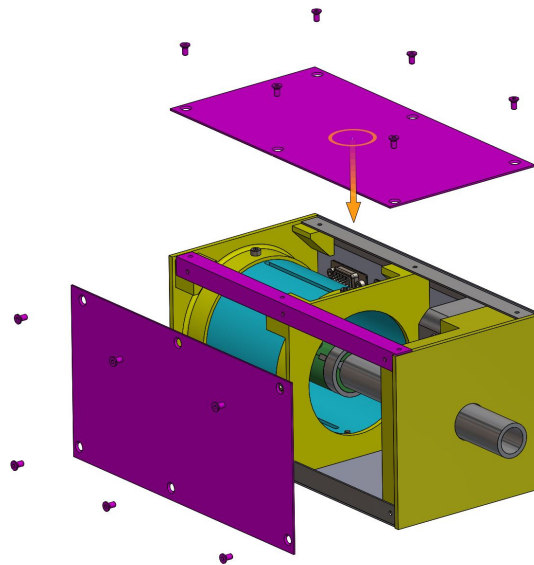
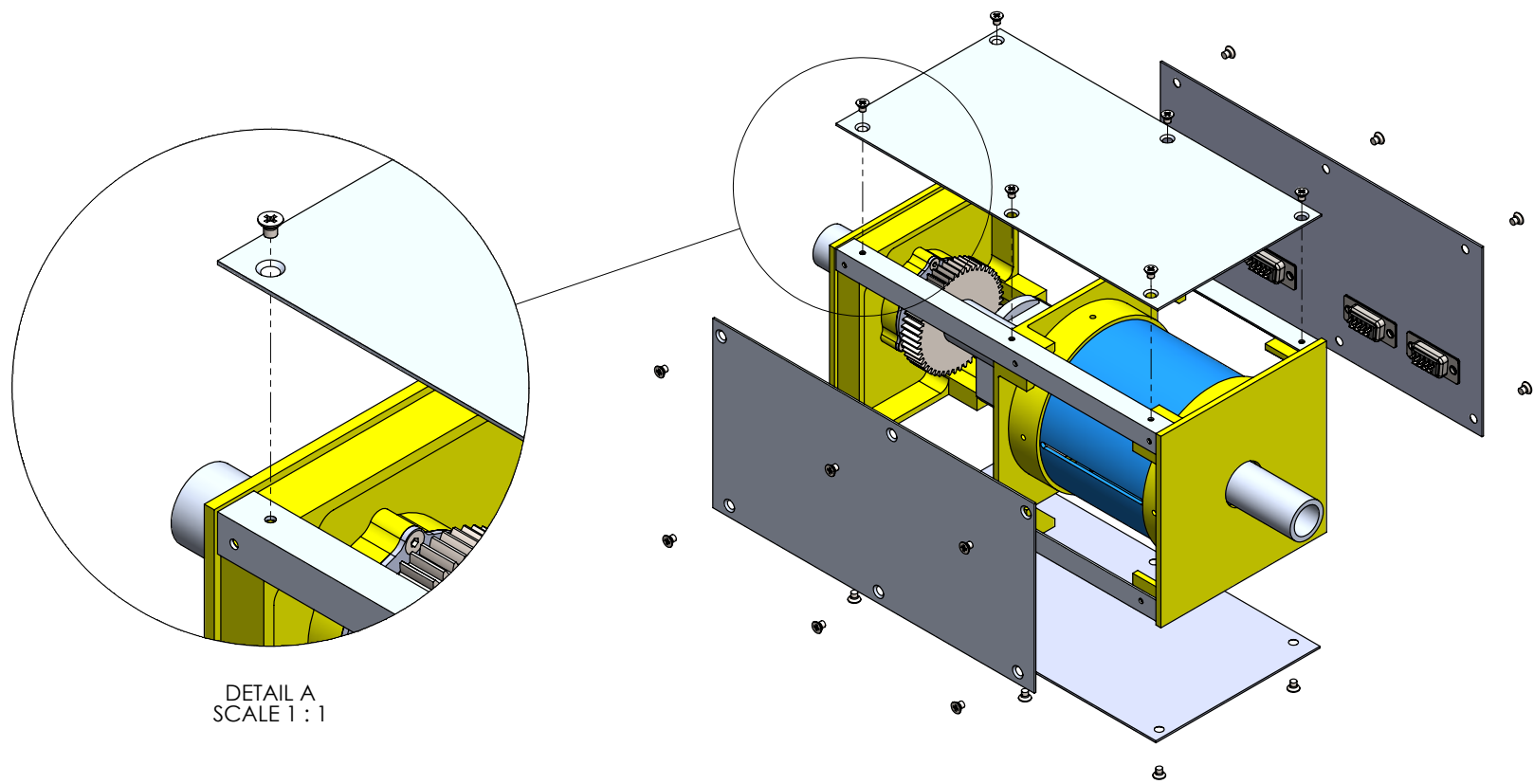


Figure 18: Place the last L-profile on the last corner and fasten the remaining walls to the assembly with the rest of the socket countersunk head screws.



DETAIL A
SCALE 1 : 1

ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	Left Lid - M3	Left Lid with motor holder	1
2	Right lid	Right Lid - Twist Capsule Side	1
3	Wall B, middle 100x100	Middle Wall	1
4	Wall 200x100	Panel Wall	3
5	Wall 200x100 With Dsub holes	Back Wall with Dsubs	1
6	L-profile	L-shaped corners	4
7	DIN EN ISO 7046-1 - M3 x 4 - Z - 4N	M3 Cross recessed screws	24

The Remaining Structure



Bachelor of Engineering

Project appendix

Winter semester 2019

Bonding test TV-30, RV-18, RV-19

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Erica Fegri and Håvar Østrem
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This report documents bonding test according to verification ID's TV-30, RV-18 and RV-19.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

Contents

1	Verification description	2
2	Results	3
3	Conclusion	4
4	Mark3 Structure	4

1 Verification description

This test is a bonding test, which is performed to verify that the contact resistance between metallic parts of the SADM are sufficiently low. This test is performed with 4 wire method/1A in Division Space and Surveillance's test facilities and instruments. The test is not performed when SADM is fully assembled due to practical causes, but includes selected transitions. It should give an indication if the fully assembled product provides sufficiently low contact resistance between the metallic parts.

Verification ID	TV-30, RV-18 and RV-19
Date	20.05.2019
Test Responsible	Erica Fegri and Håvar Østrem
Supervisor	Lars Helge Surdal
Test instrument	Hoioki 3548
Certificate date	Date: 27.02.19
Cert. No.	19-474709
Customer ID:	1-191573

Table 1

The requirements that are verified in this test is the following:

Requirement-ID	Description
R5-15	Metallic parts of each electrical equipment chassis (case) shall be mutually bonded together by direct metal contact (preferred method). Bonding interfaces shall be designed to achieve contact resistance as follow: - < 2.5mΩ for 1 junction - < 4.3mΩ for 2 junctions - < 5.9mΩ for 3 junctions - < 7.5mΩ for 4 junctions - < 9.0mΩ for 5 junctions - < 10.6mΩ for 6 junctions
R5-17	The bonding test shall be performed with the SADM switched off. The measurements shall be done with the four-wire method and 1A and with both directions of polarity [...].
R5-18	The bonding test shall be performed without any harness connection

Table 2

2 Results

Left lid - bottom front L-profile	1 junction	0.72mΩ
Middle wall - bottom front L-profile	1 junction	2.95mΩ
Right lid - bottom front L-profile	1 junction	0.32mΩ
Wall top - top front L-profile	1 junction	0.91mΩ
Wall top - Left lid	2 junctions	0.61mΩ
Wall top - middle wall	2 junctions	0.67mΩ
Wall top - right lid	2 junctions	0.49mΩ
Right lid - middle wall	2 junctions	0.44mΩ
Middle wall - left lid	2 junctions	0.45mΩ
Right lid - left lid	2 junctions	0.33mΩ
Motor chassis - left lid	1 junction	0.30mΩ
Shaft - left lid	2 junctions	∞mΩ

Table 3: Test results. See 4 for drawing.

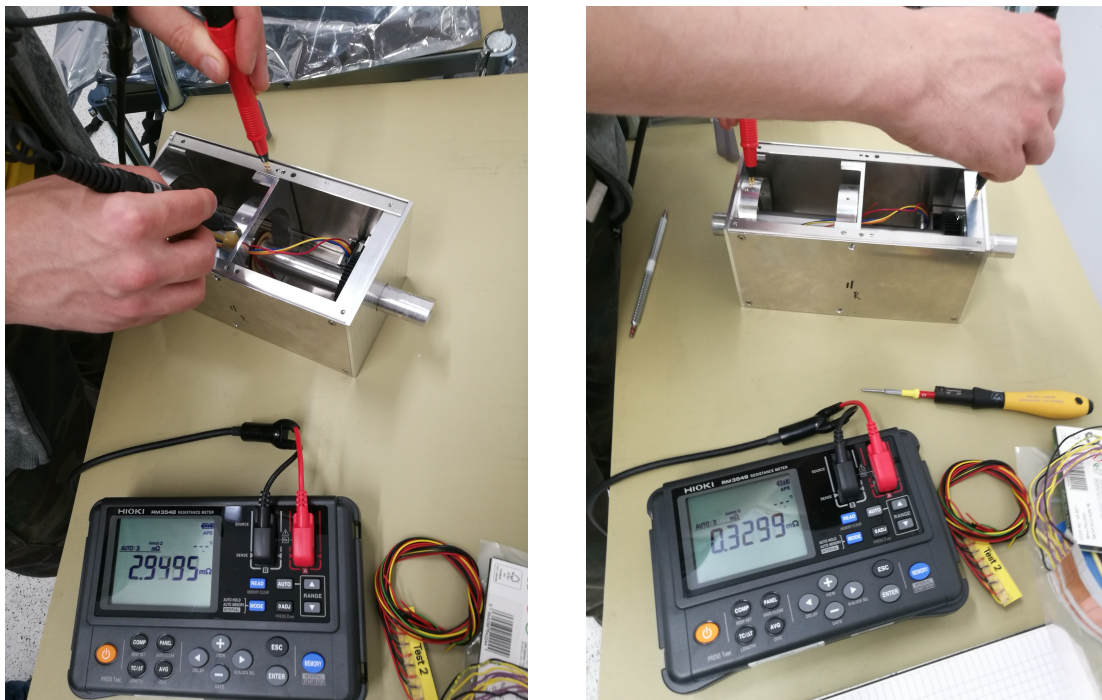


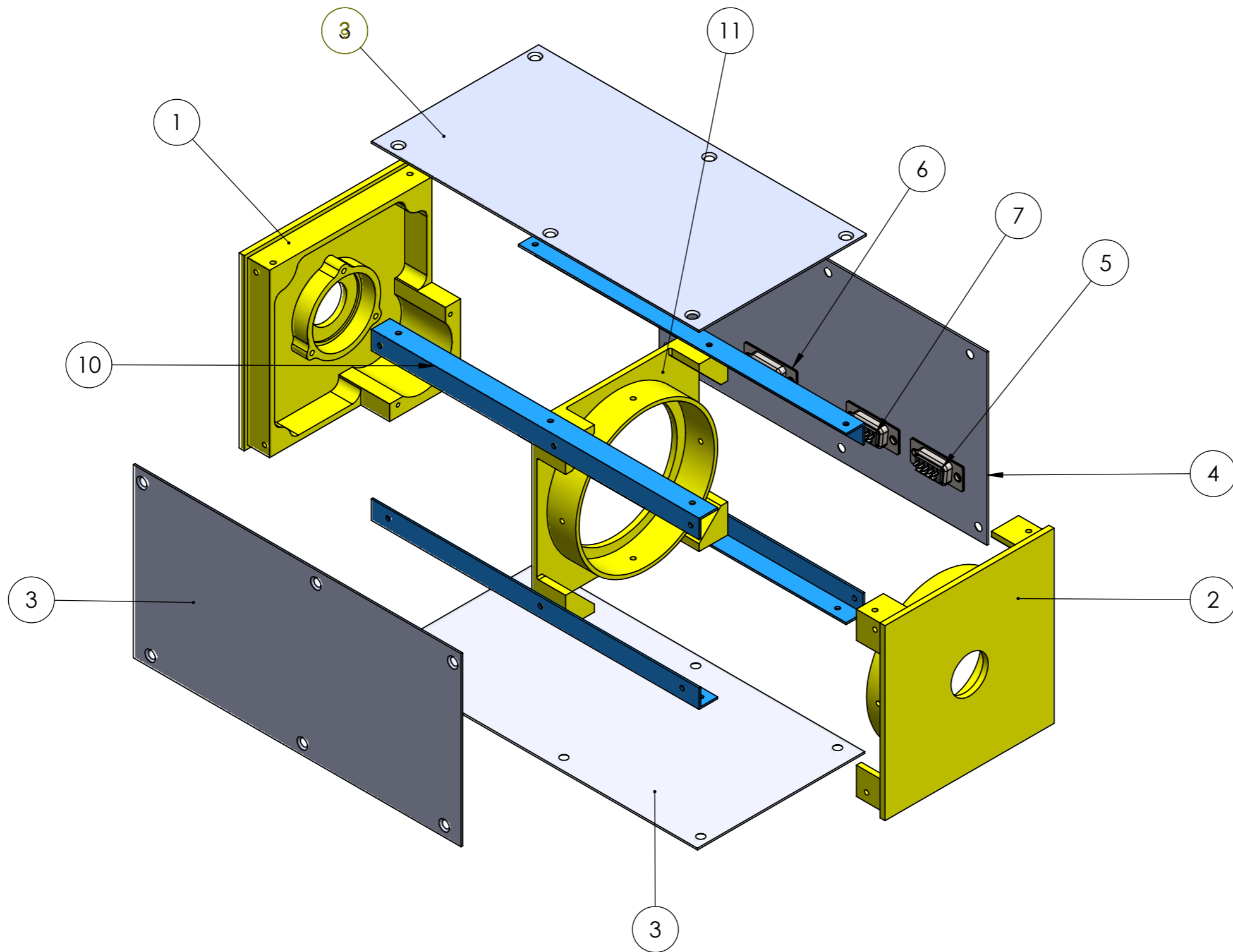
Figure 1: Examples of how the test is performed.

3 Conclusion

The test resulted in very low contact resistance. The test *Middle wall - bottom front L-profile* results in a value that is too high according to the requirement R-15. Considering that these parts will be bonded by another three screws, the deviation of $0.45m\Omega$ is considered as acceptable as the screws will both function as bonding and provide directly contact between the parts when tighten.

The test *Shaft - left lid* is infinite, their only physical contact with the schassis is trough bearings. The test is preformed without harness as specified in R5-18, and since the bearing provides a large contact resistance this establish that the shaft needs redundant grounding.

The test is considered as successful.



ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	Left Lid - M3	Left Lid	1
2	Right lid	Right Lid	1
3	Wall 200x100	Panel Wall	3
4	Wall 200x100 With Dsub holes	Panel Wall - Satellite Interface, with power transfer	1
6	D-T	Dsub	3
10	L-profile	L Shaped Wall Supports	4
11	Wall B, middle 100x100	Middle Wall	1

Bachelor of Engineering

Project appendix

Winter semester 2019

Electrical Schematics - DS HWL

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Erica Fegri
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document is the electrical schematics of the HWL control cabinet.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

Control Cabinet

Document book

2	19.05.2019	Erica	
1	14.05.2019	Erica	
0	13.02.2019	Erica	
REV.	DATE	NAME	CHANGES
Kongsberg Space			REVISION 2
		CONTRACT :	User data 1 User data 2
			SCHEME 01

Document realized with version : 2018.0.3.18

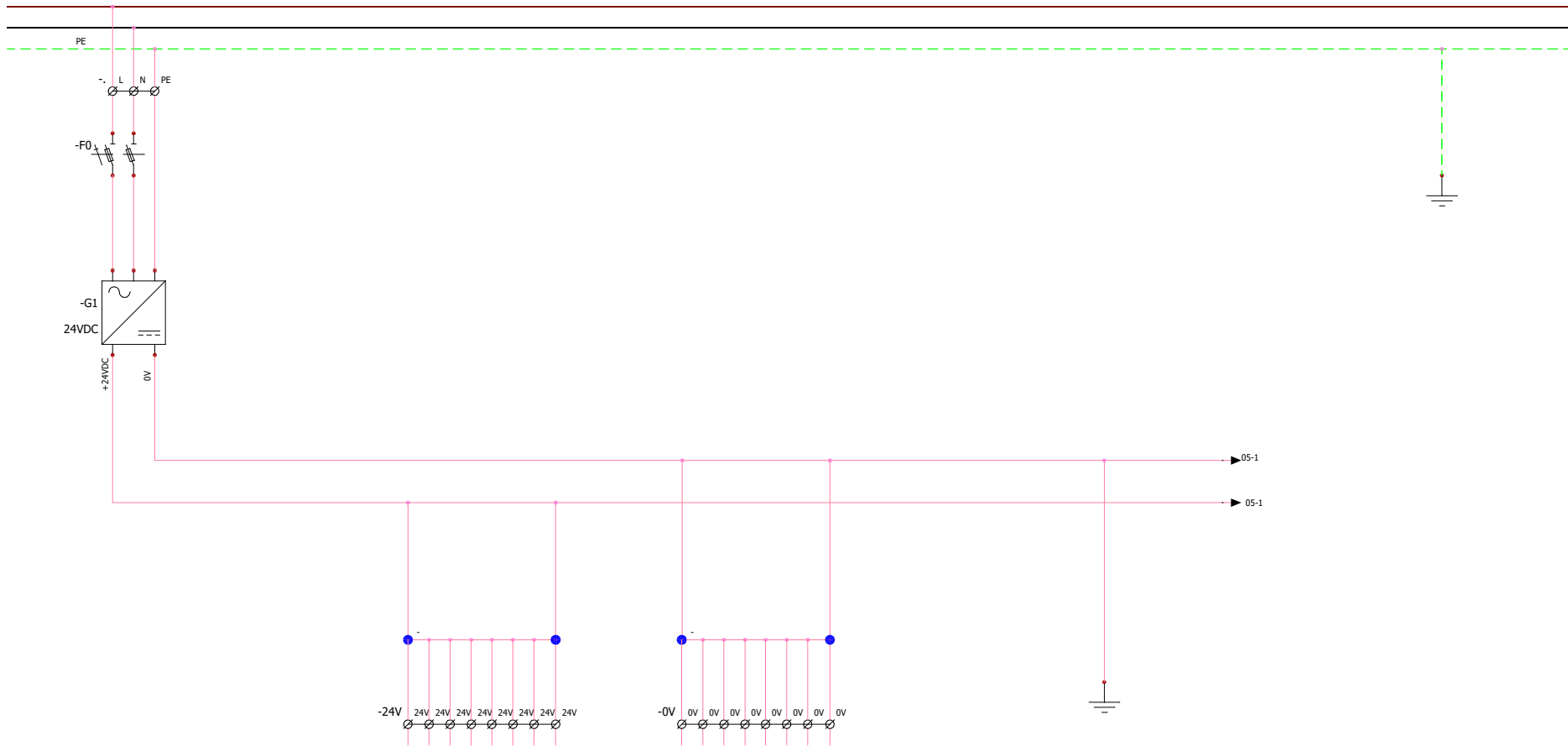
1-Document book

Drawing	Function	Location	Revision	Date	Created by	Description	Folder designation
01	=F1	+L1	1	14.05.2019	Erica	Cover page	
02	=F1	+L1	1	14.05.2019	Erica	Drawings list	
03	=F1	+L1	1	14.05.2019	Erica	Wiring line diagram	
04	=F1	+L1	1	14.05.2019	Erica	Electrical scheme	
05	=F1	+L1	1	14.05.2019	Erica		
07	=F1	+L1	1	14.05.2019	Erica		
08	=F1	+L1	1	14.05.2019	Erica		
Power transfer	=F1	+L1+SADM	1	14.05.2019	Erica		
09	=F1	+L1	0	16.05.2019	Erica		

SOLIDWORKS Electrical

		Document book				2	19.05.2019	Erica		REVISION 2
						1	14.05.2019	Erica		
						0	13.02.2019	Erica		
						REV.	DATE	NAME	CHANGES	SCHEME 02
CONTRACT:				LOCATION: +L1		Control Cabinet				
				User data 1		User data 2				

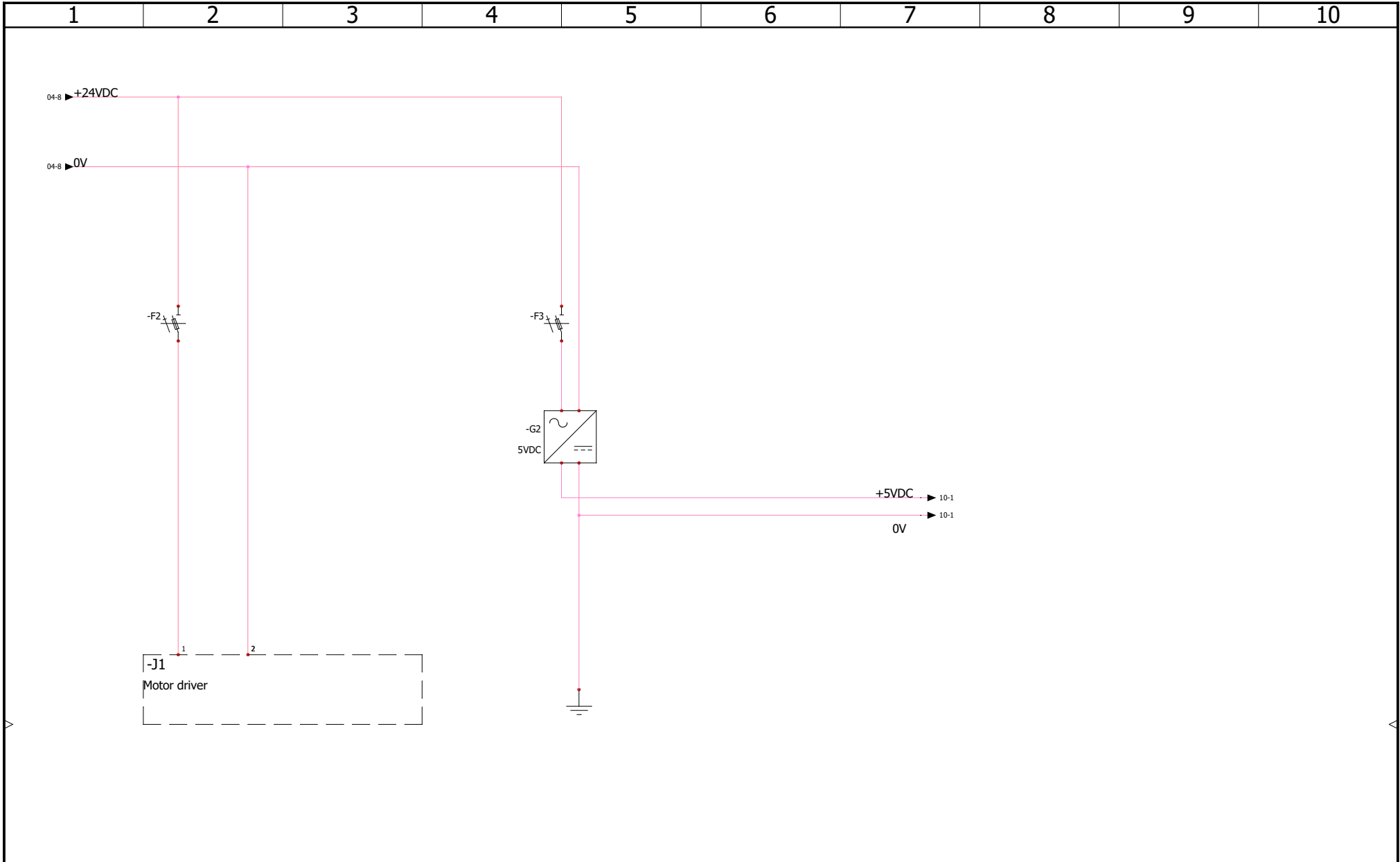
Document realized with version : 2018.0.3.18



SOLIDWORKS Electrical

CONTRACT:	LOCATION: +L1	Control Cabinet	Document book		2	19.05.2019	Erica		REVISION
			1	14.05.2019	Erica		2		
			0	13.02.2019	Erica				
			REV.	DATE	NAME	CHANGES	SCHEME		
			User data 1	User data 2				04	

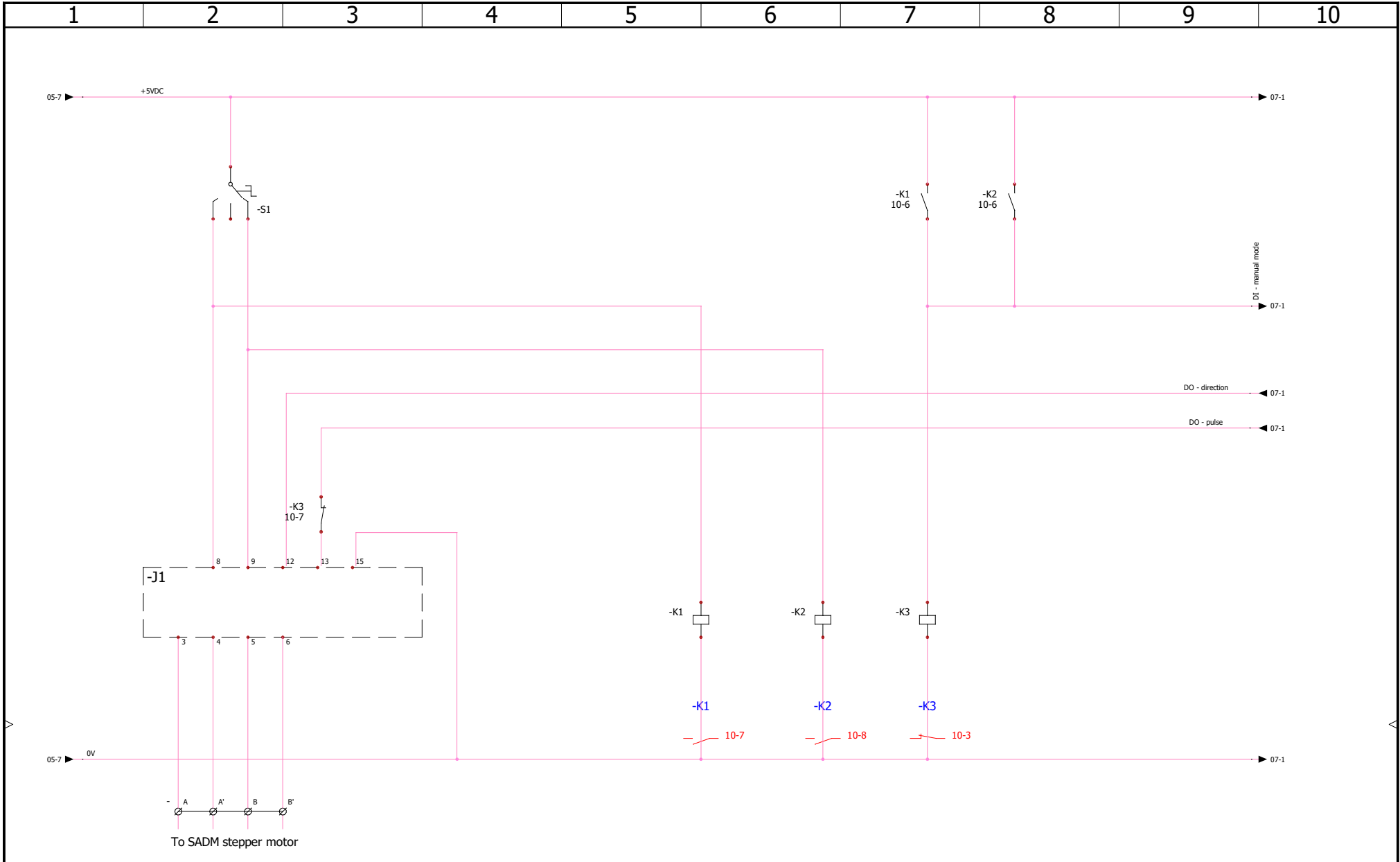
Document realized with version : 2018.0.3.18



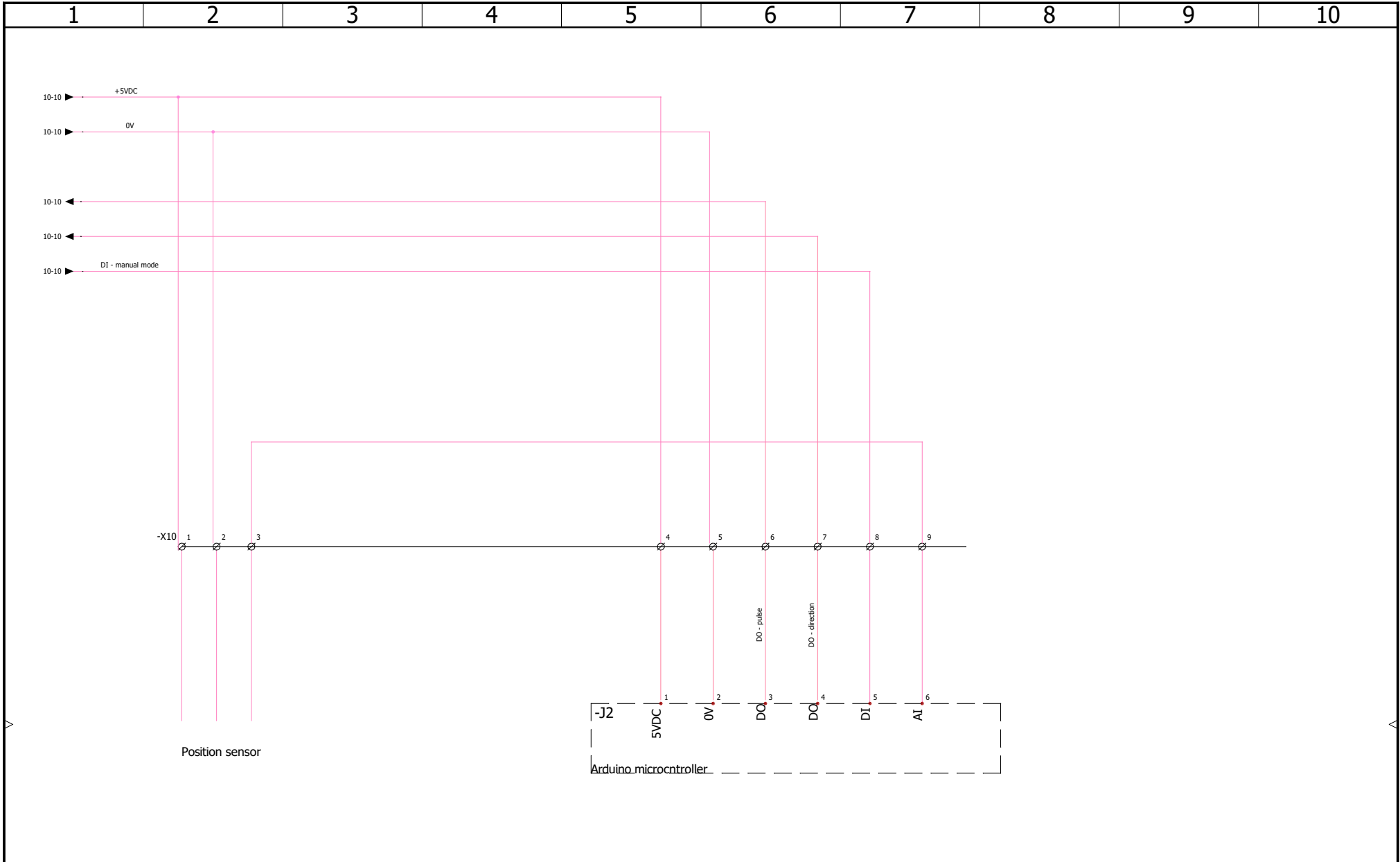
SOLIDWORKS Electrical

CONTRACT:		LOCATION: +L1		Control Cabinet		Document book		2	19.05.2019	Erica		REVISION
						1	14.05.2019	Erica		2		
User data 1		User data 2		0	18.02.2019	Erica		REV.	DATE	NAME	CHANGES	SCHEME
								05				

Document realized with version : 2018.0.3.18



Document book		REVISION									
		2	19.05.2019	Erica	2						
CONTRACT:		LOCATION: +L1		Control Cabinet		User data 1		User data 2		SCHEME	
		REV.		DATE		NAME		CHANGES		10	



CONTRACT:		LOCATION: +L1		Control Cabinet		2		19.05.2019	Erica	REVISION	
						1		14.05.2019	Erica		2
						0		18.02.2019	Erica		
						REV.	DATE	NAME	CHANGES	SCHEME	
				User data 1		User data 2			07		

Bachelor of Engineering**Project appendix****Winter semester 2019****Mail correspondence - OEM Motor**

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Erica Fegri
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains the mail correspondence with the OEM Motor sales office.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

Fra: Erica Fegri
Sendt: fredag 15. mars 2019 kl. 16:35
Til: Torgny Berglund
Emne: SV: phyBasic step motor - forespørsel på pris

Tusen takk for hurtig svar.

Kan vi stille parametere kun med bruk av de digitale inngangene, eller er vi nødt til å ha den her? Vi ønsker å starte/stoppe og regulere hastighet kun ved bruk av digitale innganger på driver.

EM-328 460 SEK Parametersättningskabel

Med vennlig hilsen
Erica Fegri

Fra: [Torgny Berglund](#)
Sendt: fredag 15. mars 2019 kl. 11:47
Til: ericafegri@gmail.com
Emne: phyBasic step motor - forespørsel på pris

Hej Erica

Jag skulle rekomendera SS2421-5041 Sanyo Denki stegmotor, Denna motor är väldigt kompakt Nema17 motor, endast 11mm.

7404201060 508 SEK Motor
EM-KP72-87.5 112 SEK Dinskenfäste
EM-314-12-24V 1379 SEK Styrning
EM-328 460 SEK Parametersättningskabel

Du hittar information om produkterna via dessa länkar:

<https://www.oemmotor.se/produkter/motorer/stegmotor/ss242--648521>

<https://www.oemmotor.se/produkter/drivelektronik/stegmotorstyrning/em-314--449479>

Med vänliga hälsningar

Torgny Berglund

Product Manager / Drive Electronics & Motors

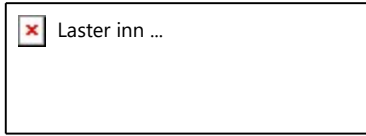
► DIRECT +46 (0)75-242 44 26

E-MAIL torgny.berglund@motor.oem.se

► ADDRESS Box 1011, 573 28 Tranås, Sweden

SWITCH +46 (0)75-242 44 00 FAX +46 (0)75-242 44 49

► WEB www.oemmotor.se



From: Erica Fegri <ericafegri@gmail.com>
To: Oscar Nyström <oscar.nystrom@oemmotor.se>
Date: 2019-03-15 10:20
Subject: SV: phyBasic step motor - forespørsel på pris

Hei igjen Oscar,

Tusen takk for svar.

Jeg har vært og kikket litt på motorer dere har, og vil gjerne spørre deg om du kan anbefale noen basert på noen kriterier?

- 2 fase, bipolar, 1.8 grader full stepvinkel
- Standard NEMA-dimensjoner
- Dreimoment ca 40 mNm.
- Så liten og lav vekt som mulig

Ønsker også tilhørende motordriver, 24V DC matespenning med DIN-skinnefeste om mulig.

Med vennlig hilsen
Erica Fegri

Fra: [Oscar Nyström](#)

Sendt: mandag 11. mars 2019 kl. 11:00

Til: [Erica Fegri](#)

Emne: Re: phyBasic step motor - forespørsel på pris

Hej Erica,

Om det är så att ni är ute efter en standard stegmotor utan specialdata som till exempel:

- * Höga/låga temperaturer
- * Vakumm
- * Radiation
- * Vibrationer

Då skulle jag rekommenderar vårt standardprogram av stegmotorer och styrningar på www.oemmotor.se.

Detta blir både billigare och snabbare leveranser.

Med vänliga hälsningar / Best regards

OSCAR NYSTRÖM

Business Area Manager

Drives and Motors

- ▶ DIRECT +46 (0)75-242 44 35
- E-MAIL oscar.nystrom@motor.oem.se
- ▶ ADDRESS Box 1011, 573 28 Tranås, Sweden
Switch +46 (0)75-242 44 00 FAX +46 (0)75-242 44 49
- ▶ WEB www.oemmotor.se

From: Erica Fegri <ericafegri@gmail.com>
To: "oscar.nystrom@oemmotor.se" <oscar.nystrom@oemmotor.se>
Date: 2019-03-06 16:49
Subject: phyBasic step motor - forespørsel på pris

Hei,

Jeg fikk din mailadresse av Eirik Normann fra Kongsberg Space and Surveillance.

Vi er en gruppe studenter som skriver bacheloroppgave for Kongsberg Space and Surveillance, og trenger i den forbindelse en steppermotor og tilhørende driver.

Vi har sett på følgende motorer;

phyBASIC-28-1-200-1,0-4Lp-MOLEX-300 (Nema 11)
phyBASIC-42-1-200-1,5-4Lp-JST-300 (Nema 17)

Kan vi få et pristilbud på 1 stk av hver av disse (til KSS), samt forslag/pris på tilhørende driver?

Vi har i utgangspunktet mellom 17-19 VDC som matespenning til motor og driver.
Dersom dere har brakett/sokkel til driver for DIN-skinne monterer hadde dette vært supert.
Når det kommer til «mating connector» MOLEX eller JST tar jeg gjerne i mot tips til hva som er fordel ulempe med hver av dem og hvilken vi bør velge.

På forhånd takk.

Med vennlig hilsen

Erica Fegri
Ingeniørstudent – Mekatronikk og kybernetikk
Universitetet i Sørøst-Norge
Tlf: 95768624

Bachelor of Engineering

Project appendix

Winter semester 2019

Flex Comparison Tables

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Håvar Østrem
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains two comparison tables for the flexible PCB. One for the width and one for the length.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

Flex Width Comparison Table

Number of flexes	Flex width [mm]	Copper width (2/3 of flex width) [mm]	Flex thickness [mm]	Total Cross section [mm ²]
1	40	26.67	0.035	0.93
1	40	26.67	0.070	1.87
1	50	33.33	0.035	1.17
1	50	33.33	0.070	2.33
1	60	40.00	0.035	1.40
1	60	40.00	0.070	2.80
2	40	26.67	0.035	1.87
2	40	26.67	0.070	3.73
2	50	33.33	0.035	2.33
2	50	33.33	0.070	4.67
2	60	40.00	0.035	2.80
2	60	40.00	0.070	5.60
3	40	26.67	0.035	2.80
3	40	26.67	0.070	5.60
3	50	33.33	0.035	3.50
3	50	33.33	0.070	7.00
3	60	40.00	0.035	4.20
3	60	40.00	0.070	8.40
4	40	26.67	0.035	3.73
4	40	26.67	0.070	7.47
4	50	33.33	0.035	4.67
4	50	33.33	0.070	9.33
4	60	40.00	0.035	5.60
4	60	40.00	0.070	11.20
6	40	26.67	0.035	5.60
6	40	26.67	0.070	11.20
6	50	33.33	0.035	7.00
6	50	33.33	0.070	14.00
6	60	40.00	0.035	8.40
6	60	40.00	0.070	16.80
8	40	26.67	0.035	7.47
8	40	26.67	0.070	14.93
8	50	33.33	0.035	9.33
8	50	33.33	0.070	18.67
8	60	40.00	0.035	11.20
8	60	40.00	0.070	22.40

Flex Length Comparison Table

Radius of outer surface of the flex shaft [mm]	Radius of inside of the capsule housing [mm]	Flex Length Minimum [mm]	Shaft Mounting angle (zero) [revolutions]
10	15	122	1.667
10	20	130	1.5
10	25	138	1.4
10	30	146	1.334
10	35	154	1.286
10	40	163	1.25
10	45	170	1.222
15	20	170	1.75
15	25	178	1.6
15	30	185	1.5
15	35	193	1.429
15	40	201	1.375
15	45	209	1.334
20	25	217	1.8
20	30	225	1.667
20	35	233	1.572
20	40	240	1.5
20	45	248	1.445
25	30	264	1.834
25	35	272	1.715
25	40	280	1.625
25	45	288	1.556
30	35	311	1.858
30	40	319	1.75
30	45	327	1.667
35	40	358	1.875
35	45	366	1.778
40	45	405	1.889

Bachelor of Engineering

Project appendix

Winter semester 2019

Software layer design documents

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Thomas Mundal
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document includes the design documents for the software "SatStat" - the software layer of the Diagnostics system.

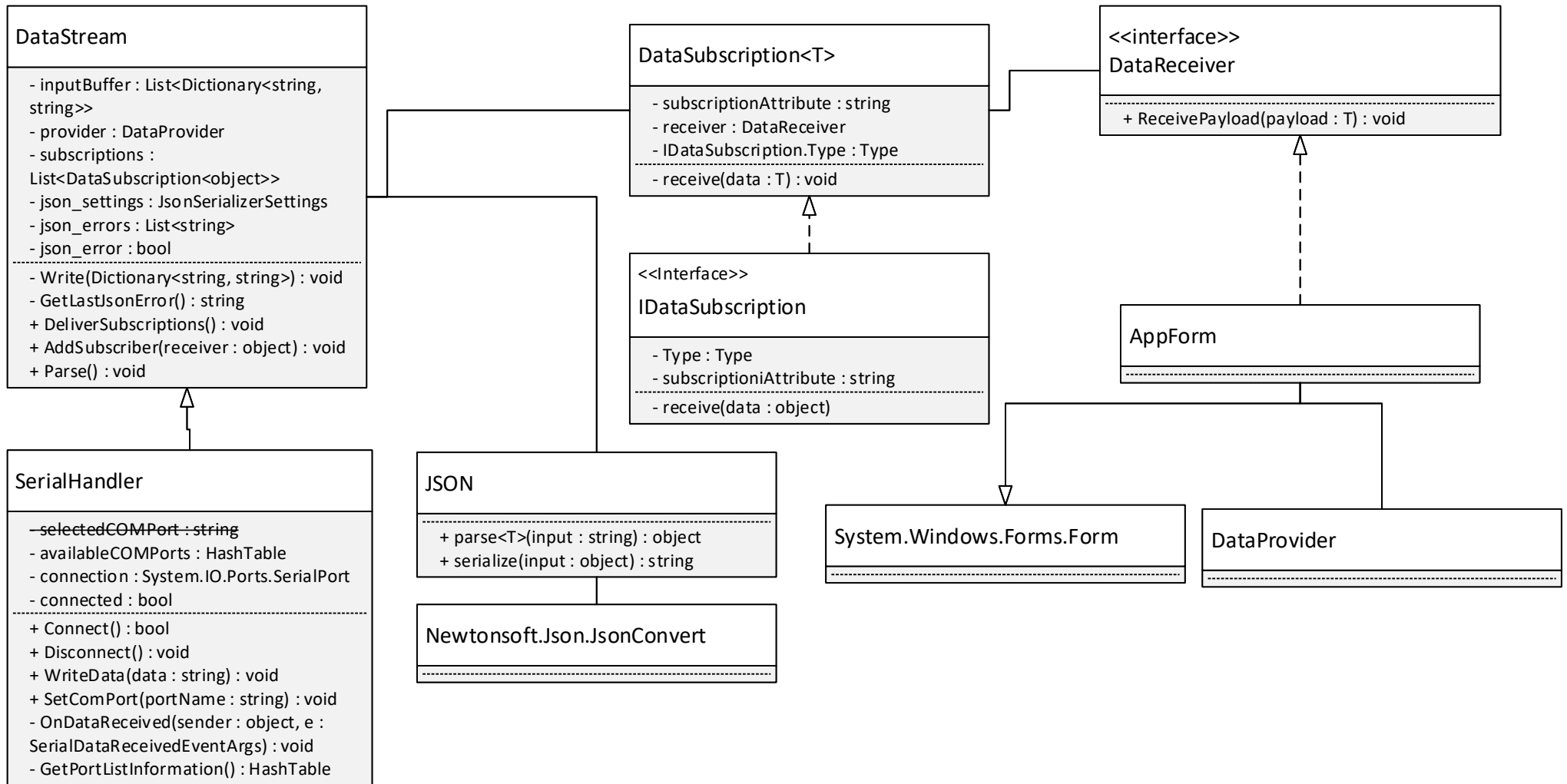
I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

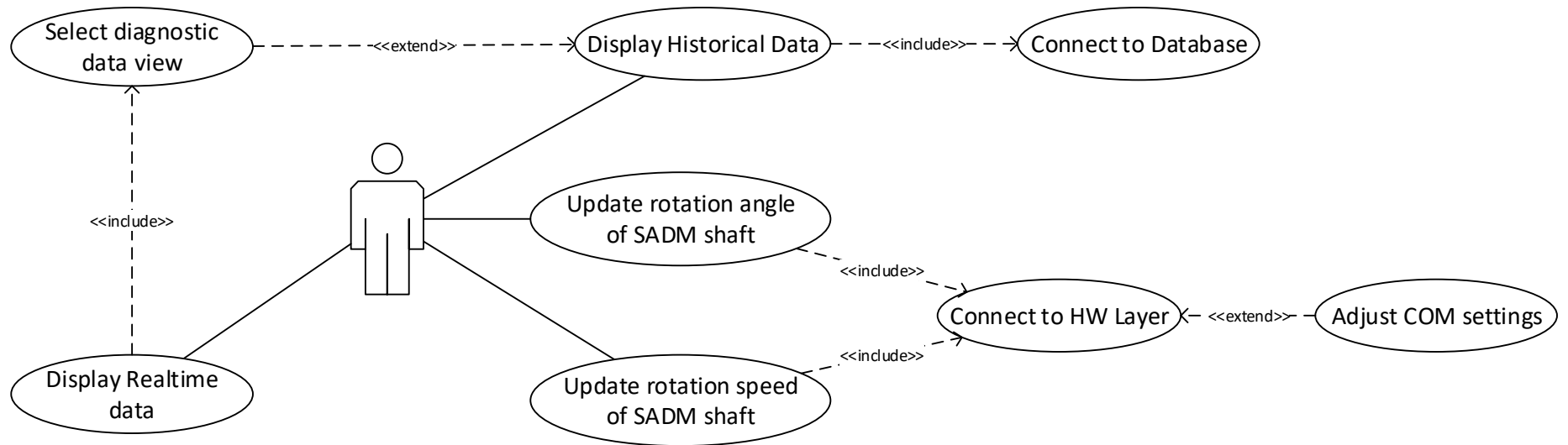
Contents

1 Software layer design documents	2
1.1 Preliminary design pre-alpha	2
1.2 Use case diagram: Alpha 1	3
1.3 Use case diagram: Alpha 3	4
1.4 Use case diagram: Beta 1	5
1.5 Use case diagram: Beta 2	6
1.6 Class diagram: Alpha 1	7
1.7 Class diagram: Alpha 2	8
1.8 Class diagram: Alpha 3	9
1.9 Class diagram: Beta 1 page 1	10
1.10 Class diagram: Beta 1 page 2	11
1.11 Class diagram: Beta 2 page 1	12
1.12 Class diagram: Beta 2 page 2	13
1.13 Alpha 2 sequence: Com Settings	14
1.14 Alpha 2 sequence: Display historical data	15
1.15 Alpha 3 sequence: Save to database	16
1.16 Alpha 3 sequence: Select diagnostic view	17
1.17 Alpha 3 sequence: Save COM settings	18
1.18 Alpha 3 sequence: Display realtime data	19
1.19 Alpha 3 Activity: Add sensor attribute	20

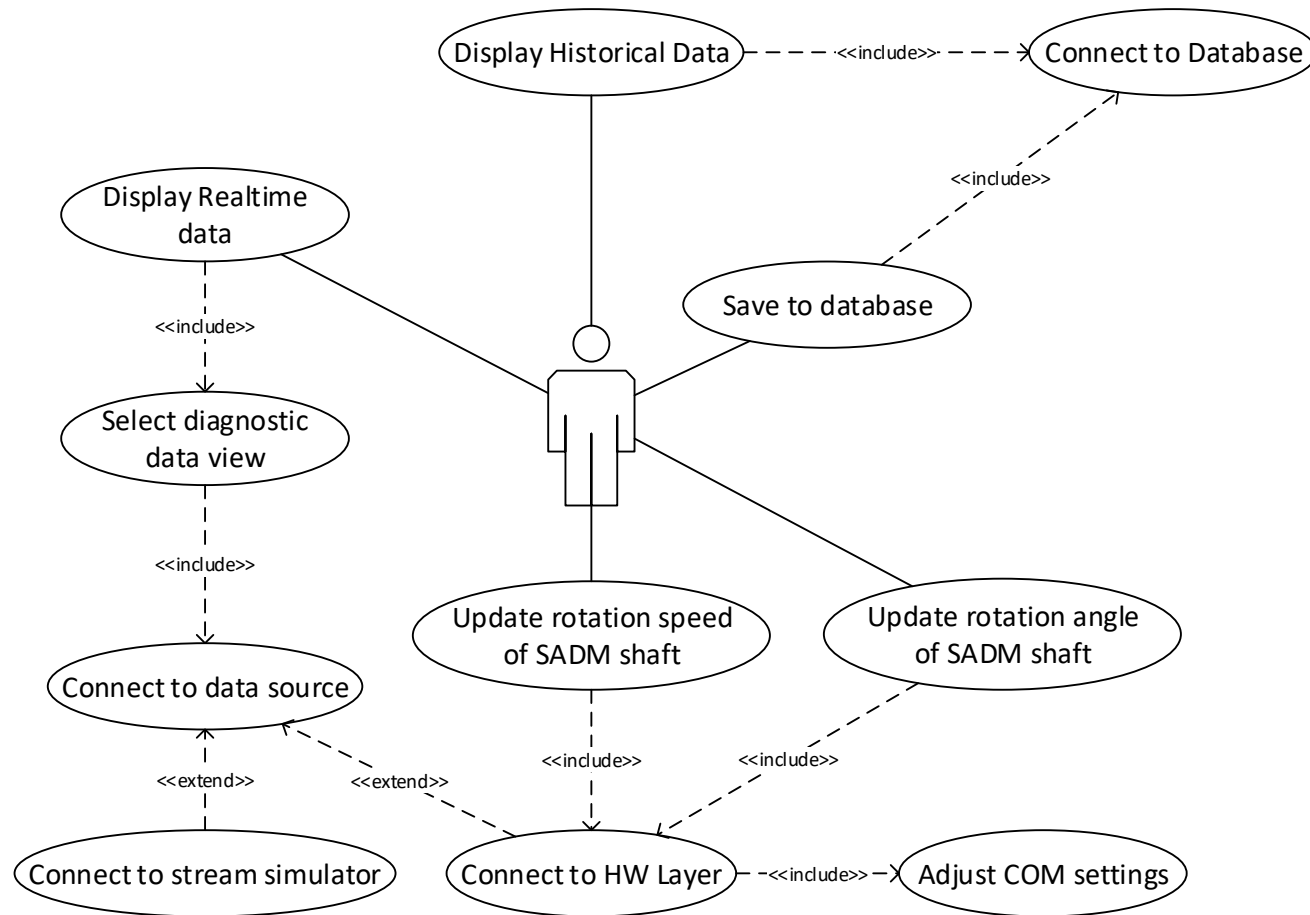
Preliminary design pre-alpha



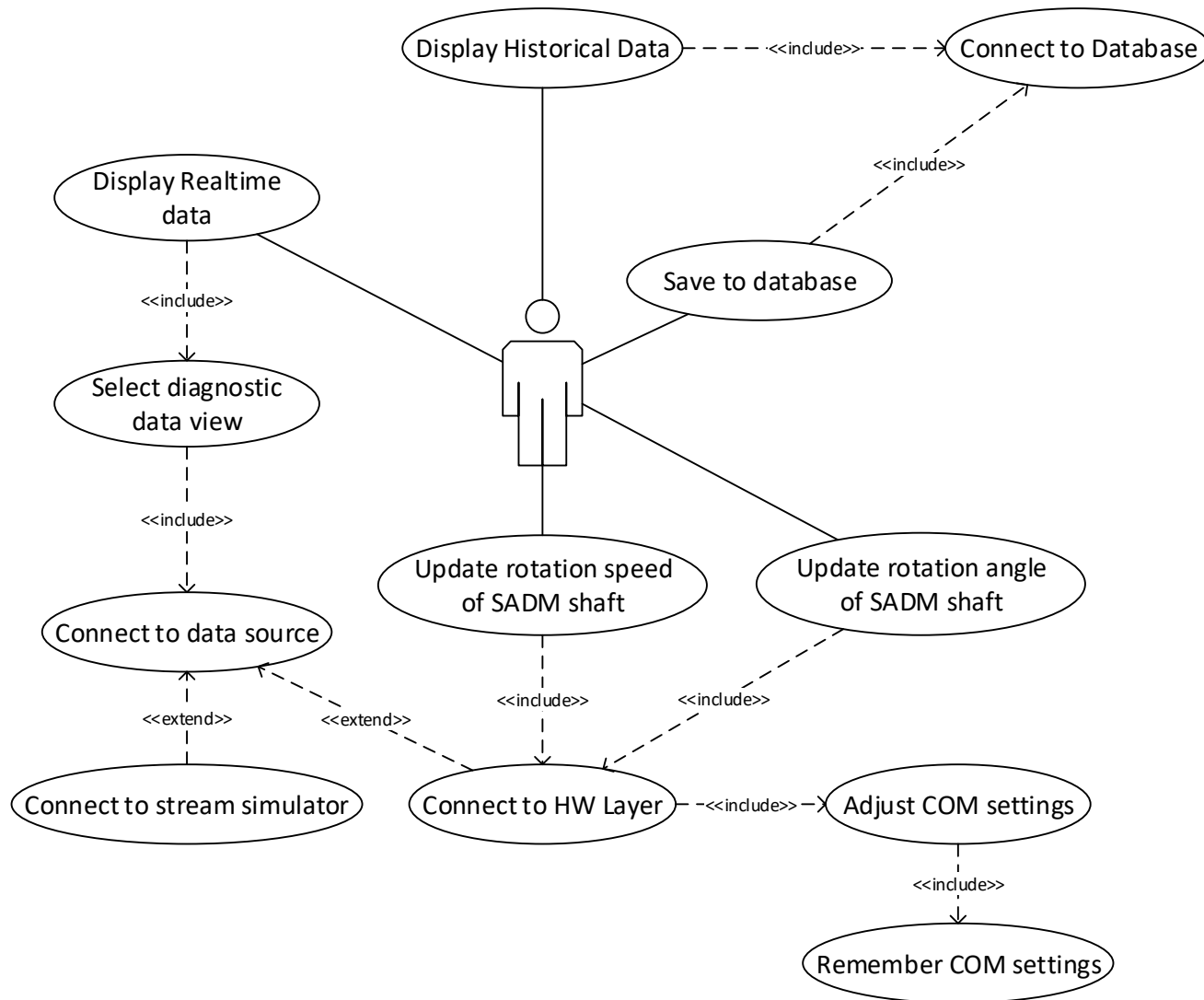
Alpha1 Use Case Diagram



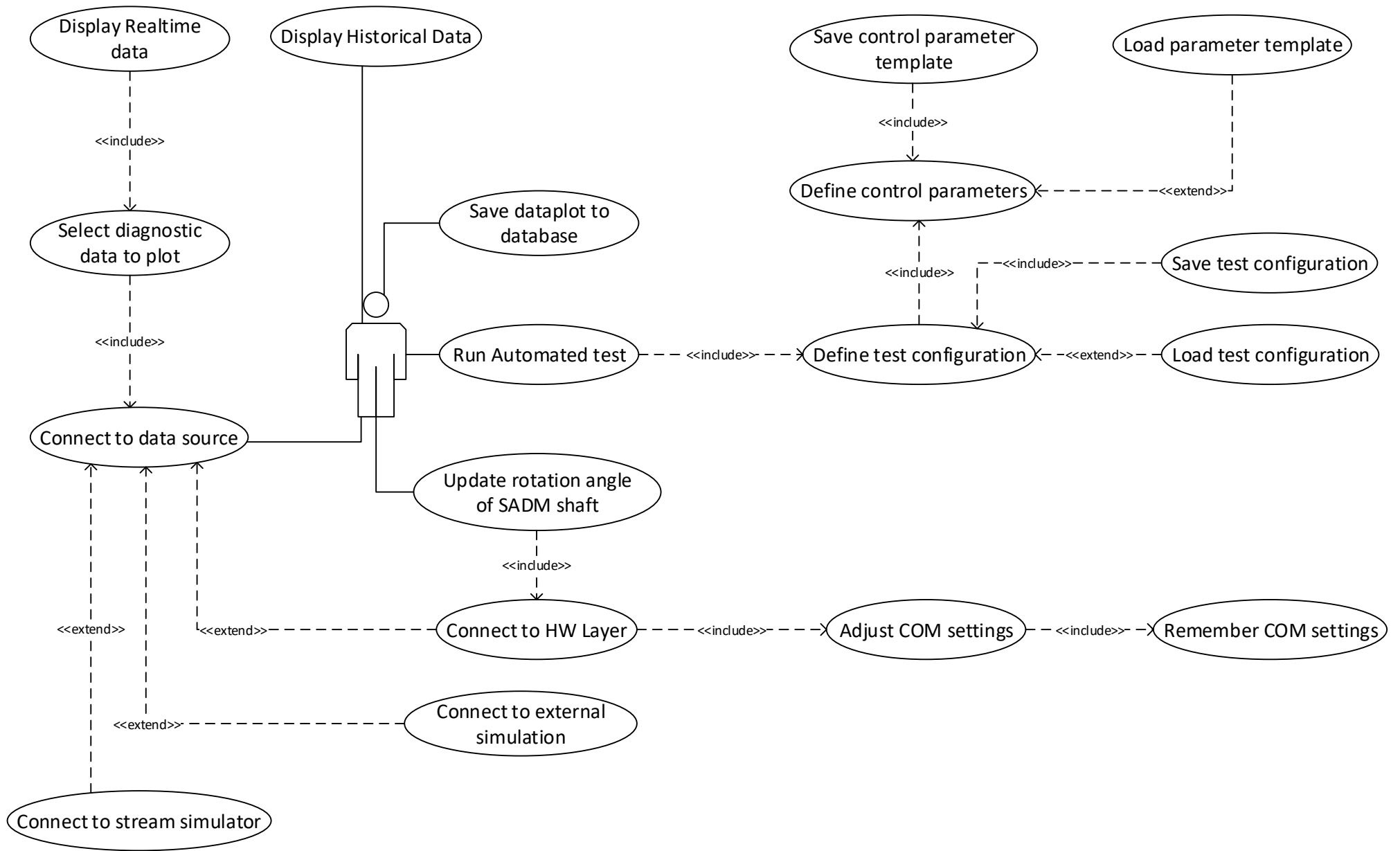
Alpha 3 Use Case Diagram



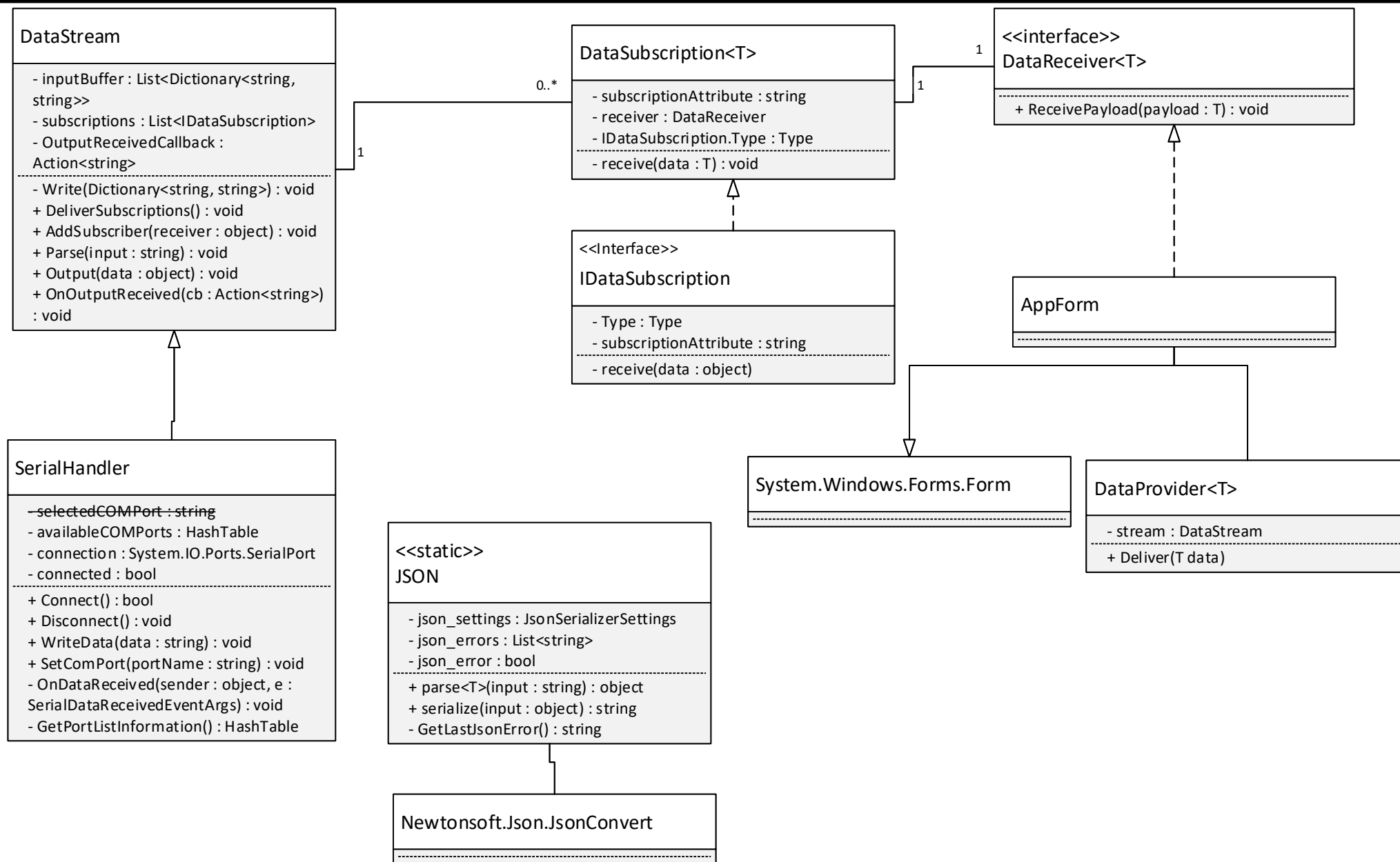
Beta1 Use Case Diagram



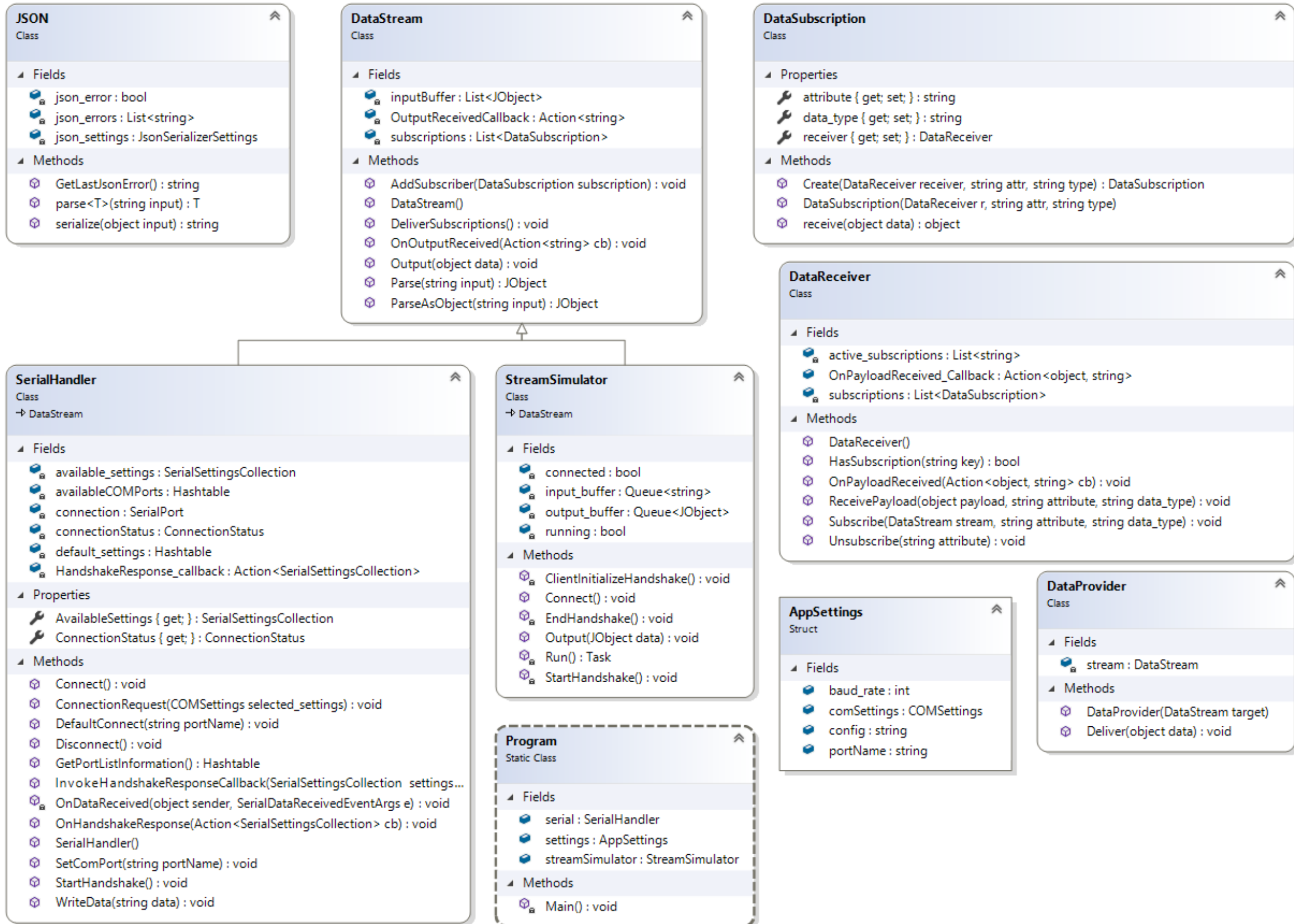
Beta 2 UseCase diagram



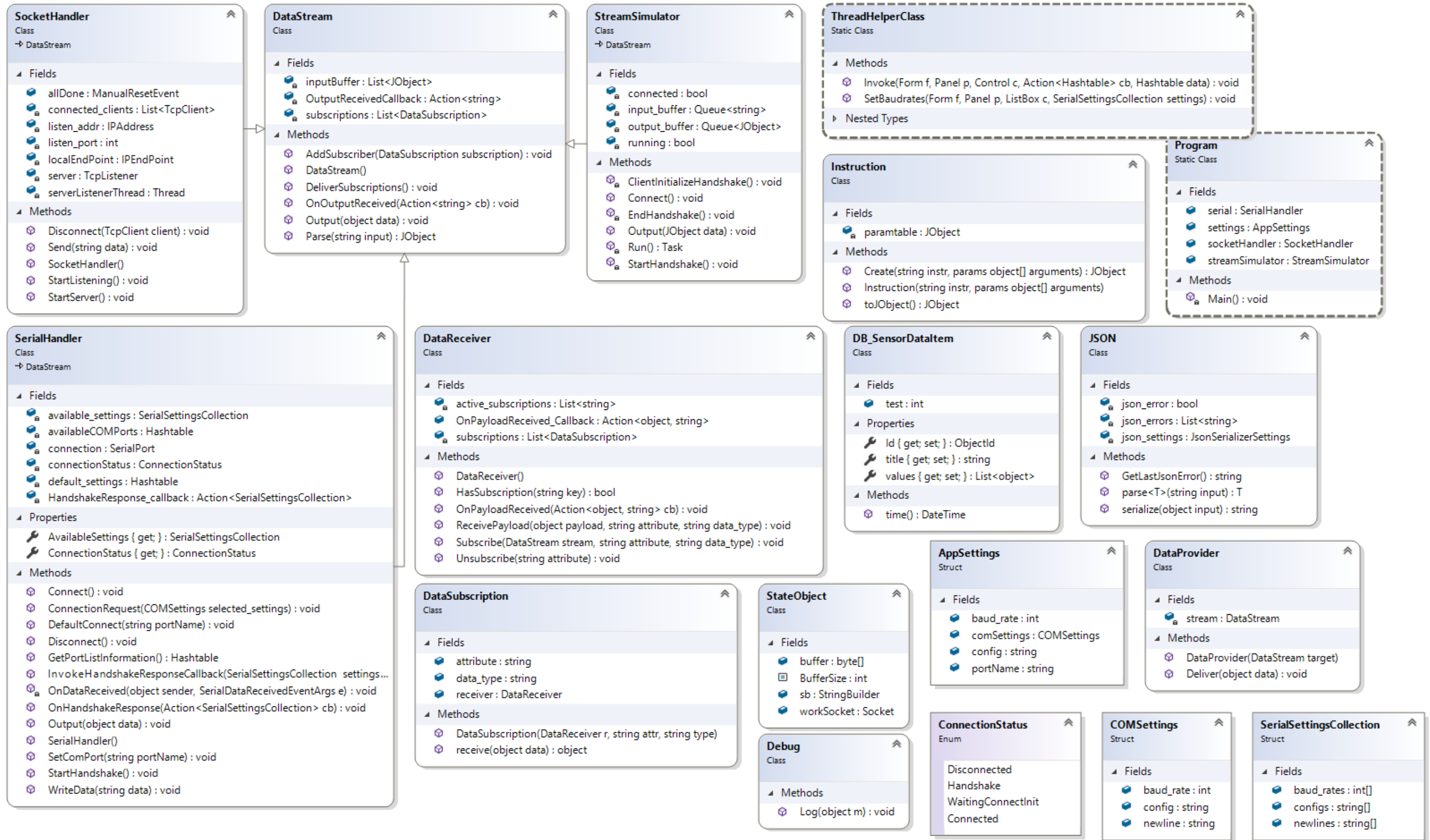
Alpha 1 Class Diagram



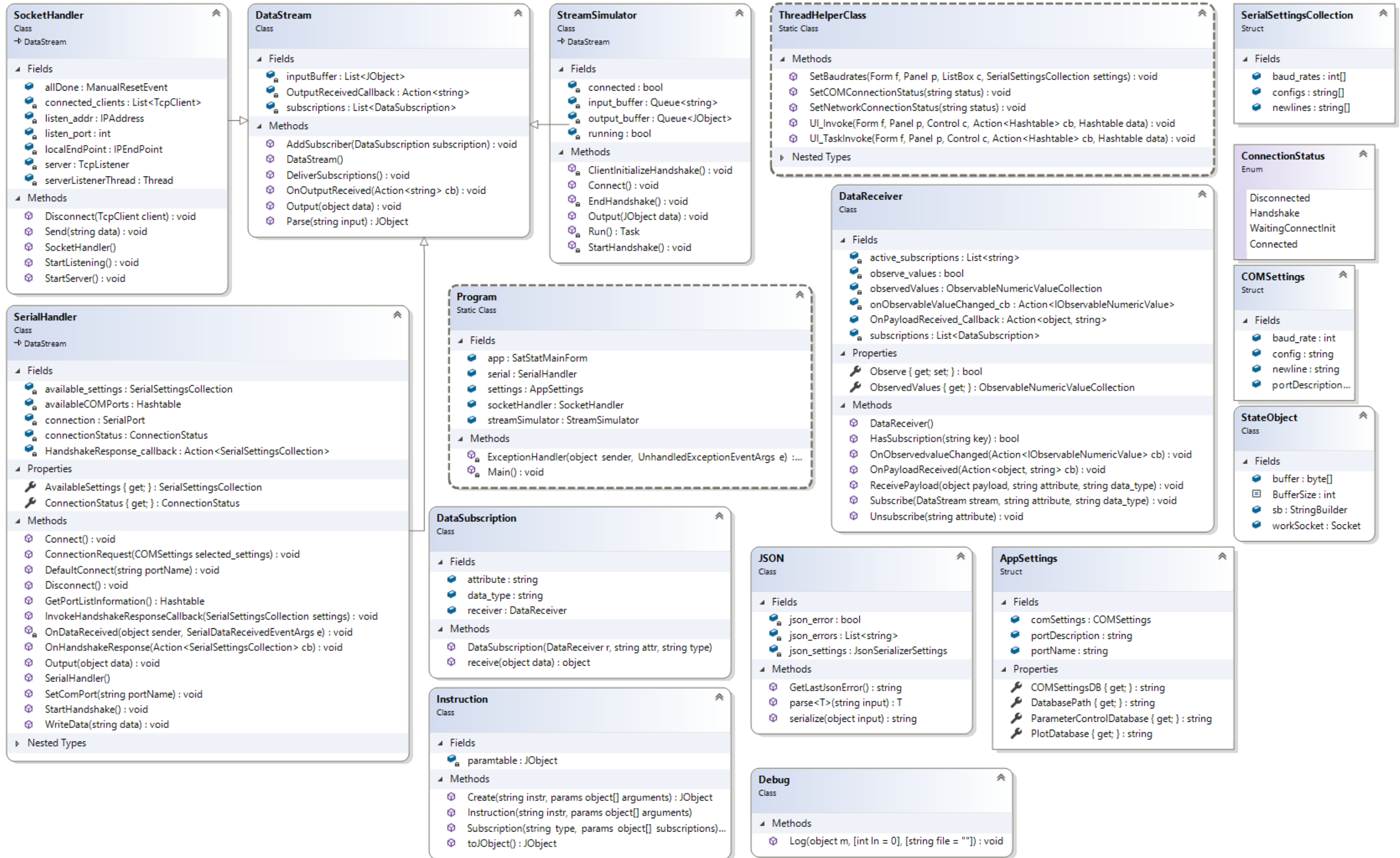
Alpha 2 Class Diagram – Exported from Visual Studio



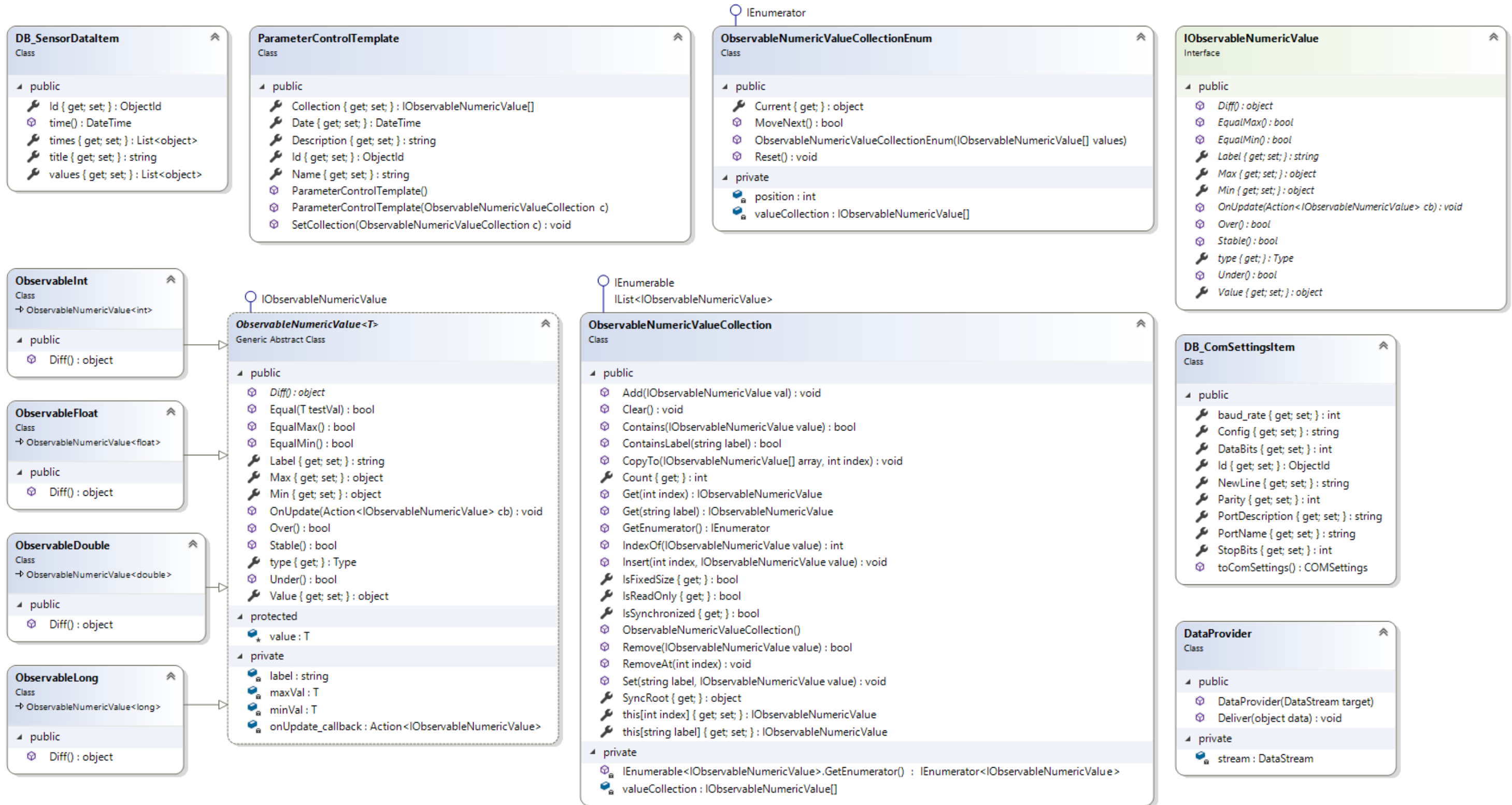
Alpha 3 – Class Diagram – Exported from Visual Studio



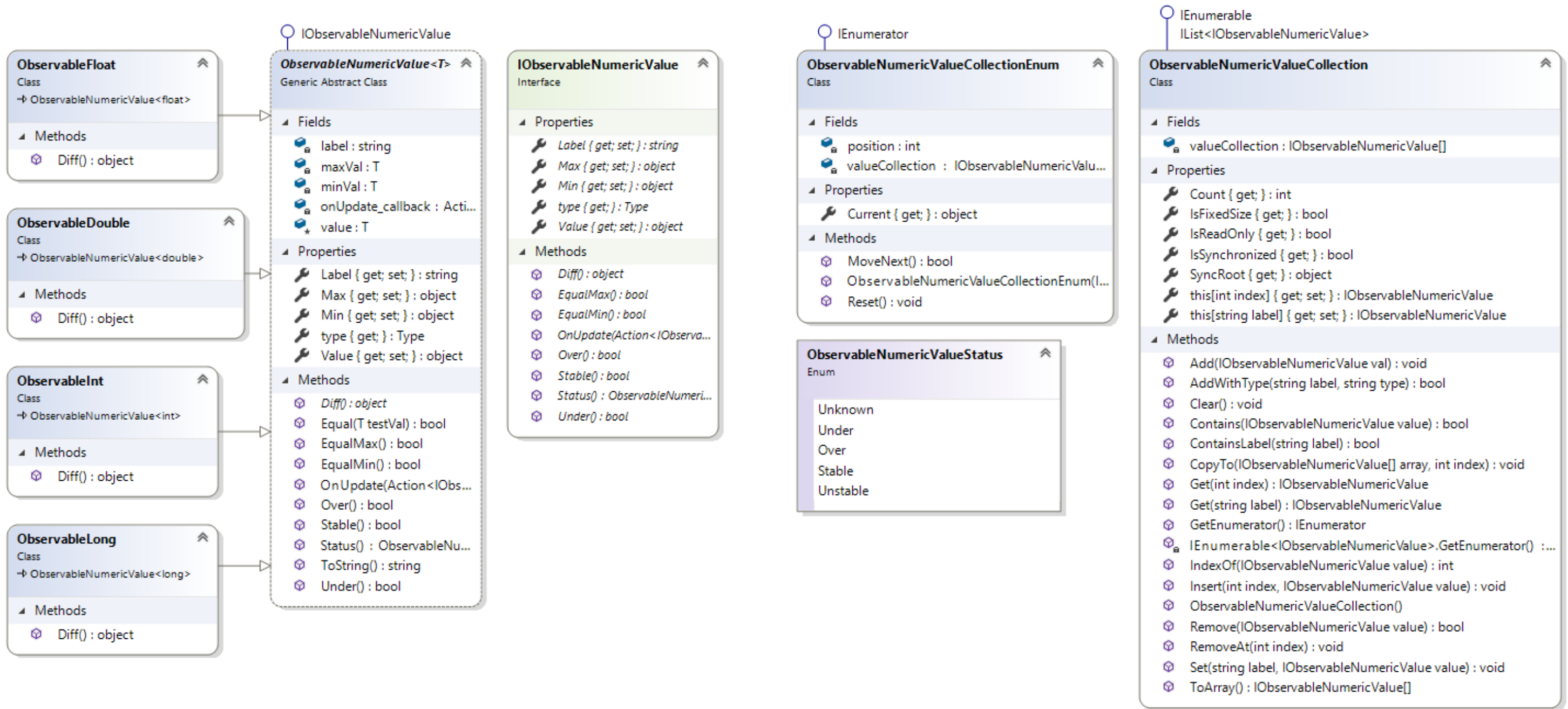
Beta 1 Class Diagram page 1 – Exported from Visual Studio



Beta 1 Class Diagram page 2 – Exported from Visual Studio



Beta 2 class diagram – page 1



TestConfiguration
Class

Fields

- currentInstructionEntry : InstructionEntry
- instructionEntryQueue : Queue<InstructionEntry>
- internalReceiver : DataReceiver
- is_running : bool
- onQueueAbortedCallback : Action
- onQueueAdvanceCallback : Action<InstructionEntry>
- onQueueCompleteCallback : Action<InstructionEntry>
- queue_position : int

Properties

- Description { get; set; } : string
- Id { get; set; } : ObjectId
- instructionEntries { get; set; } : List<InstructionEntry>
- IsRunning { get; } : bool
- Name { get; set; } : string
- ParameterControlTemplate { get; set; } : ParameterControlTemplate

Methods

- Abort() : void
- AddInstructionEntry(Instruction instr, List<string> observedLabels, [int uindex = -1]) : InstructionEntry
- AddInstructionEntry(InstructionEntry entry) : void
- HasInstructionEntries() : bool
- InstructionEntryIndex(InstructionEntry entry) : int
- loadParameterControlTemplate(ParameterControlTemplate template, DataStream stream) : void
- OnObservedValueChange(IObservableNumericValue val) : void
- OnPayloadReceive(object payload) : void
- OnQueueAbort(Action callback) : void
- OnQueueAdvance(Action<InstructionEntry> callback) : void
- OnQueueComplete(Action<InstructionEntry> callback) : void
- RemoveInstructionEntry(Instruction instr) : void
- RemoveInstructionEntry(InstructionEntry entry) : void
- Run(DataStream stream) : void
- RunQueuedInstruction(DataStream stream) : void
- Save([Action<TestConfiguration> cb = null]) : void
- setParameterControlTemplate(ParameterControlTemplate template) : void
- TestConfiguration()
- ToString() : string

InstructionEntry
Class

Properties

- instruction { get; set; } : Instruction
- observedValueLabels { get; set; } : List<string>
- observedValues { get; set; } : ObservableNumericValueCollection

Methods

- setInstructionIndex(int index) : void

Instruction
Class

Fields

- _ui_index : int
- feedbackStatus : ObservableNumericValueStatus
- paramtable : JObject

Properties

- Label { get; set; } : string
- SerializedParamTable { get; set; } : string
- UI_Index { get; set; } : int

Methods

- Create(string instr) : JObject
- Create(string instr, JObject paramTable) : JObject
- Create(string instr, params object[] arguments) : JObject
- Instruction()
- Instruction(string instr)
- Instruction(string instr, JObject paramTable)
- Instruction(string instr, params object[] arguments)
- toJObject() : JObject

InstructionUIEntry
Struct

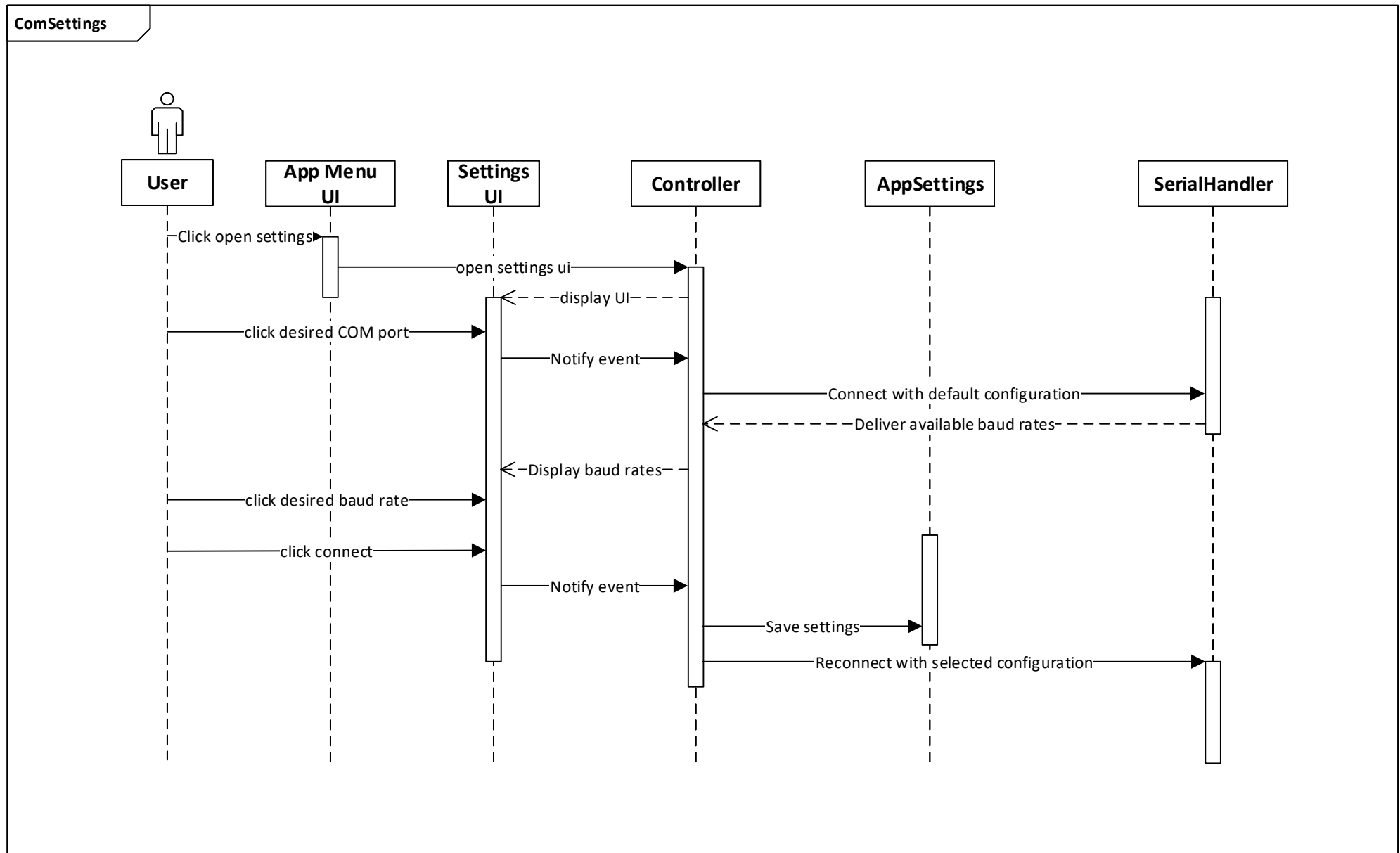
Fields

- label : string
- parameters : JObject

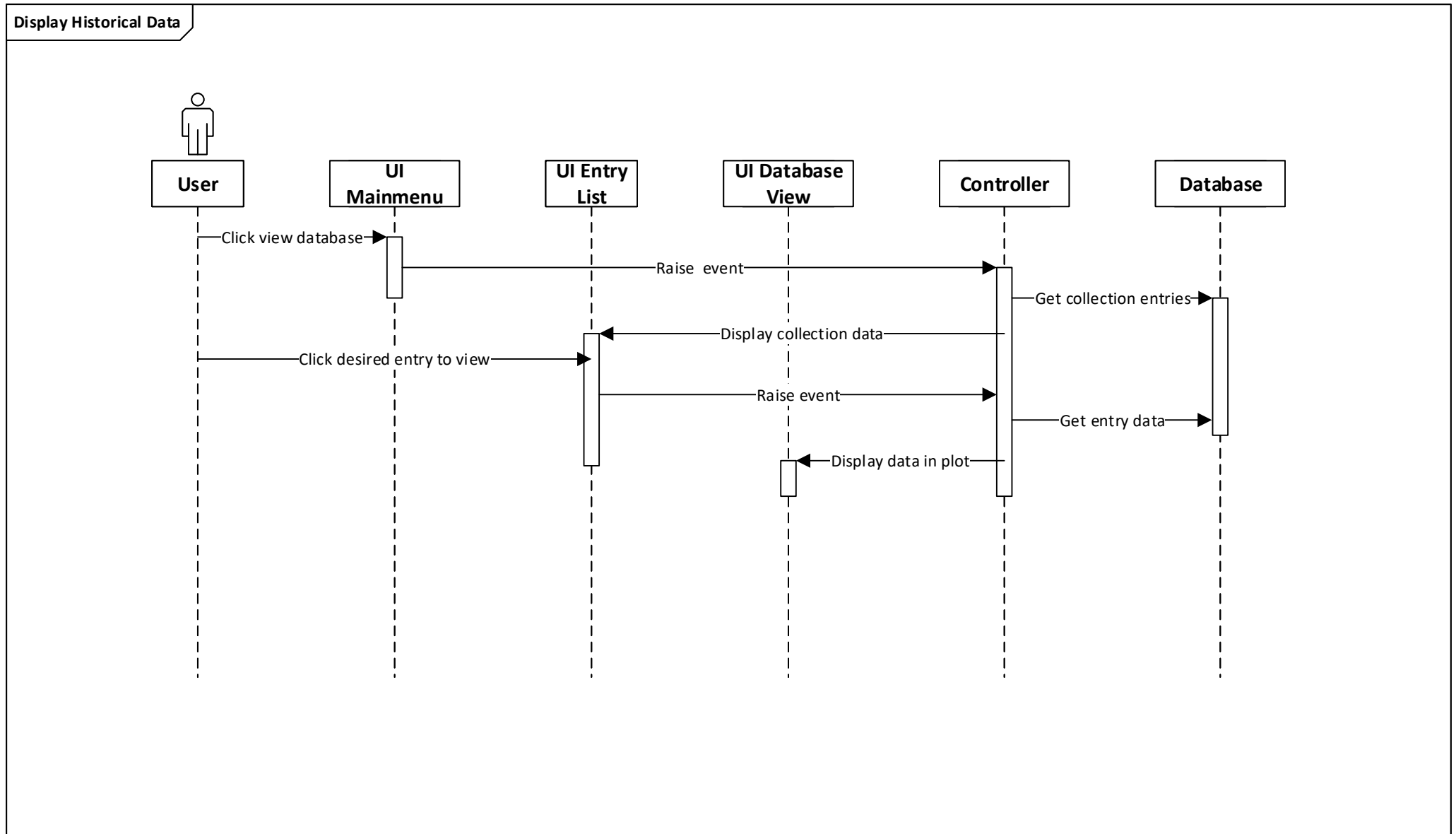
Methods

- ToString() : string

Alpha 2 Sequence: Com Settings

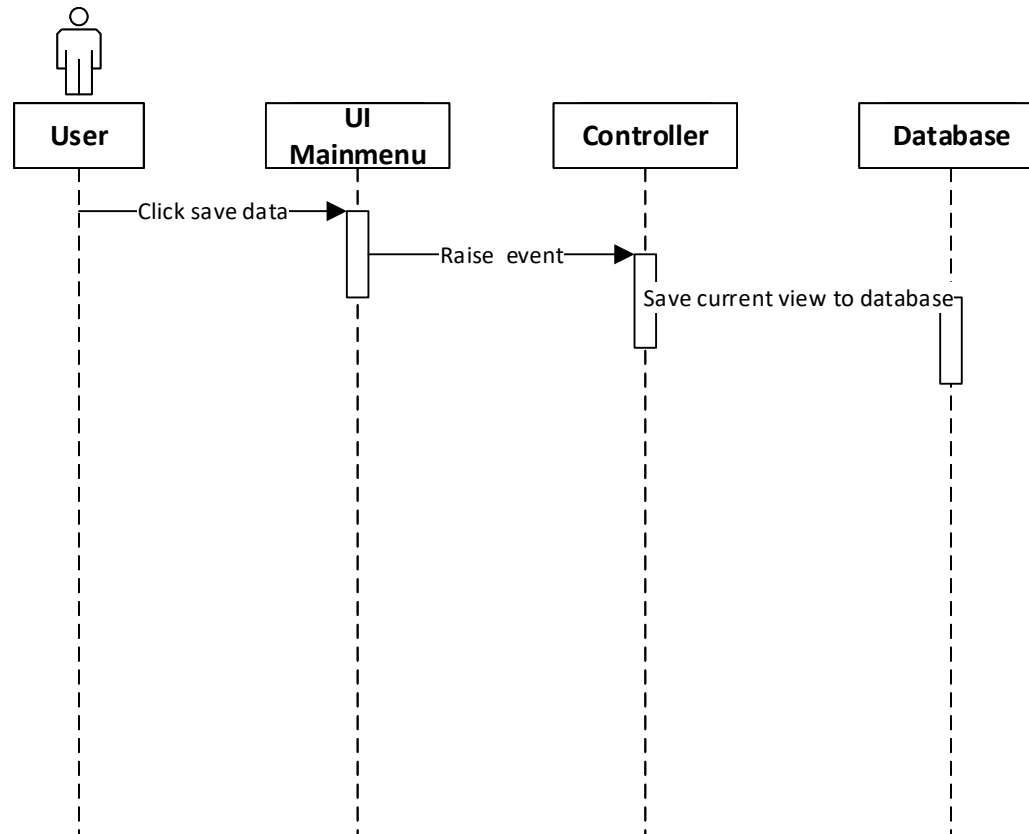


Alpha 2 Sequence: Display historical data

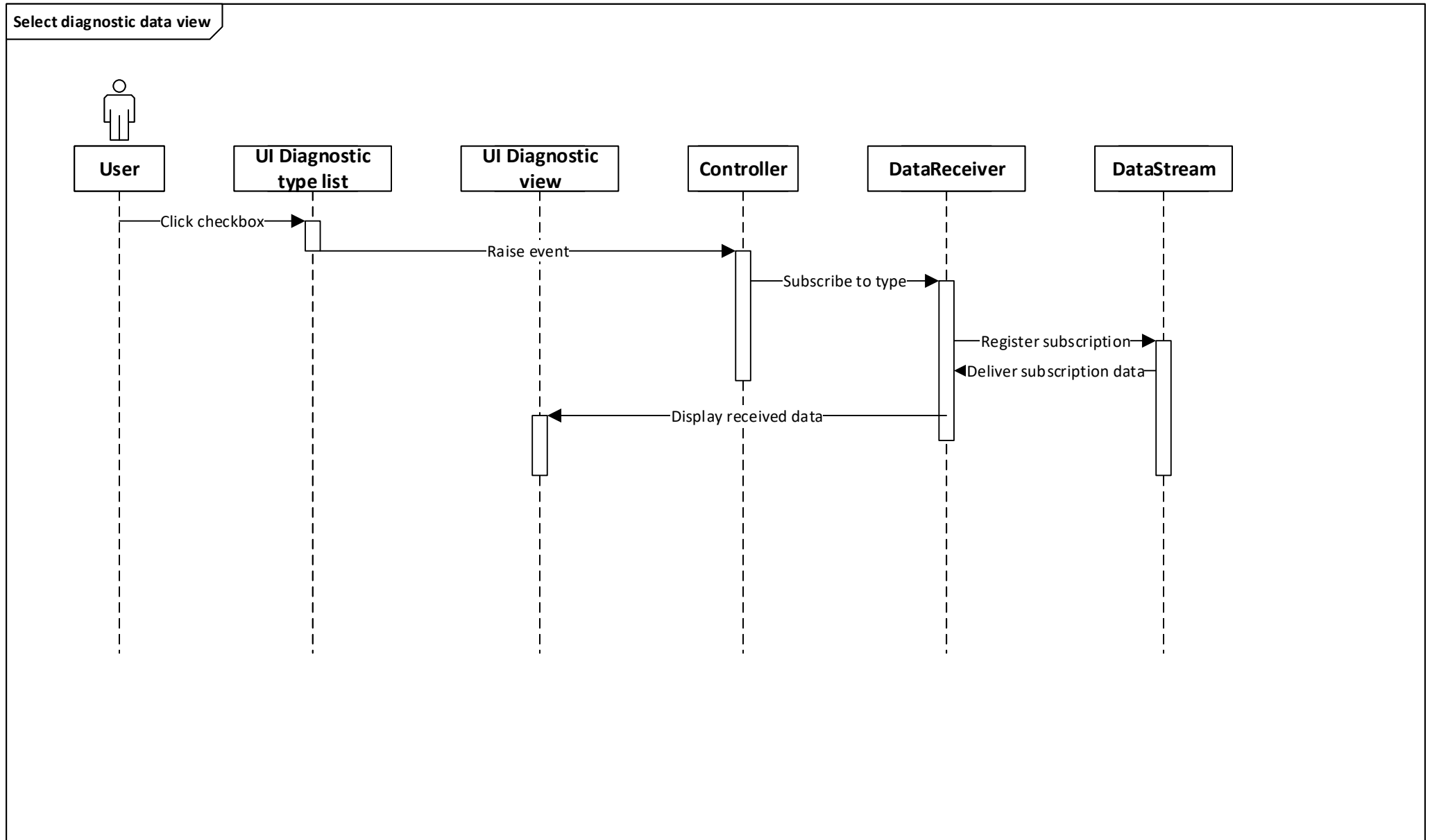


Alpha 3 Sequence: Save to database

Save to database

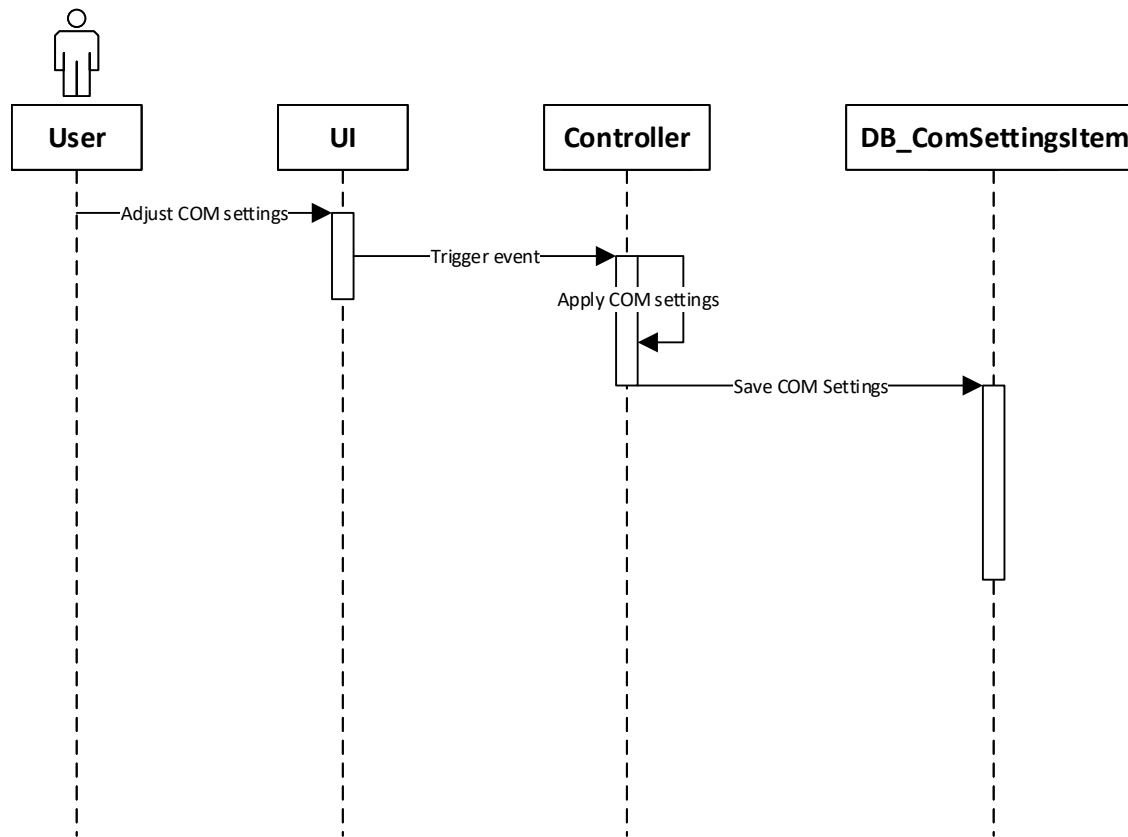


Alpha 3 Sequence: Select diagnostic view

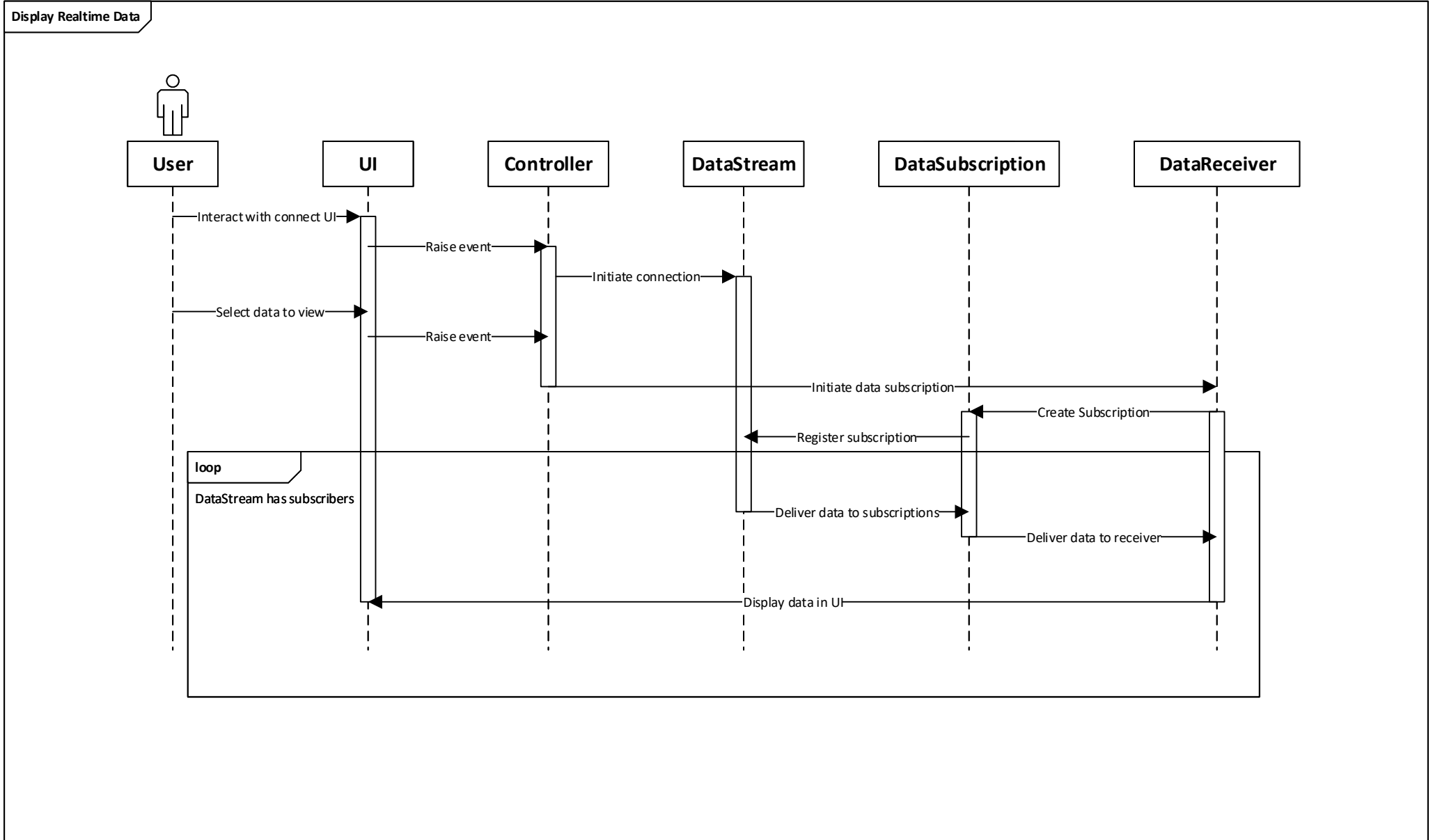


Alpha 3 Sequence: Save COM Settings

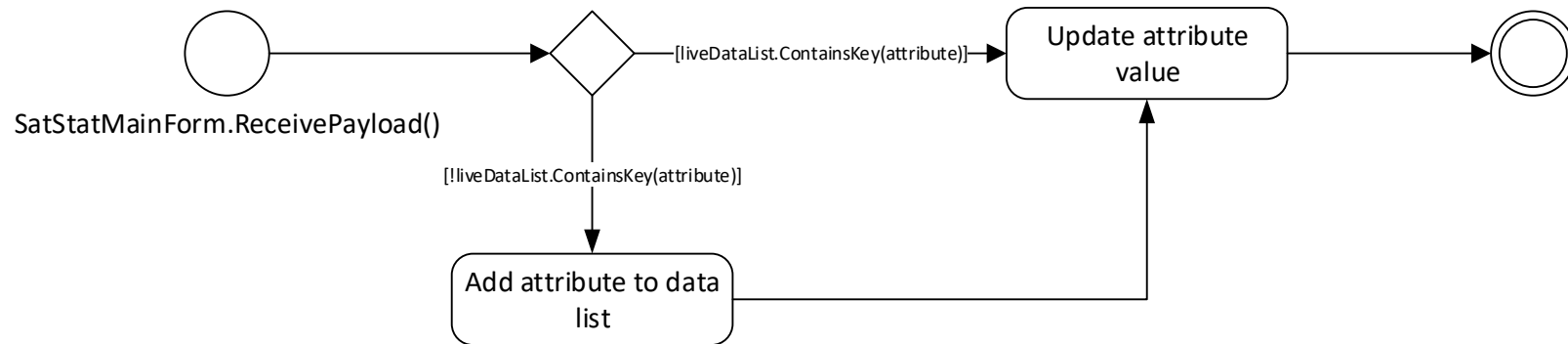
Remember COM settings



Alpha 3 Sequence: Display Realtime Data



Alpha 3 Activity: Add sensor attribute



Thomas Mundal 11.04.2019



Bachelor of Engineering

Project appendix

Winter semester 2019

DS SWL Alpha1 requirements

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Thomas Mundal and Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains requirement descriptions for Diagnostics System Software Layer Alpha 1

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

Diagnostics System - Software Layer version Alpha1

Target release	ALPHA1
Epic	USN5-78 - Software layer of Diagnostics System IN PROGRESS
Document status	V 1.0
Document owner	Thomas Mundal
Designer	Thomas Mundal
Tech lead	Thomas Mundal
Technical writers	Thomas Mundal
QA	Thomas Mundal

Objective

This document describes the requirements associated with the software layer (SWL) of the diagnostics system (DS). The main purpose of the SWL is to act as a human-machine-interface (HMI) between the hardware control unit and the mechanical components of the Solar Array Drive Mechanism (SADM) unit. Together with the hardware layer (HWL), the DS shall act as a calibration, diagnostics, and testing system for the mechanical component of the SADM unit.

Roadmap

ID	Task Name	Start	Finish	Duration	10 feb 2019							17 feb 2019							24 feb 2019							3 mar 2019						
					11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	1	2	3	4	5	6	7	8	9	10
1	Design	11.02.2019	13.02.2019	3d	█																											
2	Implementation	13.02.2019	16.02.2019	4d	█																											
3	Verification	15.02.2019	17.02.2019	3d	█																											
4	Validation	16.02.2019	17.02.2019	2d	█																											
5	Alpha 1	18.02.2019	18.02.2019	0d								◆																				
6	Design	19.02.2019	22.02.2019	4d								█																				
7	Implementation	21.02.2019	24.02.2019	4d								█																				
8	Verification	23.02.2019	24.02.2019	2d								█																				
9	Validation	24.02.2019	24.02.2019	1d								█																				
10	Alpha 2	25.02.2019	25.02.2019	0d															◆													
11	Design	26.02.2019	03.03.2019	6d															█													
12	Implementation	28.02.2019	06.03.2019	7d															█													
13	Verification	05.03.2019	10.03.2019	6d															█													
14	Validation	08.03.2019	10.03.2019	3d															█													
15	Alpha 3	11.03.2019	11.03.2019	0d																						◆						

Requirements

Requirement ID	Description	Importance	Version
R4-12A	The DS Software Layer should use JSON for input data	HIGH	ALPHA 1
Date	Discussions and decisions		

User Story

USN5-108 - As a software engineer, I need the diagnostic system to use a standardized way of exchanging data

DONE

Verification ID	Verification method	Status
UTV-3	Unit Test: Assert input format match	DISCARDED
UTV-5	Assert datatype and value as expected for: double	PASSED
UTV-8	Assert datatype and value as expected for string	PASSED
UTV-9	Assert datatype and value as expected for float	PASSED
UTV-10	Assert datatype and value as expected for int	PASSED
UTV-11	Assert datatype and value as expected for long	PASSED
UTV-12	Assert datatype and value as expected for JSONArray	PASSED
UTV-22	Assert correct values received when multiple subscribers subscribe on same stream to different keys	PASSED
UTV-23	Assert correct values received when multiple subscribers subscribe on same stream to same key	PASSED
UTV-25	Assert datatype and value as expected for JObject	PASSED

Requirement ID	Description	Importance	Version
R4-13A	The DS Software Layer should use JSON for output data	HIGH	ALPHA 1

Date	Discussion and decisions

User Story

USN5-108 - As a software engineer, I need the diagnostic system to use a standardized way of exchanging data

DONE

Verification ID	Verification method	Status
UTV-4	Unit Test: assert output format match	DISCARDED
UTV-13	Assert datatype and value as expected for: double	PASSED
UTV-14	Assert datatype and value as expected for string	PASSED
UTV-15	Assert datatype and value as expected for float	PASSED
UTV-16	Assert datatype and value as expected for int	PASSED
UTV-17	Assert datatype and value as expected for long	PASSED
UTV-18	Assert datatype and value as expected for generic list	DISCARDED
UTV-19	Assert datatype and value as expected for key value pair with generic value	DISCARDED
UTV-20	Assert datatype and value as expected for Hashtable	PASSED
UTV-21	Assert datatype and value as expected for JObject	PASSED

Requirement ID	Description	Importance	Version
R2-27	The DS Software Layer shall display diagnostics data in real time	HIGH	ALPHA 1

Date	Discussion and decisions

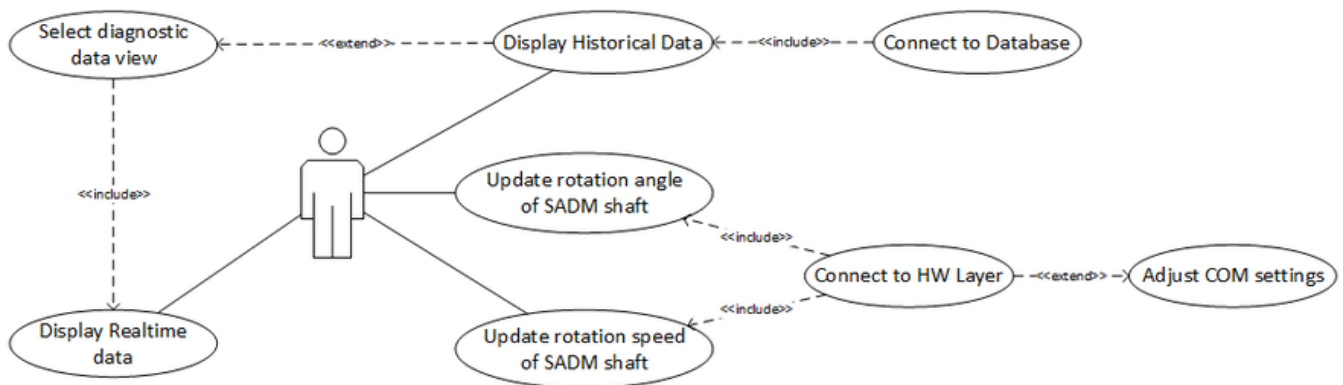
User Story

USN5-74 - As a test engineer, I need a software that can give me diagnostic data in real time
DONE

Verification ID	Verification method	Status
IV06	Ensure that data is displayed on screen	PASSED
AV-34	Compare timestamp from when data is sent to when data is displayed	PENDING

User interaction and design

Use case diagram



Class diagram for version Alpha1



Bachelor of Engineering

Project appendix

Winter semester 2019

DS SWL Alpha2 requirements

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Thomas Mundal
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains requirement descriptions for Diagnostics System Software Layer Alpha 2

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

Diagnostics System - Software Layer version Alpha2

Target release	ALPHA2
Epic	USN5-78 - Software layer of Diagnostics System IN PROGRESS
Document status	V 1.0
Document owner	Thomas Mundal
Designer	Thomas Mundal
Tech lead	Thomas Mundal
Technical writers	Thomas Mundal
QA	Thomas Mundal

Objective

This document describes the requirements associated with the software layer (SWL) of the diagnostics system (DS). The main purpose of the SWL is to act as a human-machine-interface (HMI) between the hardware control unit and the mechanical components of the Solar Array Drive Mechanism (SADM) unit. Together with the hardware layer (HWL), the DS shall act as a calibration, diagnostics, and testing system for the mechanical component of the SADM unit.

Roadmap

ID	Task Name	Start	Finish	Duration	10 feb 2019					17 feb 2019					24 feb 2019					3 mar 2019								
					11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	1	2	3	4	5	6
1	Design	11.02.2019	13.02.2019	3d	█																							
2	Implementation	13.02.2019	16.02.2019	4d	█																							
3	Verification	15.02.2019	17.02.2019	3d	█																							
4	Validation	16.02.2019	17.02.2019	2d	█																							
5	Alpha1	18.02.2019	18.02.2019	0d												◆												
6	Design	19.02.2019	22.02.2019	4d						█																		
7	Implementation	21.02.2019	01.03.2019	9d						█																		
8	Verification	23.02.2019	02.03.2019	8d						█																		
9	Validation	24.02.2019	02.03.2019	7d						█																		
10	Alpha2	03.03.2019	03.03.2019	0d												◆												
11	Design	01.03.2019	06.03.2019	6d											█													
12	Implementation	04.03.2019	10.03.2019	7d											█													
13	Verification	07.03.2019	10.03.2019	4d											█													
14	Validation	09.03.2019	10.03.2019	2d											█													
15	Alpha3	11.03.2019	11.03.2019	0d																							◆	

Requirements

Requirement ID	Description	Importance	Version
R4-14	DS Software layer GUI shall support changing underlying COM port settings	MEDIUM	ALPHA 2
Date	Discussion and decisions		

User Story

USN5-76 - As a test engineer, I need a simple way of configuring the communication between the diagnostics system software and hardware **DONE**

Verification ID	Verification method	Status
IV-07	Ensure that com settings are correct	PASSED

Requirement ID	Description	Importance	Version
R2-28	The DS Software Layer shall have a modular design for easy addition of diagnostic components	HIGH	ALPHA 2
Date	Discussion and decisions		

User Story

USN5-77 - As a software engineer, I need the diagnostic system to be easily expandable to allow for other diagnostics components **DONE**

Verification ID	Verification method	Status
IV-16	Code audit	PASSED

Requirement ID	Description	Importance	Version
R2-32	The DS Software layer shall be able to calibrate position and speed of the SADM drive shaft	HIGH	ALPHA 2

Relationships

Children	R4-20
Date	Discussion and decisions

User Story

USN5-131 - As a test engineer, I need the ability to calibrate position and speed of the SADM drive shaft so that I can set correct initial conditions for operation **DONE**

Verification ID	Verification method	Status
IV-13	Ensure that the SADM correctly responds to calibration input	PASSED

Requirement ID	Description	Importance	Version
R4-20	The DS Software Layer should provide an interface for adjusting calibration settings such as position and speed of the SADM motor	HIGH	ALPHA 2

Relationship

Parent R2-32

Date Discussion and decisions

User Story

USN5-131 - As a test engineer, I need the ability to calibrate position and speed of the SADM drive shaft so that I can set correct initial conditions for operation **DONE**

Verification ID	Verification method	Status
IV-14	Ensure that interface records and delivers the right configuration information	PASSED

Requirement ID	Description	Importance	Version
R4-25	Shall follow a unified communication protocol	HIGH	ALPHA 2

Relationship

Date Discussion and decisions

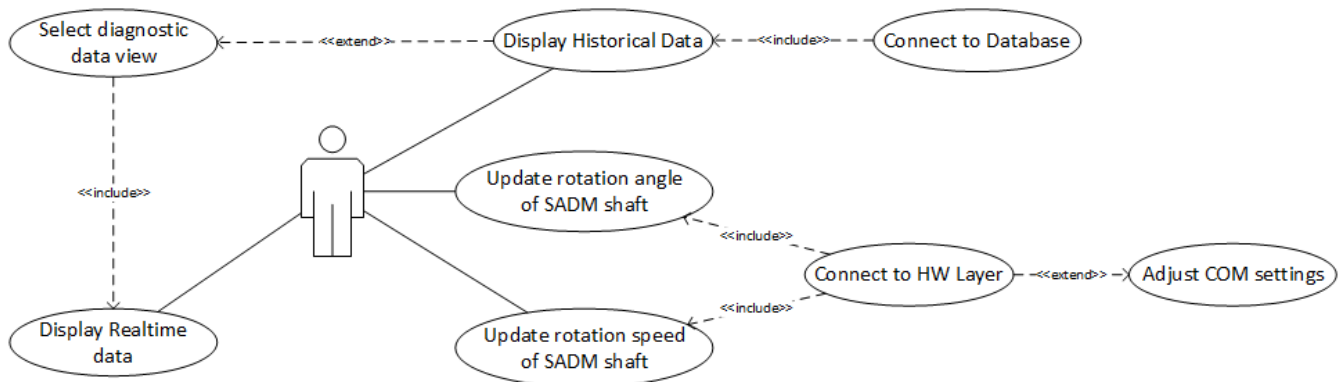
User Story

USN5-226 - As a computer engineer, I need to define a communication protocol so that both hardware- and software-layer can communicate correctly **DONE**

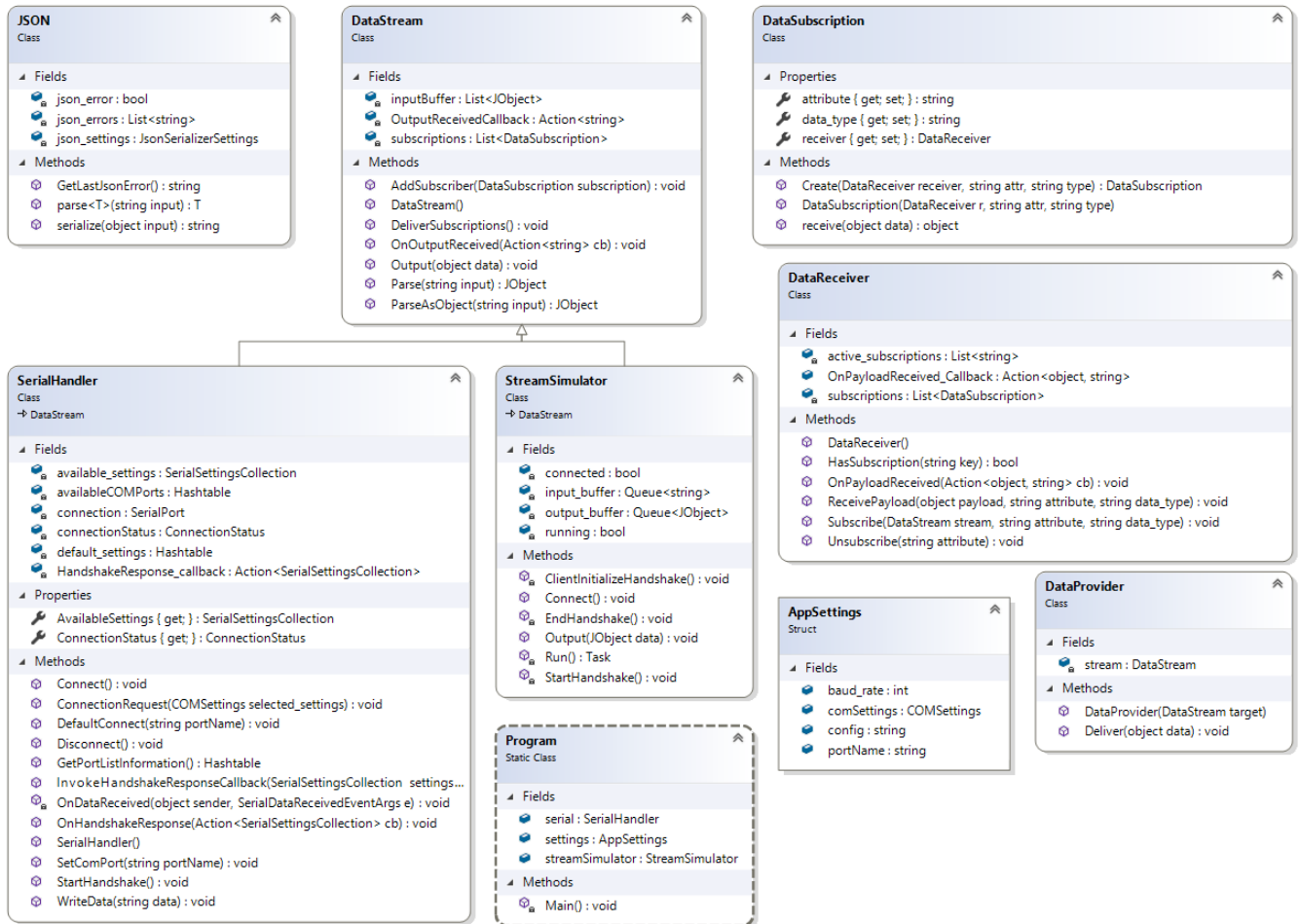
Verification ID	Verification method	Status
IV-17	Code audit	PASSED
TV-43	Functional test: System behaves as expected and follows defined protocol	PASSED

User interaction and design

Use case diagram



Class diagram for version Alpha1



Sequence diagram - Adjust COM settings and connect to HW layer



Bachelor of Engineering

Project appendix

Winter semester 2019

DS SWL Alpha3 requirements

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Thomas Mundal
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains requirement descriptions for Diagnostics System Software Layer Alpha 3

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

Diagnostics System - Software Layer version Alpha3

Target release	ALPHA3
Epic	USN5-78 - Software layer of Diagnostics System IN PROGRESS
Document status	DRAFT
Document owner	Thomas Mundal
Designer	Thomas Mundal
Tech lead	Thomas Mundal
Technical writers	Thomas Mundal
QA	Thomas Mundal

Objective

This document describes the requirements associated with the software layer (SWL) of the diagnostics system (DS). The main purpose of the SWL is to act as a human-machine-interface (HMI) between the hardware control unit and the mechanical components of the Solar Array Drive Mechanism (SADM) unit. Together with the hardware layer (HWL), the DS shall act as a calibration, diagnostics, and testing system for the mechanical component of the SADM unit.

Roadmap

ID	Task Name	Start	Finish	Duration	10 feb 2019											17 feb 2019					24 feb 2019					3 mar 2019				
					11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	1	2	3	4	5	6	7	8
1	Design	11.02.2019	13.02.2019	3d	█																									
2	Implementation	13.02.2019	16.02.2019	4d	█																									
3	Verification	15.02.2019	17.02.2019	3d	█																									
4	Validation	16.02.2019	17.02.2019	2d	█																									
5	Alpha1	18.02.2019	18.02.2019	0d	◆																									
6	Design	19.02.2019	22.02.2019	4d	█																									
7	Implementation	21.02.2019	01.03.2019	9d	█																									
8	Verification	23.02.2019	02.03.2019	8d	█																									
9	Validation	24.02.2019	02.03.2019	7d	█																									
10	Alpha2	03.03.2019	03.03.2019	0d	◆																									
11	Design	01.03.2019	06.03.2019	6d	█																									
12	Implementation	04.03.2019	10.03.2019	7d	█																									
13	Verification	07.03.2019	10.03.2019	4d	█																									
14	Validation	09.03.2019	10.03.2019	2d	█																									
15	Alpha3	11.03.2019	11.03.2019	0d	◆																									

Requirements

Requirement ID	Description	Importance	Version
R4-15	DS Software layer GUI shall support selection of diagnostic data output view	MEDIUM	ALPHA 3
Date	Discussion and decisions		

User Story

USN5-87 - As a test engineer, I need an easy and intuitive way of choosing which diagnostics data I want to look at
DONE

Verification ID	Verification method	Status
TV-37	Functional test: View should display correctly after changed to desired output type	PASSED

Requirement ID	Description	Importance	Version
R4-16	DS Software layer GUI shall support displaying historical data	MEDIUM	ALPHA 3
Date	Discussion and decisions		

User Story

USN5-75 - As a test engineer, I need a software that can display historical diagnostic data
DONE

Verification ID	Verification method	Status
TV-38	Functional test: Output should display historical data according to selected time settings	PASSED

Requirement ID	Description	Importance	Version
R2-29	The DS Software Layer shall save diagnostic data to a database	MEDIUM	ALPHA 3
Date	Discussion and decisions		

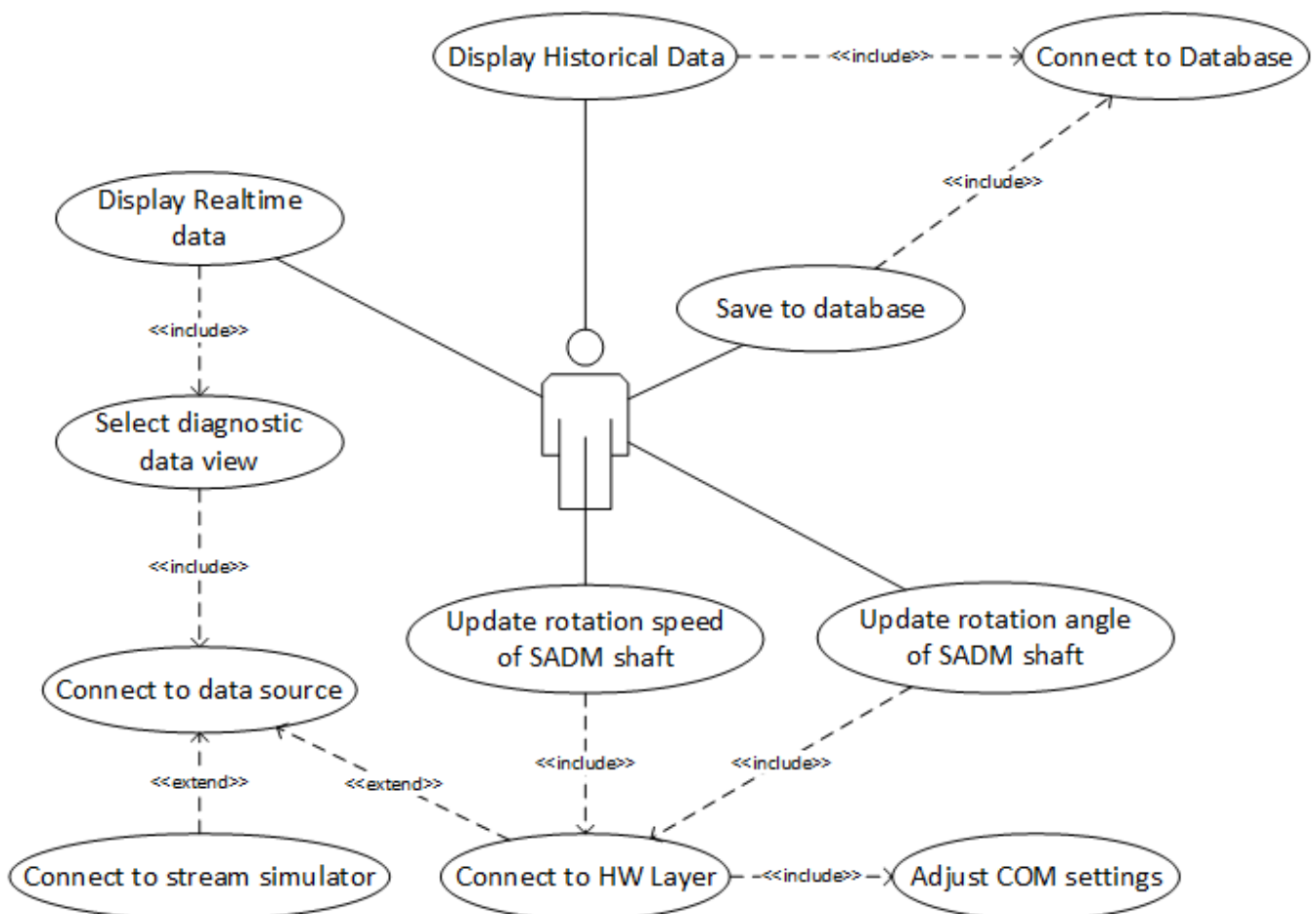
User Story

USN5-118 - As a software engineer, I need a standardized interface for communicating with a database so that the system can save historical data **DONE**

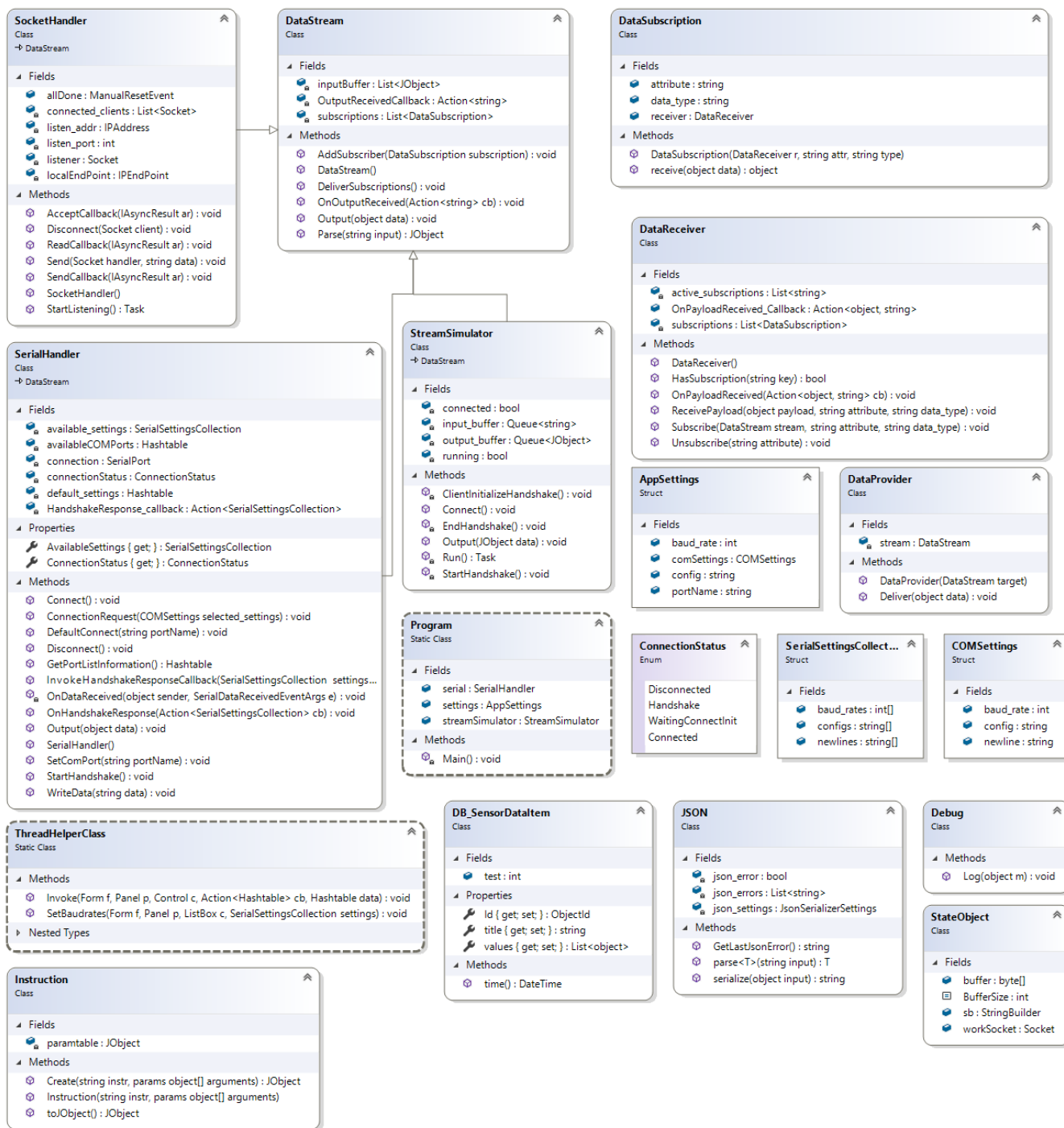
Verification ID	Verification method	Status
UTV-1	Unit Test: Assert database query success	DISCARDED
UTV-2	Unit Test: Assert value saved equal to value input	DISCARDED
TV-49	Functional test: Data is saved to database and is readable as values for data plot	PASSED

User interaction and design

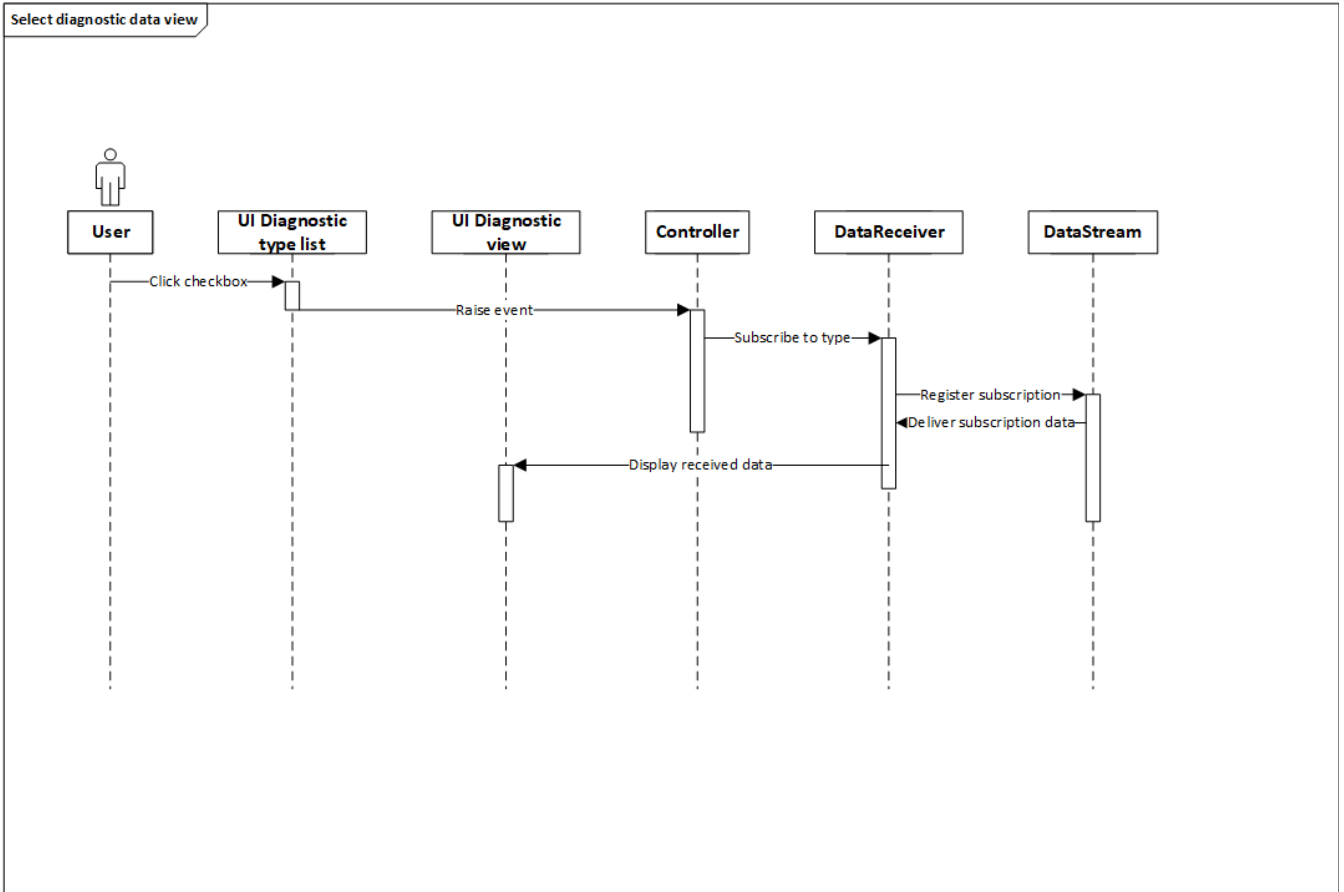
Use case diagram Alpha3

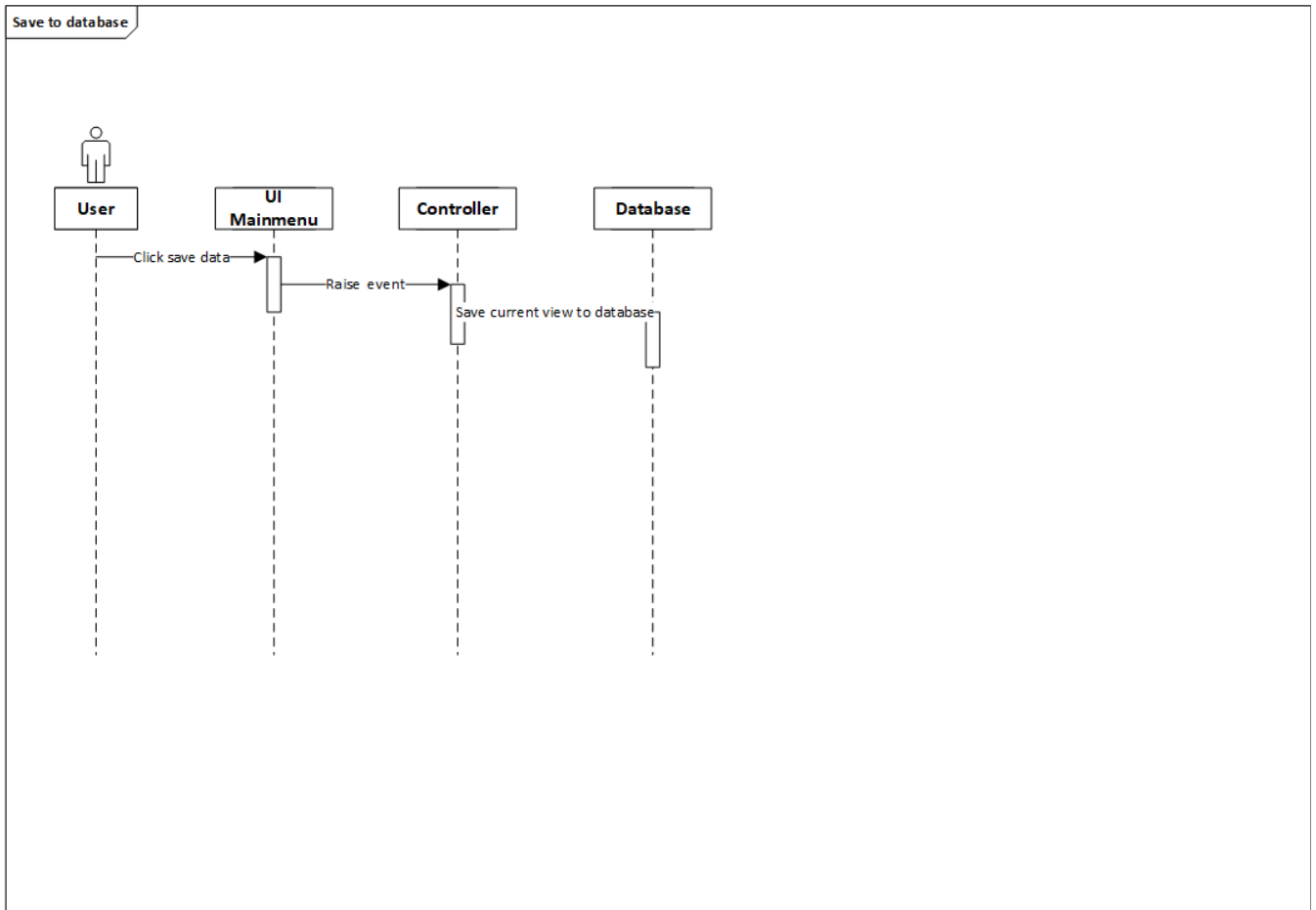


Class diagram Alpha3



Sequence diagrams







Bachelor of Engineering

Project appendix

Winter semester 2019

DS SWL Beta1 requirements

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Thomas Mundal
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains requirement descriptions for Diagnostics System Software Layer Beta 1

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

Diagnostics System - Software Layer version Beta1

Target release	BETA1
Epic	USN5-78 - Software layer of Diagnostics System IN PROGRESS
Document status	DRAFT
Document owner	Thomas Mundal
Designer	Thomas Mundal
Tech lead	Thomas Mundal
Technical writers	Thomas Mundal
QA	Thomas Mundal

Objective

This document describes the requirements associated with the software layer (SWL) of the diagnostics system (DS). The main purpose of the SWL is to act as a human-machine-interface (HMI) between the hardware control unit and the mechanical components of the Solar Array Drive Mechanism (SADM) unit. Together with the hardware layer (HWL), the DS shall act as a calibration, diagnostics, and testing system for the mechanical component of the SADM unit.

Roadmap

ID	Task Name	Start	Finish	Duration	04 mar 2019							31 mar 2019							7 apr 2019							14 apr 2019						
					25	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15						
1	Save OOM (R2-33)	25.03.2019	25.03.2019	1d	█																											
2	Plot optimizations (R2-27, R4-15)	25.03.2019	30.03.2019	6d	██████████																											
3	UI diag settings (R2-34)	28.03.2019	31.03.2019	4d	██████																											
4	UI live values status indicator (R2-34)	29.03.2019	31.03.2019	3d	█████																											
5	Code structure review and optimization (R2-28)	01.04.2019	07.04.2019	7d	██████████																											
6	UI optimization and design overhaul	01.04.2019	07.04.2019	7d	██████████																											
7	Automated tests (R2-34, R2-35)	08.04.2019	14.04.2019	7d	██████████																											
8	UI diag settings	08.04.2019	09.04.2019	2d	███																											
9	Diag settings template saving	09.04.2019	11.04.2019	3d	████																											
10	Run tests from template	08.04.2019	14.04.2019	7d	██████████																											
11	Beta1	15.04.2019	15.04.2019	0d	◆																											
12																																

Revised roadmap

ID	Task Name	Start	Finish	Duration	31 mar 2019							7 apr 2019							14 apr 2019							21 apr 2019						
					31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22					
1	Save COM (R2-33)	31.03.2019	31.03.2019	1d	[Gantt bar]																											
2	Plot optimizations (R2-27, R4-15)	31.03.2019	05.04.2019	6d	[Gantt bar]																											
3	UI diag settings (R2-34)	02.04.2019	05.04.2019	4d	[Gantt bar]																											
4	UI live values status indicator (R2-34)	06.04.2019	08.04.2019	3d	[Gantt bar]																											
5	Diag settings template saving	09.04.2019	14.04.2019	6d	[Gantt bar]																											
6	Beta1	15.04.2019	15.04.2019	0d	[Milestone diamond]																											
7	Run tests from template	15.04.2019	21.04.2019	7d	[Gantt bar]																											
8	Automated tests (R2-34, R2-35)	15.04.2019	21.04.2019	7d	[Gantt bar]																											
9	Beta2	22.04.2019	22.04.2019	0d	[Milestone diamond]																											
10	UI optimization and design overhaul	22.04.2019	22.04.2019	0d	[Milestone diamond]																											
11	Code structure review and optimization (R2-28)	22.04.2019	22.04.2019	0d	[Milestone diamond]																											

Requirements

Requirement ID	Description	Importance	Version
R2-33	The DS Software layer should be able to remember previous COM settings and try to establish a connection to HW layer with these	MEDIUM	BETA 1
Relationship			
Date			
Discussion and decisions			
User Story			
USN5-146 - As a test engineer, I would like to not have to adjust COM settings every time I have to connect to the HW layer so that the software is easier to use DONE			
Verification ID	Verification method	Status	
TV-42	Unit Test: Assert writing to config file success	PASSED	
TV-43	Unit Test: Assert written config data equal to input	PASSED	
IV-15	Verify that correct config data is written to config file, and no other config data is overwritten	PASSED	

Requirement ID	Description	Importance	Version
R2-27	The DS Software Layer shall display diagnostics data in real time	HIGH	BETA 1
Relationship			
Date	Discussion and decisions		
User Story			
<p>USN5-275 - As a test engineer, I would like to see the current value of a sensor input so that I can keep an eye on the current status of the system DONE</p>			
Verification ID	Verification method	Status	
TV-50	Inspect live data values from live data table in the user interface	PASSED	

Requirement ID	Description	Importance	Version
R2-34	DS SW should be able to configure desired maximum and minimum limits for operational sensor values to display status indicator on live input values during diagnostics run. Values outside accepted values should indicate deviation		BETA 1
Relationship			
Children	R2-34.01, R2-34.02, R4-22, R4-23		
Date	Discussion and decisions		
11.04.2019	<p>In this version, this functionality will be implemented as a DataGridView control where the user can edit a numeric value for maximum and minimum values. Each row in the DataGridView has three columns; "Data attribute", "Minimum" and "Maximum". A new row is added to this grid when a data attribute is subscribed to using the subscription view.</p>		
User Story			

USN5-279 - As a test engineer, I need the ability to adjust accepted values for sensor readings to be able to verify that the system is operating as intended **DONE**

Verification ID	Verification method	Status
UTV-26	Verify functionality for ObservableNumericValue and ObservableNumericValueCollection	PASSED

Requirement ID	Description	Importance	Version
R2-34.01	DS SW should provide functionality to save diagnostics settings to a template		BETA 1

Relationship

Parent R2-34

Date Discussion and decisions

User Story

USN5-281 - As a test engineer, I want the ability to save accepted sensor value ranges to a template, so that I can load the same settings for testing at any point in time **DONE**

Verification ID	Verification method	Status
TV-51.A	Functional test: Entry appears in UI after saving. Data is displayed in parameter control UI after loading. Correct values for minimum and maximum values appear after loading.	PASSED

Requirement ID	Description	Importance	Version
R2-34.02	DS SW should be able to load and apply settings from a diagnostics settings template		BETA 1

Relationship

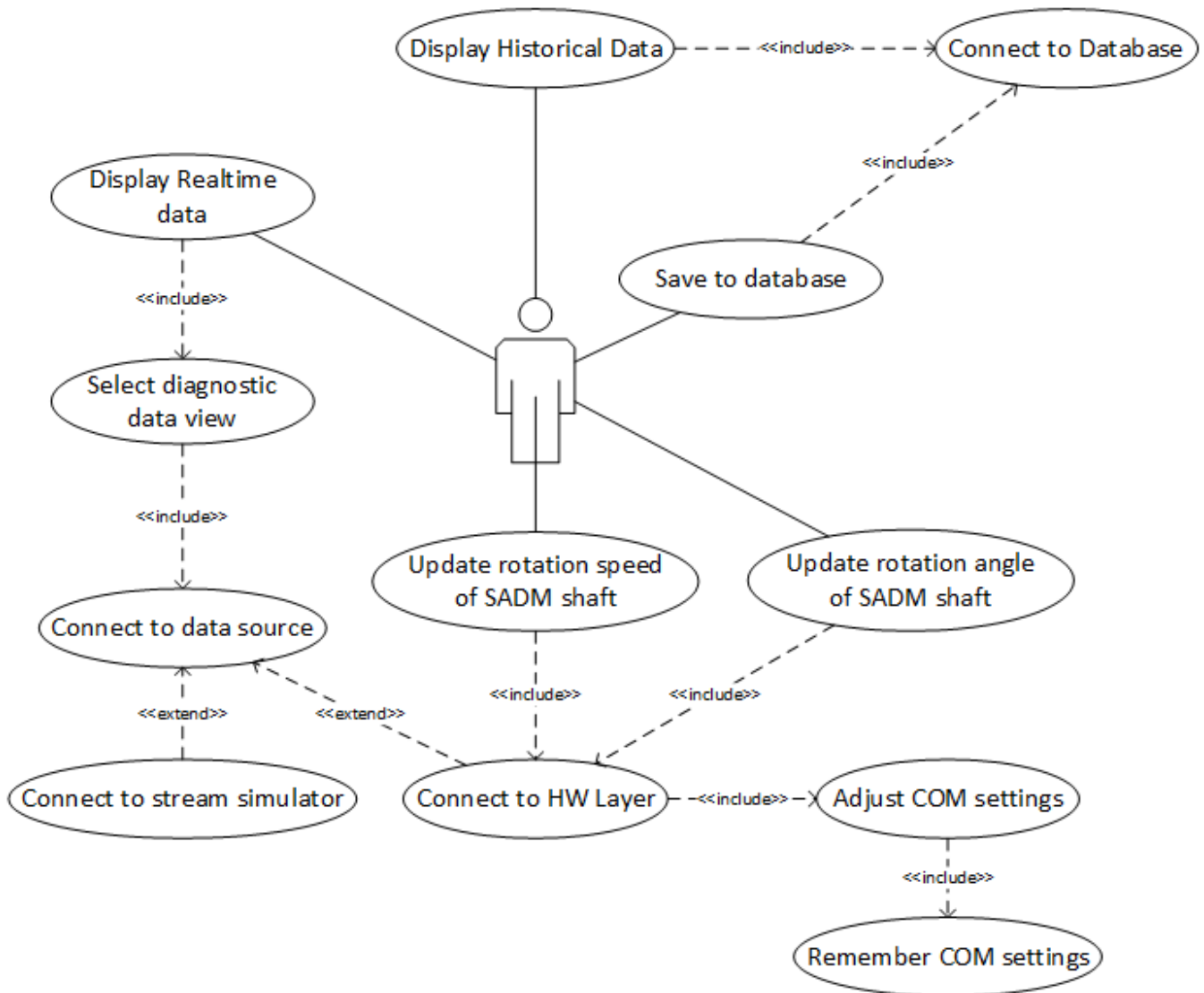
Parent R2-34

Date Discussion and decisions

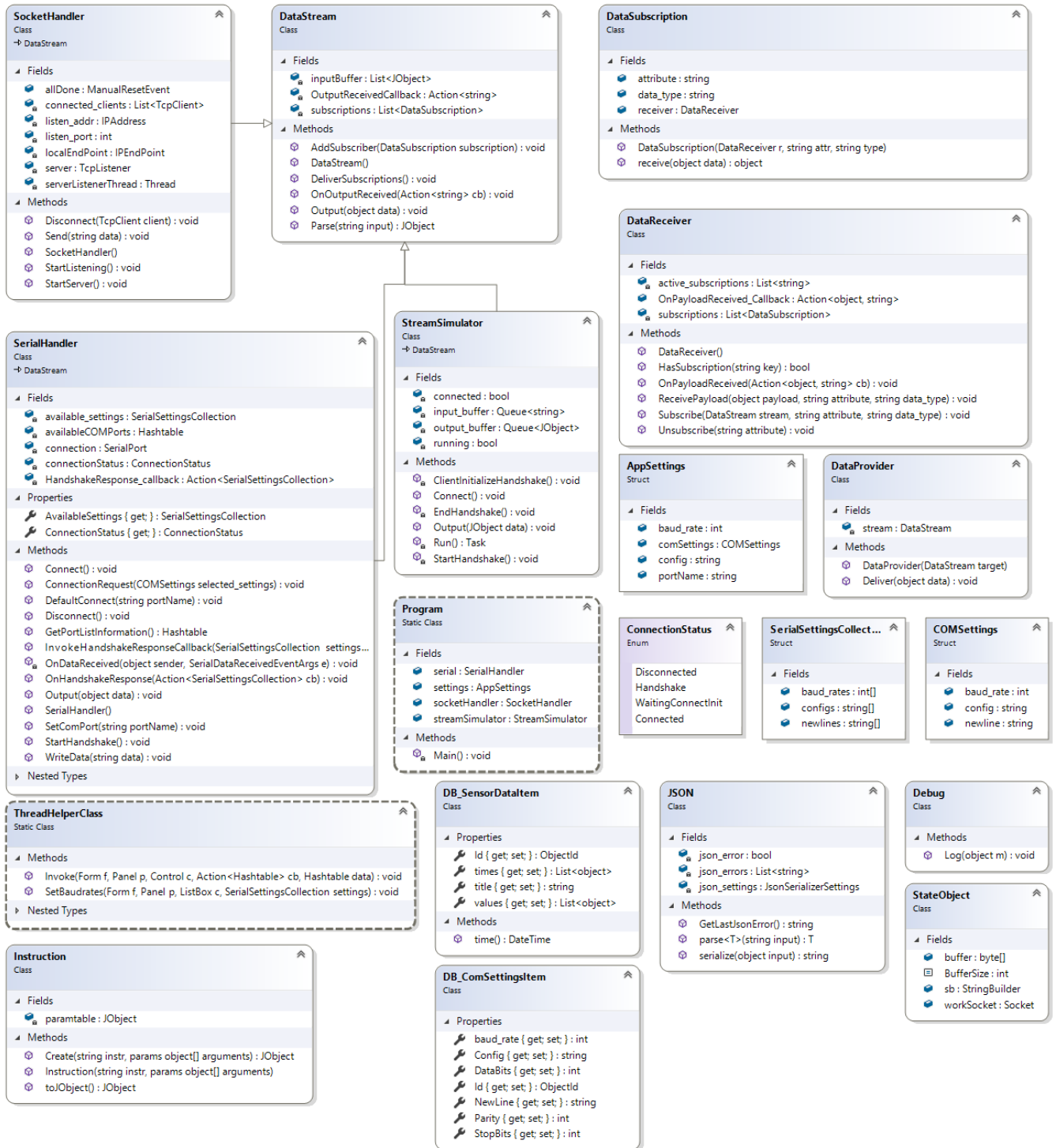
User Story		
Verification ID	Verification method	Status
TV-51.B	Functional test: Entry appears in UI after saving. Data is displayed in parameter control UI after loading. Correct values for minimum and maximum values appear after loading.	PASSED

User interaction and design

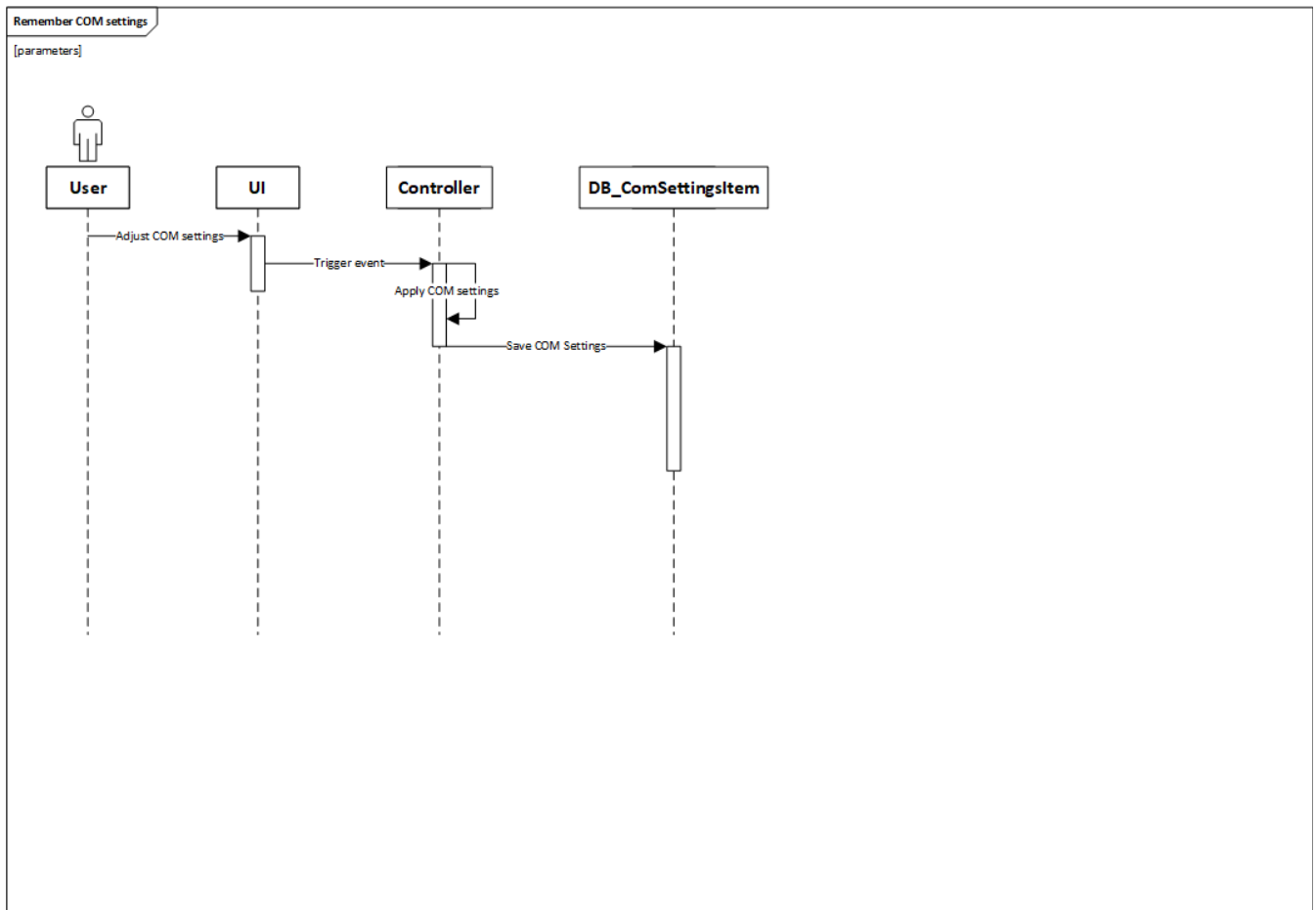
Use case diagram Beta1

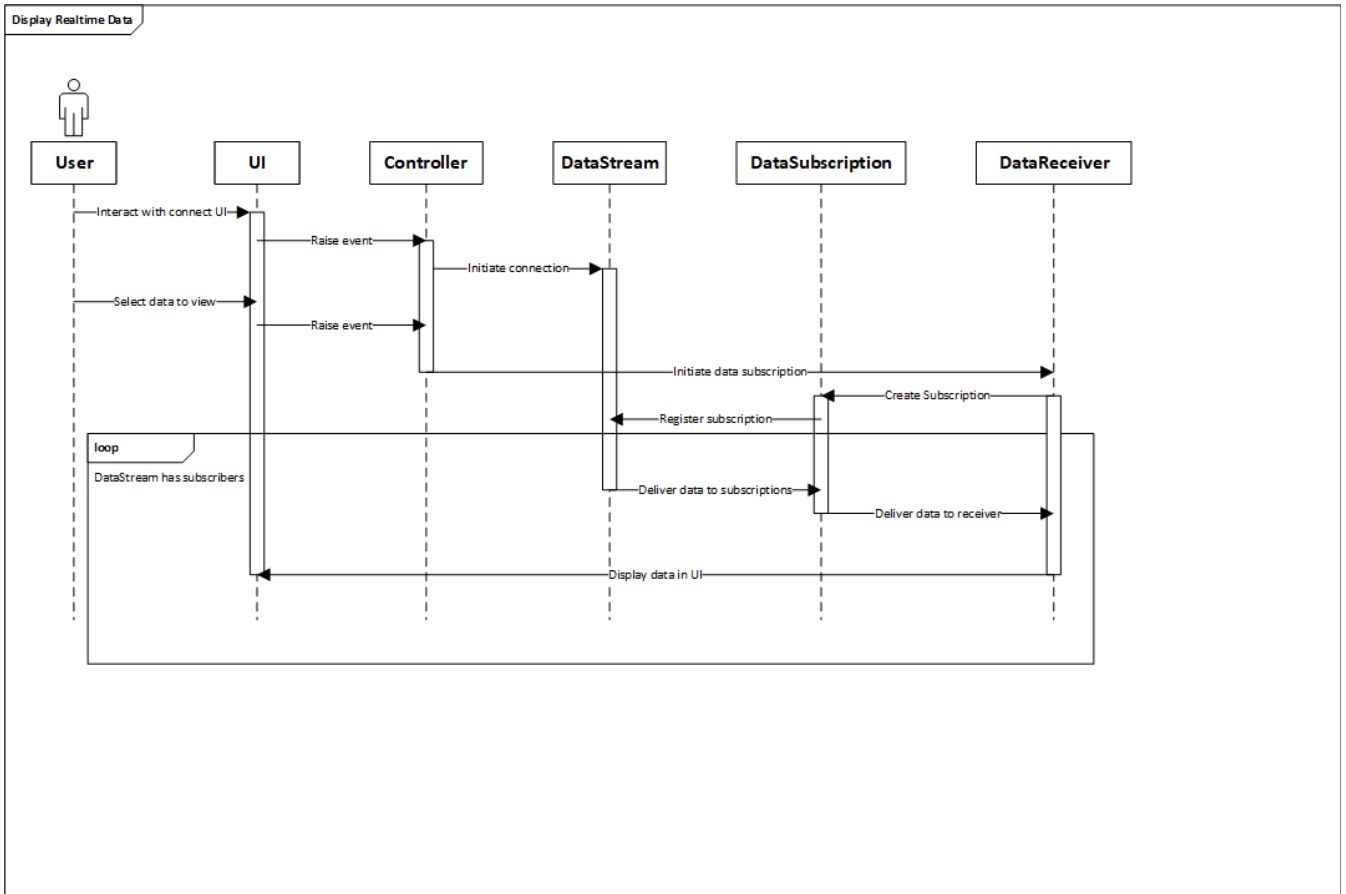


Class diagram Beta1

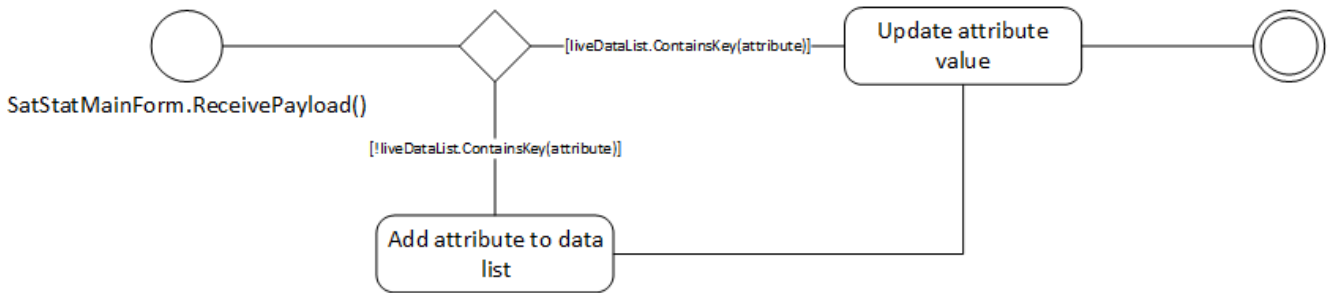


Sequence diagrams





Activity Diagram: Add control input row to parameter control input



Thomas Mundal 11.04.2019

Open Questions

Question	Answer	Date Answered

Out of Scope

The plotting library was changed from LiveCharts to OxyPlot. OxyPlot proved to be more optimized to handle data coming in real time. When using LiveCharts there was observed significant slowdowns the more data points were added to the plot, and the slowdown started in a very early stage only after a few hundred data points.



Bachelor of Engineering

Project appendix

Winter semester 2019

DS SWL Beta2 requirements

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Thomas Mundal
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains requirement descriptions for Diagnostics System Software Layer Beta 2

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

Diagnostics System - Software Layer version Beta2

Target release	BETA2
Epic	USN5-78 - Software layer of Diagnostics System IN PROGRESS
Document status	DRAFT
Document owner	Thomas Mundal
Designer	Thomas Mundal
Tech lead	Thomas Mundal
Technical writers	Thomas Mundal
QA	Thomas Mundal

Objective

This document describes the requirements associated with the software layer (SWL) of the diagnostics system (DS). The main purpose of the SWL is to act as a human-machine-interface (HMI) between the hardware control unit and the mechanical components of the Solar Array Drive Mechanism (SADM) unit. Together with the hardware layer (HWL), the DS shall act as a calibration, diagnostics, and testing system for the mechanical component of the SADM unit.

Roadmap

ID	Task Name	Start	Finish	Duration	31 mar 2019							7 apr 2019							14 apr 2019							21 apr 2019						
					31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22					
1	Save COM (R2-33)	31.03.2019	31.03.2019	1d	█																											
2	Plot optimizations (R2-27, R4-15)	31.03.2019	05.04.2019	6d	██████████																											
3	UI diag settings (R2-34)	02.04.2019	05.04.2019	4d	██████																											
4	UI live values status indicator (R2-34)	06.04.2019	08.04.2019	3d	████																											
5	Diag settings template saving	09.04.2019	14.04.2019	6d	██████████																											
6	Beta1	15.04.2019	15.04.2019	0d	◆																											
7	Run tests from template	15.04.2019	21.04.2019	7d	██████████																											
8	Automated tests (R2-34, R2-35)	15.04.2019	21.04.2019	7d	██████████																											
9	Beta2	22.04.2019	22.04.2019	0d	◆																											
10	UI optimization and design overhaul	22.04.2019	22.04.2019	0d	◆																											
11	Code structure review and optimization (R2-28)	22.04.2019	22.04.2019	0d	◆																											

Requirements

Requirement ID	Description	Importance	Version
R2-35	Should be able to run acceptance test on applied test parameters and generate test report		BETA 1
Relationship			
Children	R2-35.01, R2-35.02, R4-24		
Date	Discussion and decisions		
User Story			
USN5-304 - As a test engineer I want the ability to load test-run settings from a template and initiate a test run based on loaded or defined parameters DONE			
Verification ID	Verification method	Status	
TV-53	Functional test: A complete test run should run based on configured instructions and give feedback based on configured test parameters	PASSED	

Requirement ID	Description	Importance	Version
R2-35.01	Should be able to define acceptance test parameters		BETA 1
Relationship			
Parent	R2-35		
Date	Discussion and decisions		
User Story			
USN5-302 - As a test engineer I want the ability to define behaviour and parameters to measure during a test run for the SADM DONE			
Verification ID	Verification method	Status	

TV-54	Functional Test: Parameters can be provided through a UI and the parameters can be used to observe whether or not a value is inside their defined bounds	PASSED
-------	--	---------------

Requirement ID	Description	Importance	Version
R2-35.02	Should provide functionality for saving acceptance test parameters to an acceptance test template		BETA 1

Relationship

Parent R2-35

Date Discussion and decisions

User Story

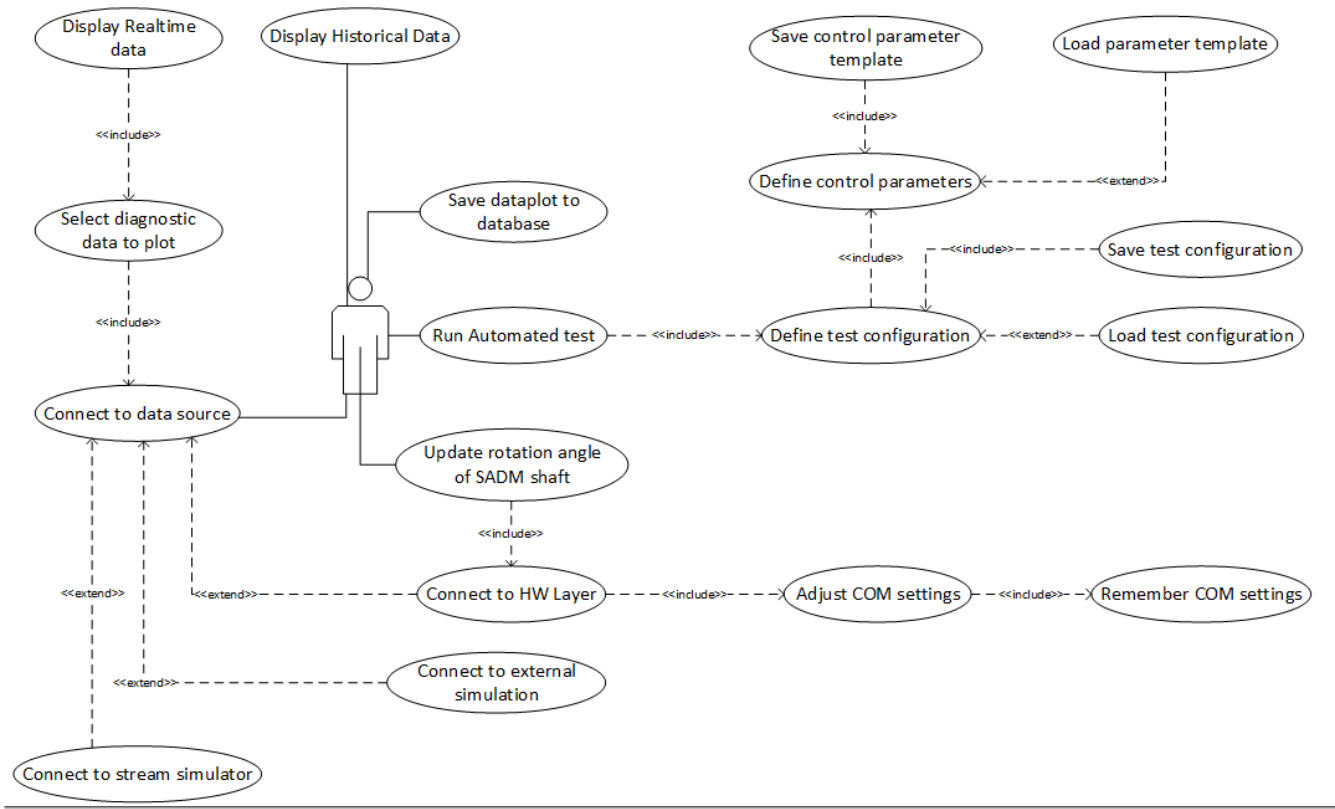
USN5-303 - As a test engineer I want the ability to save a defined test-run as a template so that I can load the same settings for a future test run **DONE**

Verification ID	Verification method	Status
TV-55	Functional test: The parameters can be saved to the database, and are loaded in the state that they was saved	PASSED

User interaction and design

Use case diagram Beta2

Beta 2 UseCase diagram



Class diagram Beta2

TestConfiguration
Class

- Fields
 - currentInstructionEntry : InstructionEntry
 - instructionEntryQueue : Queue<InstructionEntry>
 - internalReceiver : DataReceiver
 - is_running : bool
 - onQueueAbortedCallback : Action
 - onQueueAdvanceCallback : Action<InstructionEntry>
 - onQueueCompleteCallback : Action<InstructionEntry>
 - queue_position : int
- Properties
 - Description { get; set; } : string
 - Id { get; set; } : ObjectId
 - instructionEntries { get; set; } : List<InstructionEntry>
 - IsRunning { get; } : bool
 - Name { get; set; } : string
 - ParameterControlTemplate { get; set; } : ParameterControlTemplate
- Methods
 - Abort() : void
 - AddInstructionEntry(Instruction instr, List<string> observedLabels, [int uindex = -1]) : InstructionEntry
 - AddInstructionEntry(InstructionEntry entry) : void
 - HasInstructionEntries() : bool
 - InstructionEntryIndex(InstructionEntry entry) : int
 - loadParameterControlTemplate(ParameterControlTemplate template, DataStream stream) : void
 - OnObservedValueChange(IObservableNumericValue val) : void
 - OnPayloadReceive(object payload) : void
 - OnQueueAbort(Action callback) : void
 - OnQueueAdvance(Action<InstructionEntry> callback) : void
 - OnQueueComplete(Action<InstructionEntry> callback) : void
 - RemoveInstructionEntry(Instruction instr) : void
 - RemoveInstructionEntry(InstructionEntry entry) : void
 - Run(DataStream stream) : void
 - RunQueuedInstruction(DataStream stream) : void
 - Save([Action<TestConfiguration> cb = null]) : void
 - setParameterControlTemplate(ParameterControlTemplate template) : void
 - TestConfiguration()
 - ToString() : string

InstructionEntry
Class

- Properties
 - instruction { get; set; } : Instruction
 - observedValueLabels { get; set; } : List<string>
 - observedValues { get; set; } : ObservableNumericValueCollection
- Methods
 - setInstructionIndex(int index) : void

Instruction
Class

- Fields
 - _ui_index : int
 - feedbackStatus : ObservableNumericValueStatus
 - paramtable : JObject
- Properties
 - Label { get; set; } : string
 - SerializedParamTable { get; set; } : string
 - UI_Index { get; set; } : int
- Methods
 - Create(string instr) : JObject
 - Create(string instr, JObject paramTable) : JObject
 - Create(string instr, params object[] arguments) : JObject
 - Instruction()
 - Instruction(string instr)
 - Instruction(string instr, JObject paramTable)
 - Instruction(string instr, params object[] arguments)
 - toObject() : JObject

InstructionUIEntry
Struct

- Fields
 - label : string
 - parameters : JObject
- Methods
 - ToString() : string

Sequence diagrams

Open Questions

Question	Answer	Date Answered

Out of Scope



Bachelor of Engineering

Project appendix

Winter semester 2019

Verification report: VR-SWL-A1

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Thomas Mundal
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains verification report related to Diagnostics System Software Layer Alpha 1

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

VR-SWL-A1

Date performed	14.02.2019
Performed by	Thomas Mundal
Tests performed (17)	UTV-5, UTV-8, UTV-9, UTV-10, UTV-11, UTV-12, UTV-13, UTV-14, UTV-15, UTV-16, UTV-17, UTV-20, UTV-21, UTV-22, UTV-23, IV-06
Tests discarded (4)	UTV-3, UTV-4, UTV-18, UTV-19, AV-34
System version	Alpha1

Discarded tests

During testing, and test driven development, some decisions was made regarding operation of the program. Some of the tests were discarded as a result of these decisions, since the operation of the program no longer relies on these tests passing. Decisions were made in part because in order to pass the tests there was a need for implementing unnecessary and redundant code that would incorporate similar behaviour that already exists in the libraries used. Using the correct data types that is meant to be used in conjunction with parsing and serializing JSON data is the correct way to solve the goal these discarded tests aimed to verify.

Inspection IV-06

Through inspection it was concluded that data is delivered in real time, in the sense that real time means "fast enough".

Analysis AV-34

The system does not at this time support sending time information from the hardware layer, resulting in the inability to perform mathematical analysis for this verification. The verification is therefore discarded and succeeded by verification IV-06.

Unit Test results dump

Group Name: SatStatTests

Group By: Hierarchy

Group Full Name: SatStatTests

Duration: 0:00:00,4115396

0 test(s) failed
0 test(s) skipped
16 test(s) passed

Result1 Name: DeliverDouble
Result1 Outcome: Passed
Result1 Duration: 0:00:00,0008839
Result1 StackTrace:
Result1 Message:
Result1 StandardOutput: 1.0
Result1 StandardError:

Result2 Name: DeliverFloat
Result2 Outcome: Passed
Result2 Duration: 0:00:00,0071098
Result2 StackTrace:
Result2 Message:
Result2 StandardOutput: 2.0
Result2 StandardError:

Result3 Name: DeliverHashtable
Result3 Outcome: Passed
Result3 Duration: 0:00:00,0049026
Result3 StackTrace:
Result3 Message:
Result3 StandardOutput: {"test":1}
Result3 StandardError:

Result4 Name: DeliverInt
Result4 Outcome: Passed
Result4 Duration: 0:00:00,000676
Result4 StackTrace:
Result4 Message:
Result4 StandardOutput: 50
Result4 StandardError:

Result5 Name: DeliverJObject
Result5 Outcome: Passed
Result5 Duration: 0:00:00,0012771
Result5 StackTrace:
Result5 Message:
Result5 StandardOutput: {"test":1}
Result5 StandardError:

```
Result6 Name: DeliverLong
Result6 Outcome: Passed
Result6 Duration: 0:00:00,0153377
Result6 StackTrace:
Result6 Message:
Result6 StandardOutput: 50
Result6 StandardError:

Result7 Name: DeliverString
Result7 Outcome: Passed
Result7 Duration: 0:00:00,0008281
Result7 StackTrace:
Result7 Message:
Result7 StandardOutput: "This is a string"
Result7 StandardError:

Result8 Name: MultipleSubscribersOnSameKey
Result8 Outcome: Passed
Result8 Duration: 0:00:00,0533323
Result8 StackTrace:
Result8 Message:
Result8 StandardOutput:
Parsing {"double":2.0, "string":"this is a string", "float":3.2,
"int":"5", "long":"5", "genericlist":["one","two","three"],
"hashtable":{"int":2,"string":"value"}}
Delivering to subscribers: 5
5
Delivering to subscribers: 5
5
Result8 StandardError:

Result9 Name: MultipleSubscriptionOnDifferentKeys
Result9 Outcome: Passed
Result9 Duration: 0:00:00,0851509
Result9 StackTrace:
Result9 Message:
Result9 StandardOutput:
Parsing {"double":2.0, "string":"this is a string", "float":3.2,
"int":"5", "long":"5", "genericlist":["one","two","three"],
"hashtable":{"int":2,"string":"value"}}
Delivering to subscribers: {
  "int": 2,
  "string": "value"
}
{
  "int": 2,
  "string": "value"
}
```

```
}  
Delivering to subscribers: 5  
5  
Result9 StandardError:  
Result10 Name: ReceiveDouble  
Result10 Outcome: Passed  
Result10 Duration: 0:00:00,0291989  
Result10 StackTrace:  
Result10 Message:  
Result10 StandardOutput:  
Parsing {"double":2.0, "string":"this is a string", "float":3.2,  
"int":"5", "long":"5", "genericlist":["one","two","three"],  
"hashtable":{"int":2,"string":"value"}}  
Delivering to subscribers: 2  
2  
Result10 StandardError:  
  
Result11 Name: ReceiveFloat  
Result11 Outcome: Passed  
Result11 Duration: 0:00:00,1161415  
Result11 StackTrace:  
Result11 Message:  
Result11 StandardOutput:  
Parsing {"double":2.0, "string":"this is a string", "float":3.2,  
"int":"5", "long":"5", "genericlist":["one","two","three"],  
"hashtable":{"int":2,"string":"value"}}  
Delivering to subscribers: 3.2  
3.2  
Result11 StandardError:  
  
Result12 Name: ReceiveInt  
Result12 Outcome: Passed  
Result12 Duration: 0:00:00,0156655  
Result12 StackTrace:  
Result12 Message:  
Result12 StandardOutput:  
Parsing {"double":2.0, "string":"this is a string", "float":3.2,  
"int":"5", "long":"5", "genericlist":["one","two","three"],  
"hashtable":{"int":2,"string":"value"}}  
Delivering to subscribers: 5  
5  
Result12 StandardError:  
  
Result13 Name: ReceiveJArray  
Result13 Outcome: Passed  
Result13 Duration: 0:00:00,0485689  
Result13 StackTrace:
```



Result13 Message:

Result13 StandardOutput:

```
Parsing {"double":2.0, "string":"this is a string", "float":3.2,
"int":"5", "long":"5", "genericlist":["one","two","three"],
"hashtable":{"int":2,"string":"value"}}
```

Delivering to subscribers: [

```
"one",
"two",
"three"
```

]

[

```
"one",
"two",
"three"
```

]

Result13 StandardError:

Result14 Name: ReceiveJObject

Result14 Outcome: Passed

Result14 Duration: 0:00:00,0267704

Result14 StackTrace:

Result14 Message:

Result14 StandardOutput:

```
Parsing {"double":2.0, "string":"this is a string", "float":3.2,
"int":"5", "long":"5", "genericlist":["one","two","three"],
"hashtable":{"int":2,"string":"value"}}
```

Delivering to subscribers: {

```
"int": 2,
"string": "value"
```

}

{

```
"int": 2,
"string": "value"
```

}

Result14 StandardError:

Result15 Name: ReceiveLong

Result15 Outcome: Passed

Result15 Duration: 0:00:00,0049686

Result15 StackTrace:

Result15 Message:

Result15 StandardOutput:

```
Parsing {"double":2.0, "string":"this is a string", "float":3.2,
"int":"5", "long":"5", "genericlist":["one","two","three"],
"hashtable":{"int":2,"string":"value"}}
```

Delivering to subscribers: 5

5

Result15 StandardError:

Result16 Name: ReceiveString

Result16 Outcome: Passed

Result16 Duration: 0:00:00,0007274

Result16 StackTrace:

Result16 Message:

Result16 StandardOutput:

```
Parsing {"double":2.0, "string":"this is a string", "float":3.2,
"int":"5", "long":"5", "genericlist":["one","two","three"],
"hashtable":{"int":2,"string":"value"}}
```

```
Delivering to subscribers: this is a string
```

```
this is a string
```

Result16 StandardError:



Bachelor of Engineering

Project appendix

Winter semester 2019

Verification report: VR-SWL-A2

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Thomas Mundal
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains verification report related to Diagnostics System Software Layer Alpha 2

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

VR-SWL-A2

Date performed	18.02.2019
Performed by	Thomas Mundal
Tests performed (16)	UTV-5, UTV-8, UTV-9, UTV-10, UTV-11, UTV-12, UTV-13, UTV-14, UTV-15, UTV-16, UTV-17, UTV-20, UTV-21, UTV-22, UTV-23, IV-07, IV-16, IV-13, IV-14, IV-17, TV-43
Tests discarded (4)	UTV-3, UTV-4, UTV-18, UTV-19
System version	Alpha2

Discarded tests

During testing, and test driven development, some decisions was made regarding operation of the program. Some of the tests were discarded as a result of these decisions, since the operation of the program no longer relies on these tests passing. Decisions were made in part because in order to pass the tests there was a need for implementing unnecessary and redundant code that would incorporate similar behaviour that already exists in the libraries used. Using the correct data types that is meant to be used in conjunction with parsing and serializing JSON data is the correct way to solve the goal these discarded tests aimed to verify.

Inspection IV-07

Insert screenshot with Arduino connected

Inspection IV-16

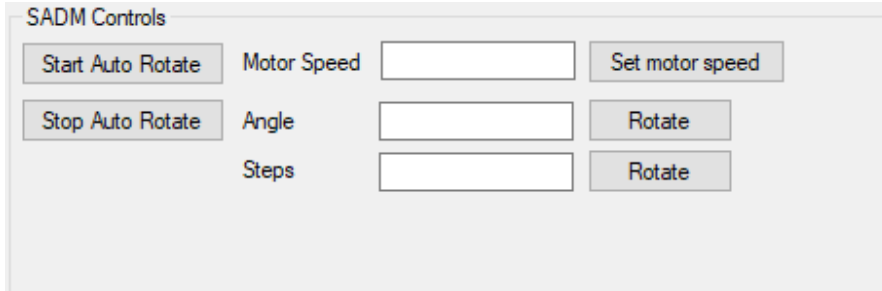
Code and architecture inspection confirms this verification. As well as the fact that the software has easily expanded to allow for other data sources, and arbitrary data field plotting.

Inspection IV-13

Verified trough inspection and functional test

Inspection IV-14

Software layer provides a minimalistic interface for SADM controls. The functionality of this has been verified through functional testing and passes.



The screenshot shows a window titled "SADM Controls". It contains several interactive elements:

- A "Start Auto Rotate" button.
- A "Motor Speed" label followed by an empty text input field and a "Set motor speed" button.
- A "Stop Auto Rotate" button.
- An "Angle" label followed by an empty text input field and a "Rotate" button.
- A "Steps" label followed by an empty text input field and a "Rotate" button.

Inspection IV-17

Code audit verifies that communication protocol is followed.

Inspection TV-43

The implementation of communication protocol is functional and works as intended.

Unit test results dump

Group Name: SatStatTests

Group By: Hierarchy

Group Full Name: SatStatTests

Duration: 0:00:00,0355984

0 test(s) failed

0 test(s) skipped

16 test(s) passed

Result1 Name: DeliverDouble

Result1 Outcome: Passed

Result1 Duration: 0:00:00,0002942

Result1 StackTrace:

Result1 Message:

Result1 StandardOutput: 1.0

Result1 StandardError:

Result2 Name: DeliverFloat
Result2 Outcome: Passed
Result2 Duration: 0:00:00,0017275
Result2 StackTrace:
Result2 Message:
Result2 StandardOutput: 2.0
Result2 StandardError:

Result3 Name: DeliverHashtable
Result3 Outcome: Passed
Result3 Duration: 0:00:00,0012222
Result3 StackTrace:
Result3 Message:
Result3 StandardOutput: {"test":1}
Result3 StandardError:

Result4 Name: DeliverInt
Result4 Outcome: Passed
Result4 Duration: 0:00:00,0002191
Result4 StackTrace:
Result4 Message:
Result4 StandardOutput: 50
Result4 StandardError:

Result5 Name: DeliverJObject
Result5 Outcome: Passed
Result5 Duration: 0:00:00,0002035
Result5 StackTrace:
Result5 Message:
Result5 StandardOutput: {"test":1}



Result5 StandardError:

Result6 Name: DeliverLong

Result6 Outcome: Passed

Result6 Duration: 0:00:00,0007648

Result6 StackTrace:

Result6 Message:

Result6 StandardOutput: 50

Result6 StandardError:

Result7 Name: DeliverString

Result7 Outcome: Passed

Result7 Duration: 0:00:00,00038

Result7 StackTrace:

Result7 Message:

Result7 StandardOutput: "This is a string"

Result7 StandardError:

Result8 Name: MultipleSubscribersOnSameKey

Result8 Outcome: Passed

Result8 Duration: 0:00:00,0003982

Result8 StackTrace:

Result8 Message:

Result8 StandardOutput:

```
Parsing {"double":2.0, "string":"this is a string", "float":3.2,
"int":"5", "long":"5", "genericlist":["one","two","three"],
"hashtable":{"int":2,"string":"value"}}
```

Delivering to subscribers: 5

System.Int32

Delivering to subscribers: 5

System.Int32

Result8 StandardError:

Result9 Name: MultipleSubscriptionOnDifferentKeys

Result9 Outcome: Passed

Result9 Duration: 0:00:00,0005235

Result9 StackTrace:

Result9 Message:

Result9 StandardOutput:

```
Parsing {"double":2.0, "string":"this is a string", "float":3.2,
"int":"5", "long":"5", "genericlist":["one","two","three"],
"hashtable":{"int":2,"string":"value"}}
```

Delivering to subscribers: 5

System.Int32

Delivering to subscribers: {

 "int": 2,

 "string": "value"

}

Newtonsoft.Json.Linq.JObject

Result9 StandardError:

Result10 Name: ReceiveDouble

Result10 Outcome: Passed

Result10 Duration: 0:00:00,0203804

Result10 StackTrace:

Result10 Message:

Result10 StandardOutput:

```
Parsing {"double":2.0, "string":"this is a string", "float":3.2,
"int":"5", "long":"5", "genericlist":["one","two","three"],
"hashtable":{"int":2,"string":"value"}}
```

Delivering to subscribers: 2

System.Double

Result10 StandardError:

Result11 Name: ReceiveFloat

Result11 Outcome: Passed

Result11 Duration: 0:00:00,0006195

Result11 StackTrace:

Result11 Message:

Result11 StandardOutput:

```
Parsing {"double":2.0, "string":"this is a string", "float":3.2,
"int":"5", "long":"5", "genericlist":["one","two","three"],
"hashtable":{"int":2,"string":"value"}}
```

Delivering to subscribers: 3.2

System.Single

Result11 StandardError:

Result12 Name: ReceiveInt

Result12 Outcome: Passed

Result12 Duration: 0:00:00,0005746

Result12 StackTrace:

Result12 Message:

Result12 StandardOutput:

```
Parsing {"double":2.0, "string":"this is a string", "float":3.2,
"int":"5", "long":"5", "genericlist":["one","two","three"],
"hashtable":{"int":2,"string":"value"}}
```

Delivering to subscribers: 5

System.Int32

Result12 StandardError:

Result13 Name: ReceiveJArray

Result13 Outcome: Passed

Result13 Duration: 0:00:00,0041124

Result13 StackTrace:

Result13 Message:

Result13 StandardOutput:

```
Parsing {"double":2.0, "string":"this is a string", "float":3.2,
"int":"5", "long":"5", "genericlist":["one","two","three"],
"hashtable":{"int":2,"string":"value"}}
```

Delivering to subscribers: [

```
"one",  
"two",  
"three"
```

]

Datatype: Newtonsoft.Json.Linq.JArray

Result13 StandardError:

Result14 Name: ReceiveJObject

Result14 Outcome: Passed

Result14 Duration: 0:00:00,0029275

Result14 StackTrace:

Result14 Message:

Result14 StandardOutput:

```
Parsing {"double":2.0, "string":"this is a string", "float":3.2,  
"int":"5", "long":"5", "genericlist":["one","two","three"],  
"hashtable":{"int":2,"string":"value"}}
```

Delivering to subscribers: {

```
"int": 2,  
"string": "value"
```

}

Newtonsoft.Json.Linq.JObject

Result14 StandardError:

Result15 Name: ReceiveLong

Result15 Outcome: Passed

Result15 Duration: 0:00:00,0006053

Result15 StackTrace:

Result15 Message:

Result15 StandardOutput:

```
Parsing {"double":2.0, "string":"this is a string", "float":3.2,  
"int":"5", "long":"5", "genericlist":["one","two","three"],  
"hashtable":{"int":2,"string":"value"}}
```

Delivering to subscribers: 5

System.Int64

Result15 StandardError:

Result16 Name: ReceiveString

Result16 Outcome: Passed

Result16 Duration: 0:00:00,0006457

Result16 StackTrace:

Result16 Message:

Result16 StandardOutput:

```
Parsing {"double":2.0, "string":"this is a string", "float":3.2,
"int":"5", "long":"5", "genericlist":["one","two","three"],
"hashtable":{"int":2,"string":"value"}}
```

Delivering to subscribers: this is a string

System.String

Result16 StandardError:

Bachelor of Engineering

Project appendix

Winter semester 2019

Verification report: VR-SWL-A3

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Thomas Mundal
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains verification report related to Diagnostics System Software Layer Alpha 3

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

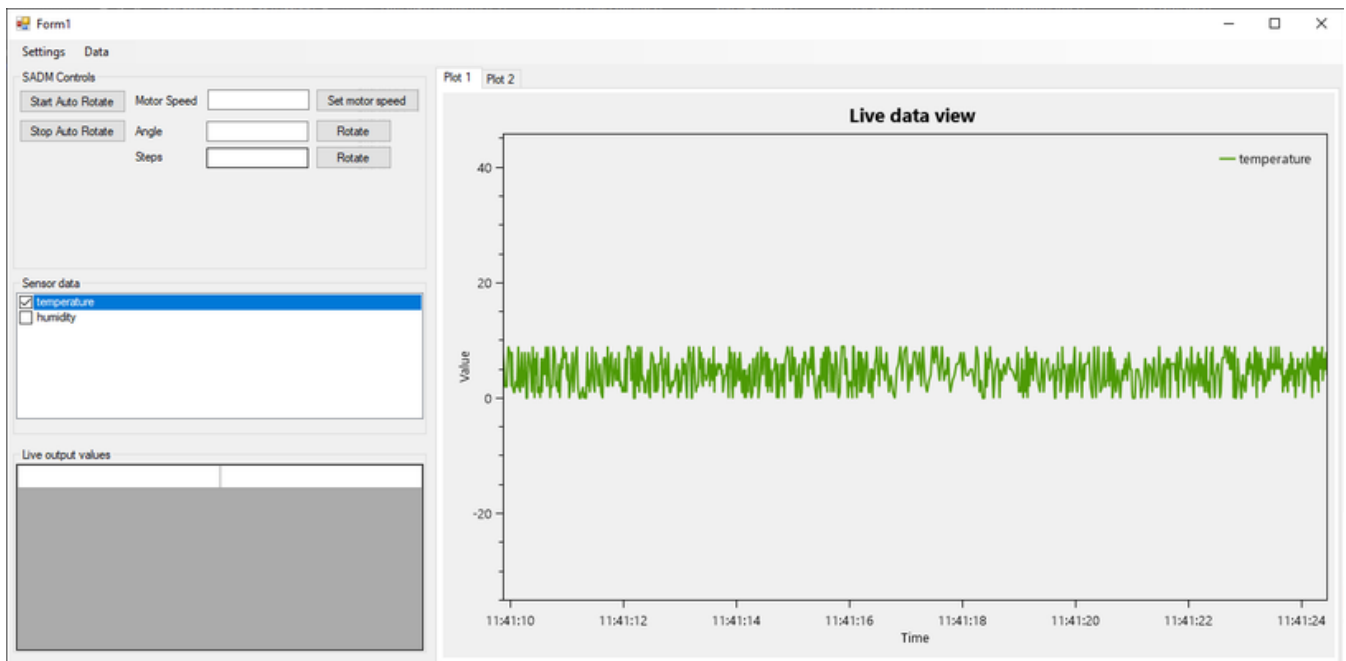
VR-SWL-A3

Date performed	12.05.2019
Performed by	Thomas Mundal
Tests performed (16)	TV-37, TV-38, TV-49
Tests discarded (4)	UTV-1, UTV-2
System version	

TV-37: Functional test

Functional test: View should display correctly after changed to desired output type

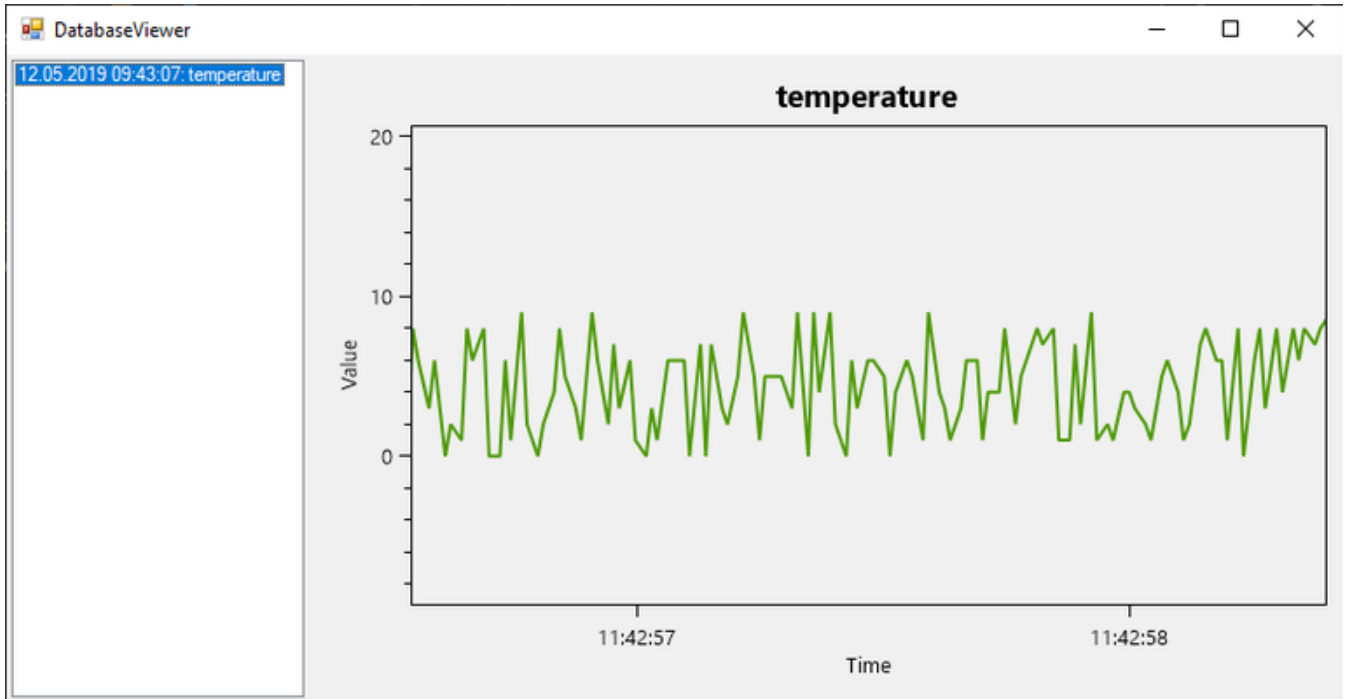
Result: Desired data to view can be selected from the "Sensor data" panel and is displayed as it arrives in the live data view plot.



TV-38: Functional test

Functional test: Output should display historical data according to selected time settings

Saved data can be accessed in the "Display database" dialog, and data saved is displayed in a plot based on the data set selected in the list to the left



TV-49: Functional test

Functional test: Data is saved to database and is readable as values for data plot

Collected data can be accessed in the "Data" menu entry by clicking "Save to database". The data is verified to be saved correctly by inspecting the data through the database viewer dialog (See image from previous test).

Discarded tests

UTV-1 and UTV-2 tests were discarded, since these do not make sense to test as an external library is used to access the database.



Bachelor of Engineering

Project appendix

Winter semester 2019

Verification report: VR-SWL-B1

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Thomas Mundal
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains verification report related to Diagnostics System Software Layer Beta 1

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

VR-SWL-B1

Date performed	08.04.2019
Performed by	Thomas Mundal
Tests performed (16)	UTV-24, UTV-6, IV-15, TV-50
Tests discarded (4)	
System version	Beta 1

Unit tests: UTV-24, UTV-6

UTV-24 and UTV-6 are unit tests that verify that the information is stored in the config entry in a database. The tests also verify that only one entry of this config data exists, and that it is overwritten the next time the data is saved as opposed to creating a new entry in the database.

Inspection: IV-15

Inspecting the saved data would require a software to decode and display the data. Rather than creating a sophisticated software to do this, the inspection was covered by functional testing instead. After connecting initially with the standard connect to COM port workflow, an attempt was made to connect with the previous settings. The connection is successfully established, which suggests that the saved data is correct.

Functional test: TV-50

TV-50 verifies that the UI can display the last value collected from a data stream that is subscribed to.

Live output values	
actual_angle_0	81,8823928833008
desired_position_0	76,0482025146484
AngleError_0	0
adjust_value_0	-0,471592396497726

Unit test report dump

Test Name: SaveComSettings

Test FullName: SatStatTests.DatabaseTests.SaveComSettings

Test Source:

D:\Programming\CSharp\SatStat\SatStat_SWLayer_Tests\DatabaseTests.cs :
line 16

Test Outcome: Passed

Test Duration: 0:00:00,5199366

Result StandardOutput: Setting document does not exist, creating....



Bachelor of Engineering

Project appendix

Winter semester 2019

Verification report: VR-SWL-B1-2

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Thomas Mundal
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains verification report related to Diagnostics System Software Layer Beta 1

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

VR-SWL-B1-2

Date performed	15.04.2019
Performed by	Thomas Mundal
Tests performed (16)	UTV-26, TV-51.A, TV-51.B
Tests discarded (4)	
System version	Beta1

Functional test TV-51

Verifications TV-51.A and TV-51.B are essentially one test covering two sub-requirements (R2-34.01 and R2-34.02). Parameter controls can be saved to a template using the save/load dialog box seen below. The functionality was verified through functional testing defined by the verifications mentioned in this report. The screenshots below show that the data is applied in the parameter control ui after loading.

ParameterControlTemplateSaveDialog

Data attribute	Minimum	Maximum	Attribute	Value	Status	Difference
temperature			temperature	6	Over	6
humidity			humidity	5	Over	5
sine			sine	-0,174326781222962	Under	-0,174326781222962

Unit test results dump

Group Name: ObservableValueTests
Group By: Hierarchy
Group Full Name: ObservableValueTests
Duration: 0:00:00,0717738
0 test(s) failed
0 test(s) skipped
9 test(s) passed

Result1 Name: TestCollection
Result1 Outcome: Passed
Result1 Duration: 0:00:00,0023342
Result1 StackTrace:
Result1 Message:
Result1 StandardOutput:
1
1
System.Int32
0
0
Result1 StandardError:

Result2 Name: TestCollectionInsert
Result2 Outcome: Passed
Result2 Duration: 0:00:00,0005133
Result2 StackTrace:
Result2 Message:
Result2 StandardOutput:
Result2 StandardError:

Result3 Name: TestConainsLabel
Result3 Outcome: Passed
Result3 Duration: 0:00:00,0002377
Result3 StackTrace:
Result3 Message:
Result3 StandardOutput:
Result3 StandardError:

Result4 Name: TestContains

Result4 Outcome: Passed
Result4 Duration: 0:00:00,0004737
Result4 StackTrace:
Result4 Message:
Result4 StandardOutput:
Result4 StandardError:

Result5 Name: TestGeneric
Result5 Outcome: Passed
Result5 Duration: 0:00:00,0659248
Result5 StackTrace:
Result5 Message:
Result5 StandardOutput: False
Result5 StandardError:

Result6 Name: TestIndexOf
Result6 Outcome: Passed
Result6 Duration: 0:00:00,0003311
Result6 StackTrace:
Result6 Message:
Result6 StandardOutput:
Result6 StandardError:

Result7 Name: TestInterfaceArray
Result7 Outcome: Passed
Result7 Duration: 0:00:00,0010013
Result7 StackTrace:
Result7 Message:
Result7 StandardOutput:
Result7 StandardError:

Result8 Name: TestInterfaceList
Result8 Outcome: Passed
Result8 Duration: 0:00:00,0004271
Result8 StackTrace:
Result8 Message:
Result8 StandardOutput:
Result8 StandardError:

Result9 Name: TestStringIndexAccess

Result9 Outcome: Passed
Result9 Duration: 0:00:00,0005306
Result9 StackTrace:
Result9 Message:
Result9 StandardOutput:
Result9 StandardError:



Bachelor of Engineering

Project appendix

Winter semester 2019

Verification report: VR-SWL-B2

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Thomas Mundal
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains verification report related to Diagnostics System Software Layer Beta 2

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

VR-SWL-B2

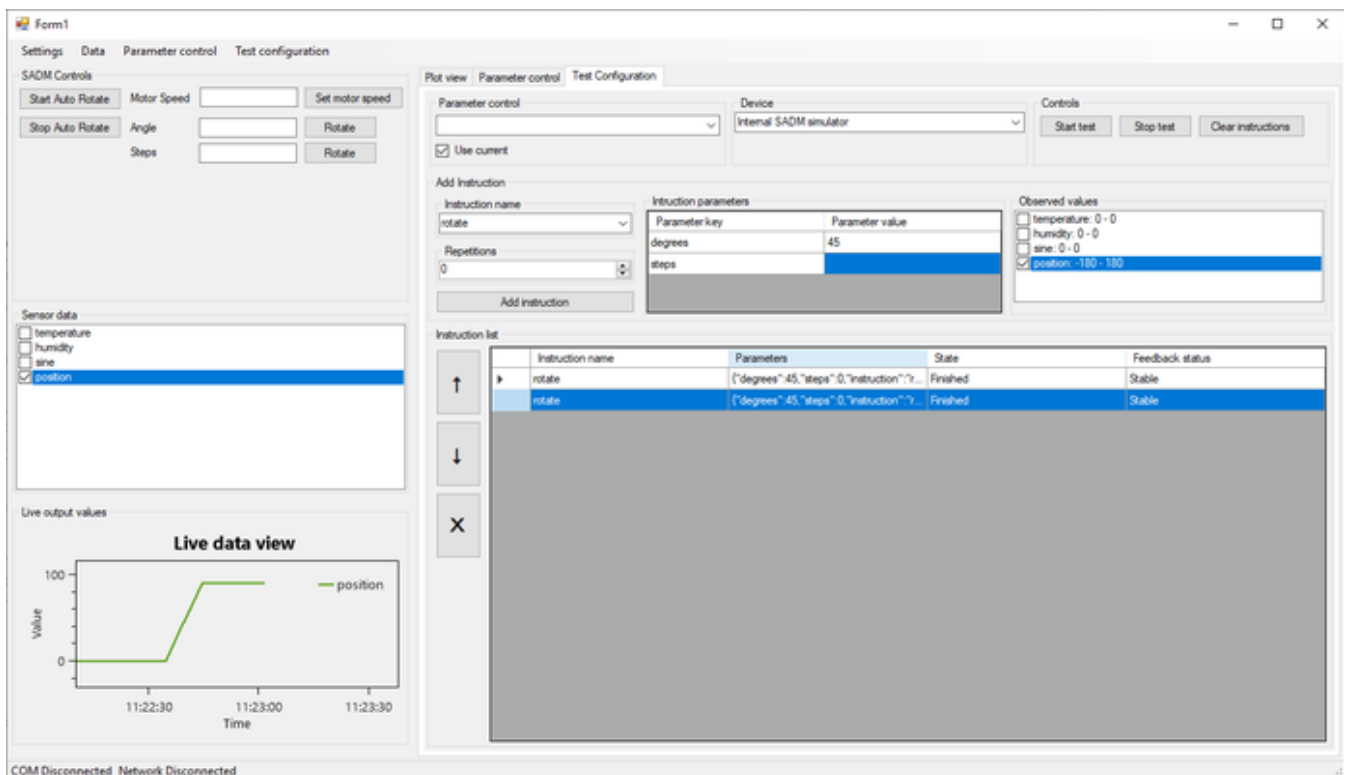
Date performed	18.02.2019
Performed by	Thomas Mundal
Tests performed (16)	TV-53, TV-54
Tests discarded (4)	
System version	Beta 2

TV-53: Functional test

Functional test: A complete test run should run based on configured instructions and give feedback based on configured test parameters

Result:

Test can be configured and run with feedback on selected observable value



TV-54: Functional test

Functional Test: Parameters can be provided through a UI and the parameters can be used to observe whether or not a value is inside their defined bounds

Results:

Control parameters can be set and observed via UI. The parameters put in the Parameter Control UI can be accessed in the UI for test configuration through the parameter control panel with the checkbox "Use current" and the observed values panel when adding an instruction. Saved parameter control templates can be accessed via parameter control panel dropdown menu (see image in previous test).

The screenshot displays the 'Form1' application window with several panels:

- SADM Controls:** Includes 'Start Auto Rotate', 'Stop Auto Rotate', 'Motor Speed' (input field), 'Set motor speed', 'Angle' (input field), 'Rotate', and 'Steps' (input field) with 'Rotate' buttons.
- Sensor data:** A list of sensors with checkboxes: temperature, humidity, sine, and position (checked).
- Live output values:** A 'Live data view' plot showing 'Value' vs 'Time' with a green line for 'position' at a value of 100.
- Parameter control:** A table with columns 'Data attribute', 'Minimum', and 'Maximum'. The 'position' row is highlighted with values -180 and 180.
- Test Configuration:** A table with columns 'Attribute', 'Value', 'Status', and 'Difference'. All entries (temperature, humidity, sine, position) show 'Not configured' status.

At the bottom left, the status bar indicates 'COM Disconnected' and 'Network Disconnected'.

Bachelor of Engineering**Project appendix****Winter semester 2019****SatStat communication protocol**

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Thomas Mundal and Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document describes the communication protocol used between hardware layer (HWL) and software layer (SWL) in the diagnostics system (SatStat).

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date	Signature
Kongsberg, May 20, 2019	

SatStat communication protocol

This document describes the communication protocol used between hardware layer (HWL) and software layer (SWL) in the diagnostics system (SatStat).

Connection modes

The protocol can follow several connection modes, depending on what communication medium is used for communication. Connection modes affect how handshake is performed and what parameters to pass in the handshake.

The connection modes planned are:

- Serial connection
- Socket connection

Connection handshake

The config entry in the HWL `serial_handshake` refers to nr. of data bits(Db), followed by parity(Par) and nr. of stop bits(Sb) in the format "DbParSb", e.g. "8N1" for 8 data bits, no parity and 1 stop bit.

Arduino serial restrictions:

- Number of data bits can be anything from 5 through 8.
- Parity must be defined as either "N" (no parity), "O" (Odd parity) or "E" (Even parity).
- Number of stop bits can be set to either 1 or 2.
- Default is "8N1", 8 data bits, no parity and 1 stop bit.

Acceptable standard baud rates (We want a baud rate of at least 9600 as anything below this might introduce critical delays):

- 9600 (Default)
- 14400
- 19200
- 28800
- 38400
- 57600
- 115200

Protocol handshake

Serial connection handshake

In a serial connection handshake, information about the supported serial communication configuration is shared. The SWL requests a handshake initialization, where the HWL responds with a collection of configuration modes supported based on some standards defined for serial communication for RS-232 on USB. With the information received, SWL can display a user interface control where the user can specify what settings to use during the communication.

The handshake is always initialized with a default serial setting of 9600 8N1.

The next step involves the SWL requesting a connection initialization with settings selected from the configuration options received in the previous step. After the HWL has received this connection initialization, the serial communication is reset and configured with the appropriate settings and reconnected with these settings.

Example handshake

```
SWL: {"serial_handshake":"init"}
HWL: {"serial_handshake":{"baud_rates":[9600,14400,19200,28800,38400,57600,115200], "configs":["8N1", "8O1"], "newlines":["\r\n", "\r", "\n"]}}
SWL: {"connection_request":{"baud_rate":9600, "config":"8N1", "newline":"\r\n"}}
HWL: {"connect":"init"} | At this point, the connection is reset and re-established with the selected configuration
HWL: {"connect":"init"} | HWL keeps sending init "ping" until the SWL has reconfigured and can read the message correctly, or if a timeout is reached
HWL: {"connect":"init"}
SWL: {"connect":"ok"} | SWL has been configured, and read the initialization message correctly
```

Timeout or incomprehensible data received at HWL

If the HWL receives something incomprehensible, or times out before receiving what's expected at the current stage in the handshake protocol, it will send a Negative Acknowledgement (NACK) as follows:

```
HWL: {"serial_handshake":"failed"}
```

Sensor data discovery

After the protocol handshake is performed, and a connection is established, the HWL provides a list of data fields available for subscription.

```
HWL: {"available_data":{"temperature":"double", "humidity":"int"}}
```

Instruction discovery

New in Beta 2.

A list of available instructions is sent so that the client knows what instructions can be run on the HWL. When this list is available, the client can select instructions from a list rather than either having to be updated when a new instruction is added, or requiring the user to remember all of the instruction names and parameters when defining automated tests. This list is sent from HWL after available data is provided. Example:

```
HWL: {"available_instructions":{"set_speed":{"rpm":"float"}}, "set_direct"
```

```
tion":{"direction":"string"},"rotate_degrees":{"steps":"float"}}}
```

Data subscription

New in Beta 1.

When the SWL has information about the data fields available, it can subscribe or unsubscribe to delivery of data on that field.

```
SWL: {"request":"subscribe", "parameters": ["temperature", "humidity"]}
SWL: {"request":"unsubscribe", "parameters": ["temperature",
"humidity"]}
```

Subscription delivery

The HWL delivers data to the SWL based on the fields subscribed to.

```
{"temperature":24.0, "humidity":5}
```

Other requests from SWL

The SWL can requests operations on the HWL through the following semantic:

For Alpha 2 and 3: {"instruction":"name_of_operation", "parameter":param}

New in Beta 1: {"request":"name_of_operation", "parameters":["param1", "param2", ...]}

If no parameter: {"request":"name_of_operation"}

Available instructions

Listed below are the possible instructions that the SWL can send to the HWL. The argument value will be replaced by a value of the given type in an actual instruction.

- **Removed in Beta 2.** The auto_rotate instructions switches auto rotate of the SADM either on or off according to the passed boolean value:
{"instruction":"auto_rotate", "enable":boolean value}
- **Prior to Beta 2.** The rotate instruction has two different versions. One that takes an argument of type int defining the number of steps, and one that takes a float defining the number of degrees to rotate.
{"instruction":"rotate", "steps":int value}
{"instruction":"rotate", "deg":float value}
- **New in Beta 2.** Separate instructions for rotating steps and rotating degrees.
{"instruction":"rotate_steps", "steps":int value}
{"instruction":"rotate_degrees", "degrees":float value}
- **New in Beta 2.** The step size is a preset value specific to a given stepper motor which represents the number of degrees the motor rotates for each full step.
{"instruction":"set_step_size", "step_size":float value}
- **New in Beta 2.** Stepping mode is a value defining the portion of a full step the motor

actually turns when prompting one step. The parameter of this instruction is the divisor of the preferred stepping mode.

```
{"instruction":"set_stepping_mode","divisor":int value}
```

- **New in Beta 2.** For the SADM, the gear ratio is defined by dividing the diameter of the shaft gear by the diameter of the motor gear.

```
{"instruction":"set_gear_ratio","ratio":float value}
```

- **New in Beta 2.** The speed can be set by providing the preferred RPM as parameter.

```
{"instruction":"set_speed","rpm":float value}
```

- **New in Beta 2.** The direction can be set to either forward or reverse.

```
{"instruction":"set_direction","direction":"forward"}
```

```
{"instruction":"set_direction","direction":"reverse"}
```

- **New in Beta 2.** Reset takes no parameters. Sending this will reset the SADM controller (the Arduino).

```
{"request":"reset"}
```

Bachelor of Engineering

Project appendix

Winter semester 2019

SatStat user manual

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Thomas Mundal
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document describes how the software SatStat is meant to be used. SatStat is the name of the software layer in the diagnostics system.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

Contents

- 1 Introduction** **2**
- 2 System requirements** **3**
- 3 Installation** **3**
- 4 Connecting to data source** **3**
 - 4.1 Connecting to COM device 3
 - 4.2 Connecting to external simulation 5
 - 4.3 Internal stream simulator 5
- 5 Reading data** **5**
 - 5.1 Selecting data to plot 5
 - 5.2 Viewing saved data plots 5
- 6 Saving data** **6**
- 7 Parameter control** **6**
 - 7.1 Define control parameters 7
 - 7.2 Save parameter control template 7
 - 7.3 Load parameter control template 8
- 8 Automated testing** **8**
 - 8.1 Create a test configuration 8
 - 8.1.1 Add parameter control 9
 - 8.1.2 Add instructions 9
 - 8.2 Save a test configuration 9
 - 8.3 Load a test configuration 10
 - 8.4 Run a test 10
- 9 References** **12**

1 Introduction

SatStat is the name given to the software that represents the software layer of the diagnostics system in the bachelor project for SatLight 2019. The main purpose of the software is to act as a human-machine-interface (HMI) towards the hardware unit that sends control signals to electrical components, as well as records sensory data from sensors installed in the mechanical unit (REF?). Using the software, a test engineer can read the measured data from installed and configured sensors, and control the motor on the mechanical unit.

The main window is displayed in figure 1, where the components of this window is highlighted in red and referenced with a number. address table 1 to get a description of each of these components.

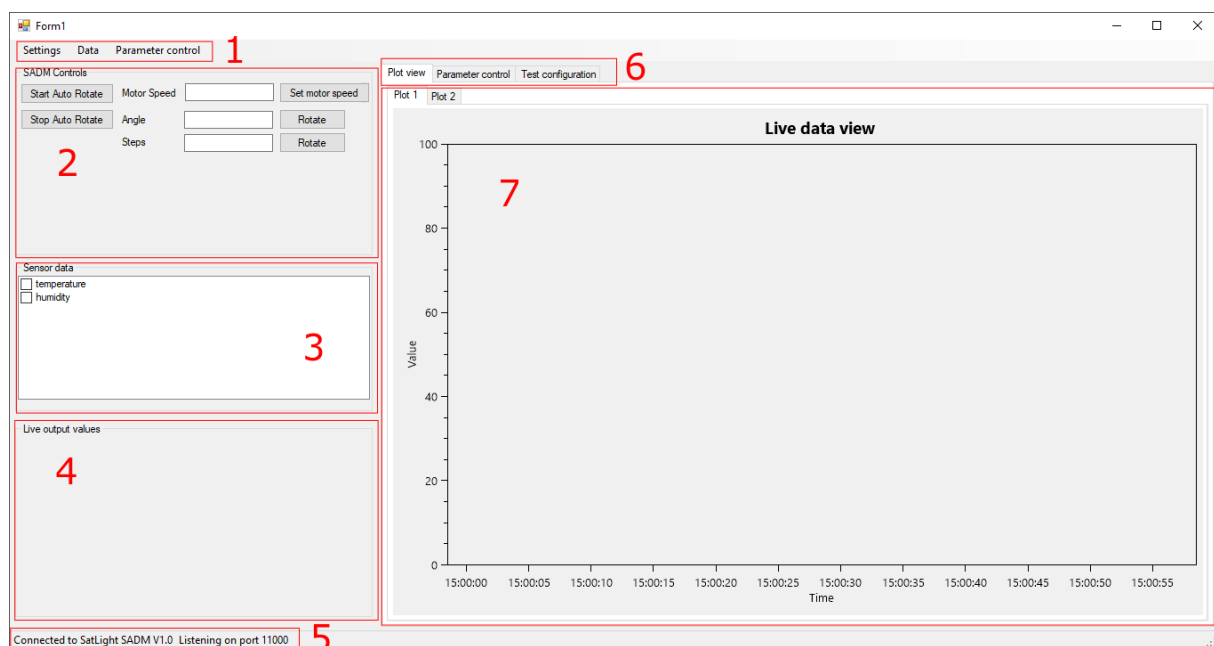


Figure 1: SatStat main window

1	This is the main menu where you can find functions related to settings, data and parameter control. The settings menu is where you connect or disconnect devices. The data menu is related to storing data in the local database and viewing data stored within that database. In the parameter control menu, you can save and load parameter control templates (ref)
2	This panel has direct controls for controlling a SADM unit if connected.
3	The sensor data list is a list of sensor outputs that the connected hardware or simulation provides. Ticking one or more of these boxes, will display the data collected in a plot in the plot view (panel 7), and the last collected value in panel 4
4	The live output values panel shows the last value that was collected from each of the ticked boxes in panel 3
5	This is the status indicator bar, showing connection status towards serial devices, or devices connected on the network
6	This is a tab panel containing tabs for different main components of the software
7	Here is where live data is displayed in a plot when data is collected from a connected device.

Table 1: Description of the components in the main window

2 System requirements

The software is built and tested on Windows 10 with .NET framework 4.6.1 [1].

3 Installation

To install the software, download the latest release from <https://github.com/thmundal/SatStat/releases>. Any official release has an installer you can use.

4 Connecting to data source

There are two main sources for data that the software can communicate with, as well as one internal simulated data source primarily for testing. Depending on the version of the software that you have, the internal stream simulator may or may not be available.

All data sources that are configured to communicate with this software must follow a pre-defined communication protocol (REF) in order for the software to correctly interpret the data received. The communication protocol is described in its own document.

4.1 Connecting to COM device

The primary data source used with SatStat is an external hardware device communicating through RS232 over USB [2]. To set up a connection towards a compatible hardware device, click "Settings" in the main menu strip, followed by "Connect to serial device..." as seen in figure 2.

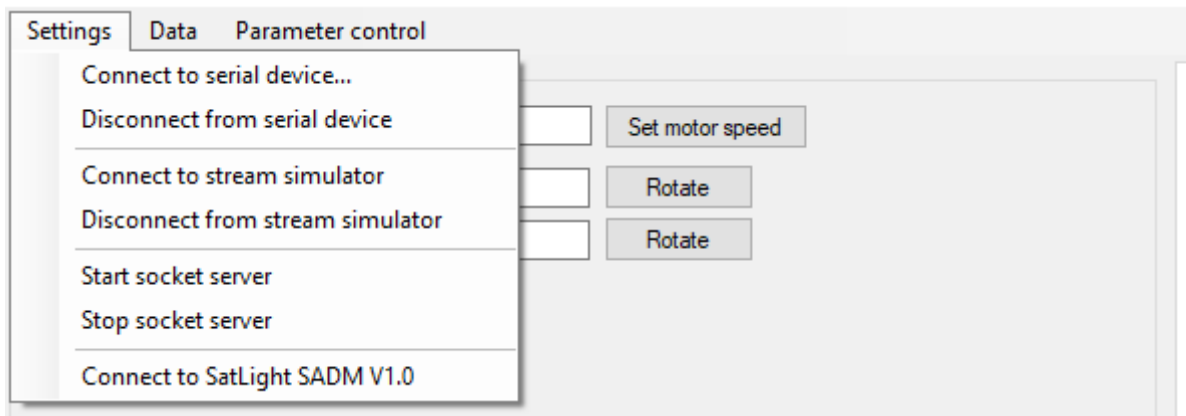


Figure 2: Connect to serial device

After clicking "Connect to serial device...", a dialog box will appear displaying all available devices connected on serial ports on your computer. Note that not all of these devices are compatible with the software, this list simply displays all devices present on the USB hub that uses serial communication.

Select your compatible serial device, and click the next button to initialize the handshake process. After a short delay, a list of available baud rates should appear. Choose the baud rate appropriate for your setup, normally 9600 is good enough.

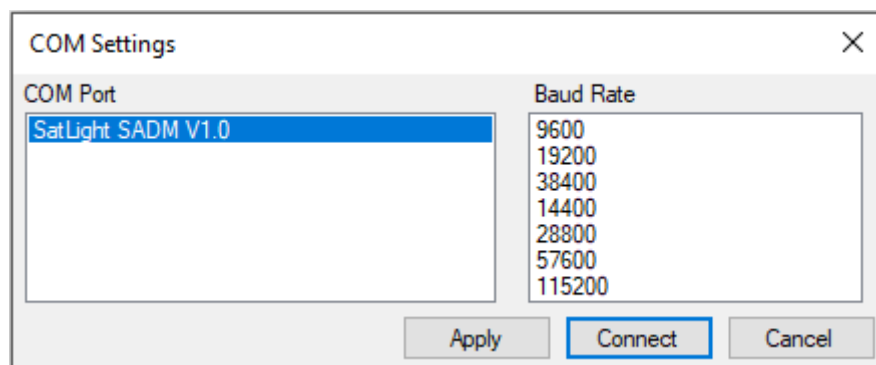


Figure 3: Dialog box for connecting to a serial device

To finalize the connection setup, ensure that you have a baud rate selected from the list and click "Connect". You should now be connected to the hardware device. A connection indicator is displayed in the bottom of the main window. If this says "Connected to [your device]" as seen in figure 4, the connection was a success. If the status displays "Serial disconnected", a connection has not been made, and any other message indicates that an error has occurred in some step of the handshake process.

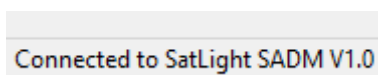


Figure 4: Status indicator indicating that a COM connection was successful

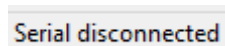


Figure 5: No serial device is connected

4.2 Connecting to external simulation

SatStat can read information from an external simulation connected through a network socket connection. The client that runs the simulation or provides the data must comply with the SatStat communication protocol. To connect an external simulation to SatStat, first start the socket server from the SatStat software. You do this from the main menu strip by clicking "Settings", then "Start socket server" as seen in figure 2. When the socket server has started, the network connection indicator at the bottom of the main window will display "Listening on port 11000" as seen in figure 6.

Listening on port 11000

Figure 6: Socket server started and ready for connection

4.3 Internal stream simulator

The internal stream simulator is mainly used for internal testing of the software, and replaces the hardware unit with a locally simulated version that can both supply data and read instructions. This feature will likely be removed from the software in later stable versions after extensive user-testing is complete. The stream simulator can be started from the "Settings" menu strip item seen in figure 2.

5 Reading data

5.1 Selecting data to plot

The available data fields from a connected device can be viewed in the panel labeled "Sensor data", seen as panel number 3 in figure 1. To view data on a specific field, simply tick the checkbox next to the name of the data field you wish to monitor, and the data will appear in real time in the live data view panel, labelled as panel number 7 in figure 1, and the current value of this data field will display in panel number 4. To stop receiving data from a field, simply untick the checkbox next to the field name in the sensor data panel.

5.2 Viewing saved data plots

If you have saved previously collected data, you can view this data in a plot. To open a previously saved data set, click the "Data" entry in the main menu strip showed highlighted as panel 1 in figure 1. In the "Data" menu, click "Display Database" to open the database view as seen in figure 7.

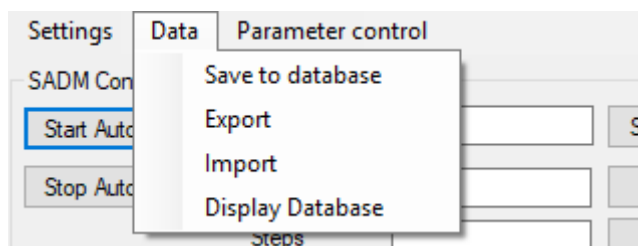


Figure 7: The data menu

After clicking "Display Database", a new window will appear showing a list of data-set on the left side, and an empty plot to the right. To view a data-set, simply click on it, and the data will appear in the plot to the right. If you wish to view multiple data-set at once, you can hold the shift or control buttons to select multiple data-set. The data-sets have data points that are associated with time, so if you select two data-sets that are recorded at separate times, the plot will adjust accordingly and the data could be spread apart very widely. An example of a loaded data set into the view database dialog is displayed in figure 8.

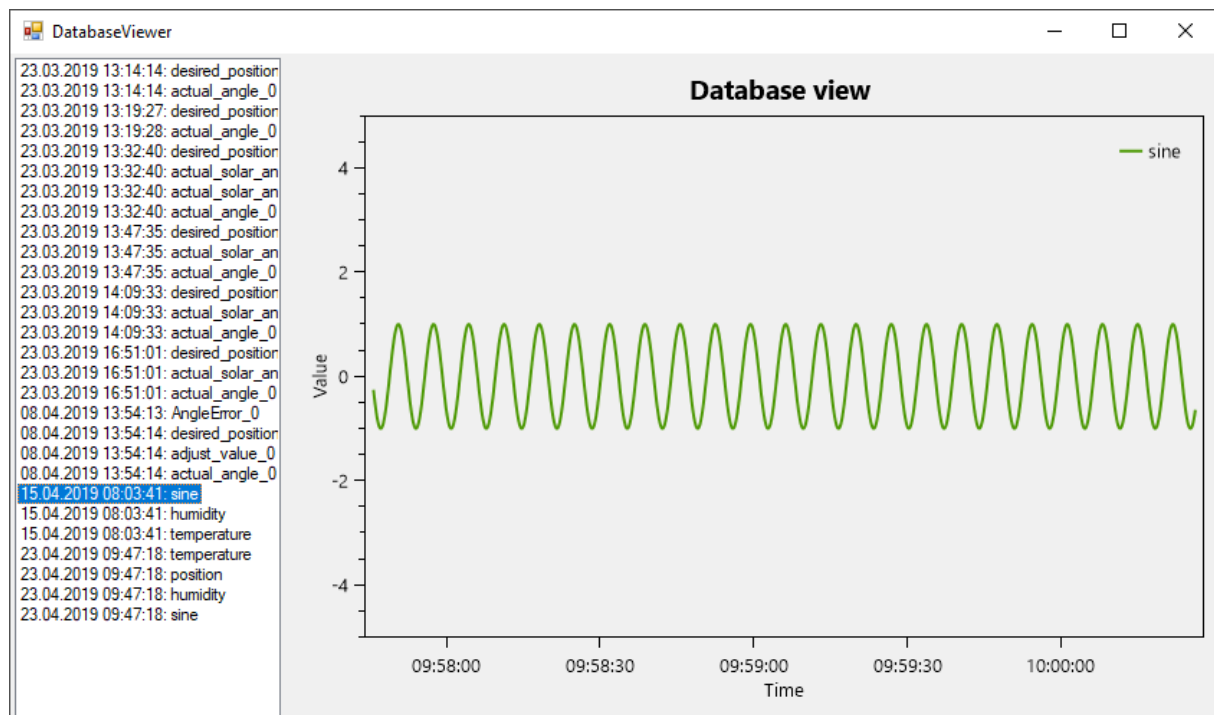


Figure 8: Database view with a loaded data set

6 Saving data

You can save data that has been collected during your current session. If you have some data collected in the main plot view, simply click the "Data" menu entry, and then click "Save to database" (see figure 7). This will save your current plot to the database, and you can view it through the database view. Keep in mind that if you have collected several fields of data, each field is saved as a separate data-set. To view multiple data set at once in the database view, please refer to section 5.2.

7 Parameter control

The parameter control feature of SatStat allows you to monitor data from a data stream with respect to upper and lower bounds for the data. These bounds are set by you as you see fit, and the software will compare the value of the data with the upper and lower bounds that you set. This will give you an indication of whether or not the system is stable or not.

The parameter control feature is accessed by the "Parameter control" tab, displayed in panel 6 in figure 1.

Annotated image of parameter control here

Data attribute	Minimum	Maximum	Attribute	Value	Status	Difference
temperature			temperature		Not configured	Not configured
humidity			humidity		Not configured	Not configured
sine			sine		Not configured	Not configured
position			position		Not configured	Not configured

Figure 9: The parameter control tab

7.1 Define control parameters

After connecting to a device that feeds data to SatStat, you should have a list of data attribute names in a table, together with a field for minimum and maximum values. This table is displayed in the "Parameter control" tab as seen in figure 9. To the right of the input table is the output table displaying the current value of the data attribute, the current status related to the minimum and maximum values, as well as the difference if the value deviates from the set range. If no data has been collected on an attribute, the status and difference will display "Not configured", as seen in figure 9.

7.2 Save parameter control template

A parameter control setup can be saved as a parameter control template to be loaded at a later date. A parameter control template can also be loaded into a test configuration where it is necessary to monitor sensor values closely during testing. To save a parameter control template, access the "Parameter Control" menu, and click "Save template". This will display the parameter control template save dialog box seen in figure 10.

ParameterControlTemplateSaveDialog

Template list

- Temphum
- Sin
- Sine stable
- Sine unstable over

Template name

Description

Delete Save Cancel

Figure 10: Parameter control save dialog

In the parameter control template save dialog box, enter the name of the template you wish

to save, and optionally add a description. The template will be saved as soon as you click the "Save" button, and the dialog will close.

7.3 Load parameter control template

To load a previously saved parameter control template, access the "Parameter Control" menu, and click "Load template". A dialog will appear similar to the parameter control template save dialog, where the save button is exchanged for a load button, as seen in figure 11. Select the template you wish to load, and click the "Load" button. This will load the parameter control values into the parameter control table, and the dialog will close.

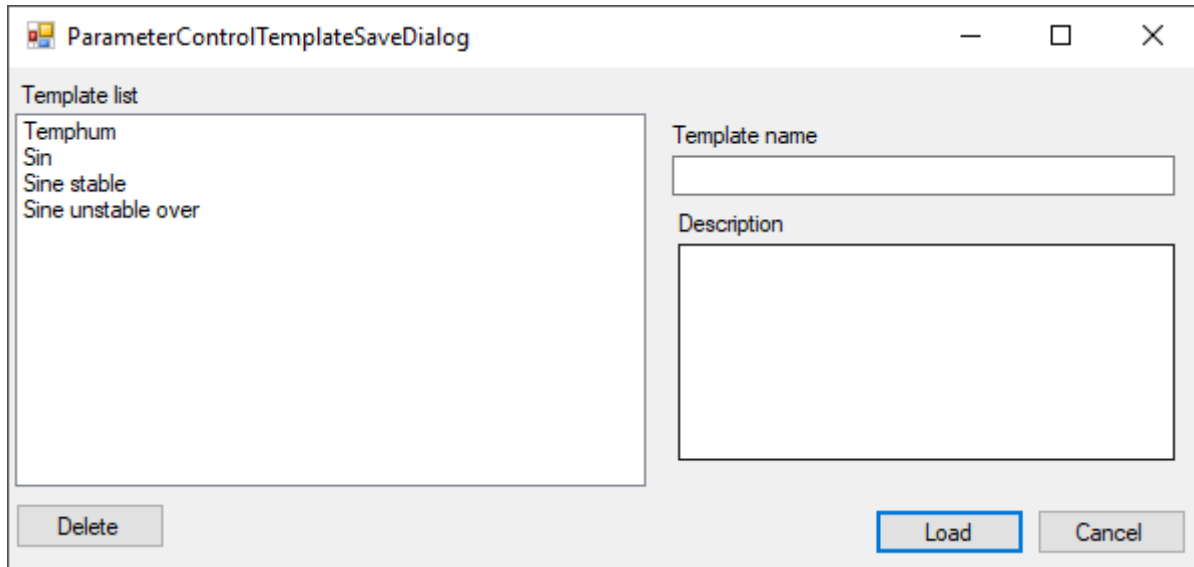


Figure 11: Parameter control load dialog

8 Automated testing

The automated testing feature is where you can set up a test configuration and queue several instructions to send to the connected unit. During testing the software can collect and measure sensor data and compare it to your defined upper and lower limits for the sensor value. You get live feedback on the state of the hardware unit while the test is progressing.

8.1 Create a test configuration

To create a test configuration, first access the "Test configuration" tab labelled in panel 6 in figure 1. In this view, you can start setting up the test configuration right away.

Figure 12: Test configuration tab

8.1.1 Add parameter control

To add parameter control to your test, select a parameter control template from the parameter control dropdown menu, or tick the checkbox under it to use the current parameter control setup (see section 7).

8.1.2 Add instructions

To add instructions to your test, refer to the "Add instruction" panel within the Test Configuration tab. Select the instruction to run from the "Instruction name" dropdown menu. When an instruction is selected, the instruction parameters table is populated with possible arguments to send with the instruction. In most cases the instruction requires the parameters to have a value if the instruction is to be performed successfully.

If you want to monitor sensor values during this particular instruction, and if you have set up parameter control, you can select the sensor value to watch in the list of checkboxes on the "Output parameters" checkbox list.

8.2 Save a test configuration

To save a test configuration that you have set up, access the "Test configuration" menu strip item, and click "Save test configuration". A dialog box will appear, where you can enter a name

for the configuration, and a description for it. Click the "Save" button when you are ready to save it.

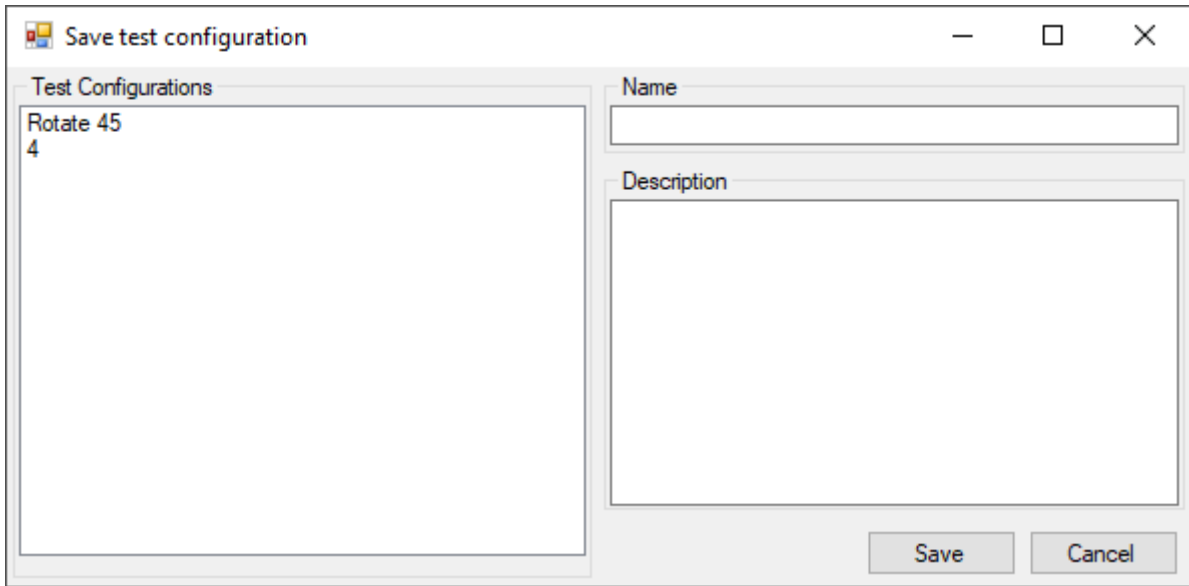


Figure 13: Test config save dialog

8.3 Load a test configuration

To save a test configuration that you have set up, access the "Test configuration" menu strip item, and click "Load test configuration". A dialog box will appear, where you can select a test configuration to load. When you have selected a saved configuration, click the "Load" button, the dialog will close and the test configuration tab will be populated with the setup loaded.

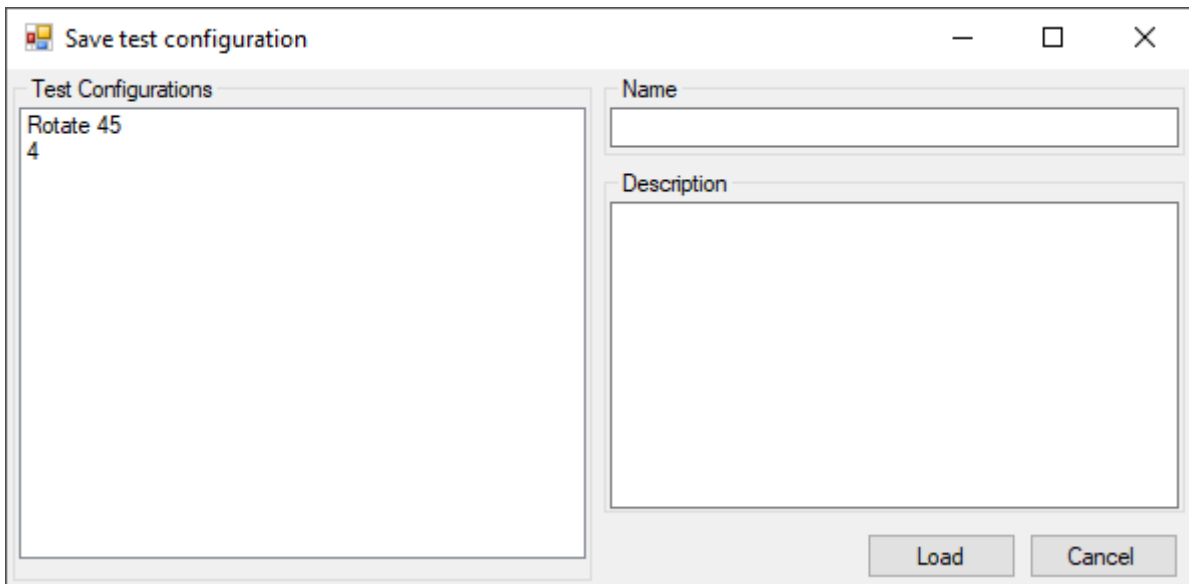


Figure 14: Test config load dialog

8.4 Run a test

To run a test that has been set up with instructions and parameter controls, you first have to select which connected device to run the test on. Select a connected device in the "Device"

dropdown menu. After everything is set up, click the "Start test" button to start the test.

9 References

- [1] Microsoft, "Download microsoft .net framework 4.6.1 (web installer) for windows 7 sp1, windows 8, windows 8.1, windows 10, windows server 2008 r2 sp1, windows server 2012 and windows server 2012 r2 from official microsoft download center," <https://www.microsoft.com/en-us/download/details.aspx?id=499810> [Accessed: May 09, 2019].
- [2] Sparkfun.com, "Serial communication - learn.sparkfun.com," <https://learn.sparkfun.com/tutorials/serial-communication/all> [Accessed May 09, 2019].



Bachelor of Engineering

Project appendix

Winter semester 2019

HWL Decisions

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

For the hardware layer (HWL) of the diagnostic system, there were a few decisions that had to be made. The purpose of this document is to justify these decisions, and to present the other alternatives that has been considered.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

Contents

1	FPGA or Microcontroller	2
2	USB or Serial port	2
3	IDE	3
4	Data-interchange format	3
5	Libraries	3
5.1	LinkedList	3
5.2	QueueArray	4
5.3	Temperature and humidity sensors and libraries	4
5.4	Stepper	5
5.5	ArduinoJson	5
5.5.1	Versions	5
5.6	Subscriber System Template Library	6
6	Replacing the input and output handlers	6
7	Removal of type specific JSON containers and general handler	7
8	Replacing the Instruction handler	9
9	References	11

1 FPGA or Microcontroller

The choice between using a FPGA device [1] or a Microcontroller unit (MCU) [2] is only applicable to the diagnostic system part of the project. The prototype part of the project is not supposed to have a controller, and it is therefore possible to leave out the considerations that would have been necessary if this controller was going to be a part of the prototype.

A main topic at one of the first meetings with the external supervisor was the importance of what the “main computer” sends and receives, and in what format. The conclusion was that the “main computer” is outside the scope of this project, and that the requirements for the diagnostic system could be defined along the way.

The project model used takes advantage of an iterative development process. This means that it starts off with an initial idea which expands rapidly throughout the different phases of the project. This was the major turning point to decide what controller to use. The initial idea was to use an Arduino MCU [3] for simplicity. If at a later point it had turned out that using a FPGA device would have been more appropriate than using an Arduino, the iterative model would have allowed a change of controller. The specific MCU used in this project is the Arduino Mega 2560 Rev3 [4]. This decision was made based on experience using this specific MCU, as well as having it available at project start.

As it turns out, the Arduino Mega microcontroller is more than capable of handling the different tasks required of the diagnostics system hardware layer. Using a FPGA device would bring unnecessary costs to the solution, and the system would barely benefit from the additional capabilities it would have brought to the table.

2 USB or Serial port

For the diagnostic system there has to be a defined interface between the HWL and the software layer (SWL). The reason for this is that the HWL has to receive “instructions” from the SWL, and return feedback data based on sensor readings. For this to work, both the HWL and the SWL needs to be able to transmit data in the same format in order to understand each other.

The main two communication methods considered are USB and Serial port communication. Both of these are valid options, but USB has the edge over Serial port in terms of transmit speed.

This decision also correlates to the choice of controller as the physical limitations and advantages of the controller are important factors. The Arduino Mega microcontroller has built in support for serial communication via the USB port. Utilizing this significantly reduced the amount of time spent getting the communication up and running. Initially it seemed like this approach could cause noteworthy delay as the program grows larger. Swapping out the temperature and humidity sensor and its corresponding library significantly reduced the overall delay. Because of this, the serial communication delay remains negligible.

3 IDE

Programming different Arduino MCUs is usually done using the Arduino IDE[5]. The reason being is that it is really light weight and allow for rapid development of smaller standalone applications. The Visual Studio IDE[6] also supports Arduino development through an extension called Visual Micro[7]. In Visual Studio it is possible to have multiple projects in one solution. This means that it's possible to modify and run both the SWL and the HWL in the same solution. It is also much easier to deal with larger projects consisting of multiple files in Visual Studio than in the Arduino IDE. Considering this, Visual Micro in conjunction with the Visual Studio IDE is the best solution for this project.

4 Data-interchange format

As mentioned earlier, the data transmitted between the SWL and the HWL need to be standardized for the two layers to understand each other. For this purpose the JSON data-interchange format [8] seemed perfect, and as both the SWL and the HWL are compatible with JSON libraries [9][10], it was unreasonable to spend time considering other alternatives.

5 Libraries

5.1 LinkedList

The HWL needs to hold different sensors in some sort of collection. The data collected from the different sensors also needs to be stored in a similar way. The optimal way to achieve this is by having a list containing key-value pairs. This makes it possible to search for a given key and retrieve the corresponding value, where the key would be the name of the sensor or data entry, and the value would be the sensor- or data object.

For a normal C++ application a hash table would probably be the best alternative to serve this purpose. Finding or developing a suitable hash table for this application would have been fairly complex considering the target platform. The HWL target platform is an Arduino Mega, and in that case most of the Standard C++ Library[11] is unavailable, including the standard map[12] and the hash function[13].

In the course "Operating Systems, Network and programming" Nils Erlend Heggem developed a linked list library namely LinkedList. This library was designed for use on an Arduino microcontroller, and therefore suits this project well. Over the course of this project additional features has been added to this library, and when utilizing it's full potential some major bugs were discovered. To solve these problems Mr. Heggem joined in for a complete overhaul of the library.

5.2 QueueArray

When receiving serial data from the SWL, that data has to remain accessible until it is time to interpret it. Unlike the sensors and their corresponding data objects, the serial data can be removed from where they are stored when retrieved, since they are being processed immediately. There are mainly two data structures used for this purpose, either a stack or a queue. In this case it is preferable to use a queue rather than a stack, because it makes sense to process the data in the order it was received. The QueueArray library[14] was used for this purpose.

5.3 Temperature and humidity sensors and libraries

In the initial stage of the project we had access to the DHT11 temperature and humidity sensor. This sensor worked great for testing purposes, but according to Limor Fried's "Dht11, dht22 and am2302 sensors" guide [15], the DHT22 sensor has a slightly lower sampling rate in favor of better accuracy. A sampling rate of 2s is negligible when it comes to temperature since it is not likely to change in that period. The libraries available for these sensors are sequential, meaning that reading the sensor will delay execution of consecutive code, and therefore the most important factor when deciding what sensor to use is the time consumed when reading the sensor. To reduce the time consumed when reading, both sensors were tested with different libraries to check what configuration turned out best. The different libraries tested are "DHTlib" [16] Versions 0.1.00 and 0.1.29, and "DHT-sensor-library" [17] both with and without utilizing the "Adafruit unified sensor driver" [18]. The results of the tests are visualized in the figures below:

```
{"temperature":26,"humidity":18}  
Execution time: 23ms
```

Figure 1: DHT11 with DHTlib V0.1.00

```
{"temperature":23,"humidity":19}  
Execution time: 21ms
```

Figure 2: DHT11 with DHTlib V0.1.29. Same result with DHT-sensor-library.

```
{"temperature":23.6,"humidity":34.2}  
Execution time: 23ms
```

Figure 3: DHT22 with DHTlib V0.1.00

```
{"humidity":29.3,"temperature":24.4}  
Execution time: 5ms
```

Figure 4: DHT22 with DHTlib V0.1.29

```
{"temperature":24.2,"humidity":30.6}  
Execution time: 4ms
```

Figure 5: DHT22 with DHT-sensor-library. Not using unified library.

```
{"temperature":24.4}  
Execution time: 3ms
```

Figure 6: DHT22 with DHT-sensor-library. Using unified library.

The test results tell us that by changing from the DHT11 sensor with "DHTlib V0.1.00" to DHT22 with the "DHT-sensor-library" utilizing the "Adafruit unified sensor driver", the delay when reading is reduced from 23ms to 3ms, which is a huge improvement.

5.4 Stepper

The stepper motor used in versions Alpha 1 through Beta 1 was the 28BYJ-48 5V Stepper Motor[19]. It was used in conjunction with the ULN2003 stepper motor driver[20]. This specific driver was programmed using the Stepper library[21] because it is simple and widely used with this specific configuration. As of Beta 2, the 28BYJ-48 stepper motor was replaced by a SANMOTION F2 - SS24[22] series stepper motor. This motor is controlled using the EM-314 stepper motor driver[23], and with this driver the Stepper library is no longer needed.

5.5 ArduinoJson

As mentioned in section 4, both the SWL and the HWL are compatible with JSON libraries. For the HWL specifically, the library used is the ArduinoJson library[10]. Not only is it the most popular JSON library for Arduino, it is also the most popular among all Arduino libraries on GitHub. This library simplifies serializing and deserializing of JSON formatted data.

5.5.1 Versions

The version of the library used in the project to begin with was version 5[24]. This version worked really well, but it seemed like version 6[25] had a slight edge over version 5 in terms of ease of use. One important difference between the two versions is that version 6 replaced the DynamicJsonBuffer[26] class with the DynamicJsonDocument[27] class. When testing the new version, it turned out that a DynamicJsonDocument is not really dynamic as the DynamicJsonBuffer was. The only difference from a StaticJsonDocument and a DynamicJsonDocument is that the static one is allocated on the stack, where as the "dynamic" one is allocated on the heap. The parts of the HWL code base using the ArduinoJson library benefit from the fact that the DynamicJsonBuffer actually is dynamic. Therefore an "upgrade" to version 6 would require a lot of additional work, which is why version 5 was never replaced with the newer one.

5.6 Subscriber System Template Library

The SWL of the DS has the ability to request available data from the HWL. This information is displayed in the UI for the user to decide what data to collect, utilizing a so called subscriber system. Up until version Beta 1 of the DS, the HWL always sent all available sensor data. The user was still able to determine what data to examine, as the SWL itself neglected the redundant data. Reading sensors and transmitting data takes time, therefore implementing a subscriber system for the HWL as well, will improve overall efficiency of the system. The initial idea for this implementation was to incorporate flags marking the different data as subscribed or not subscribed. This idea was discarded since the data containers also had to be rewritten in order to support generic data types. This led to the implementation of the Subscriber System Template Library (SSTL).

6 Replacing the input and output handlers

In Alpha 1, there were separate handlers for input and output (Fig 7). This seemed like a good idea at first, but as it turned out, it complicated things a lot. The Input_handler was responsible for receiving serial data and reading sensors, whereas the Output_handler was responsible for sending serial data and controlling the SADM. When the code base started growing larger, these two handlers became responsible for almost everything. The structure of the code became messy, and it became clear that structural changes had to be made. For versions Alpha 2 and later, these handlers were replaced with the Serial_handler which handles serial input and output, and the Instruction_handler which handles interpretation and execution of instructions (Fig 8). The addition of the Sensor_container and the SADM_functions class relieved the previously mentioned handlers of reading the sensors and controlling the SADM.

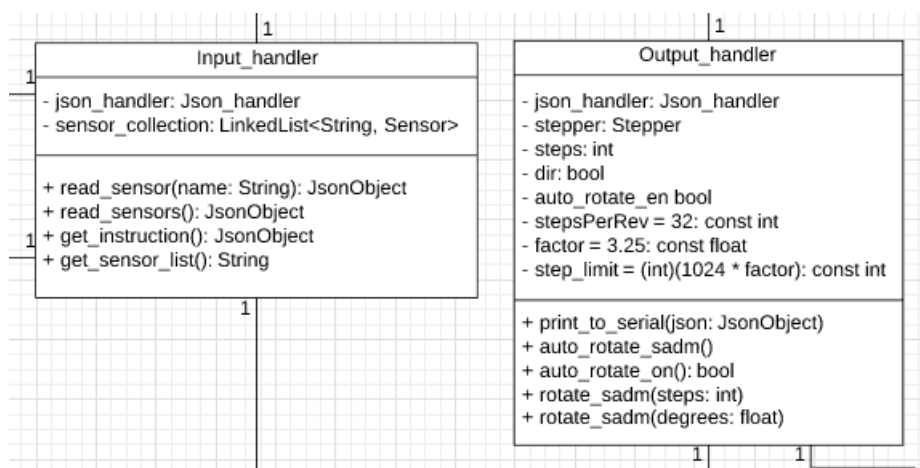


Figure 7: Input- and output handlers.

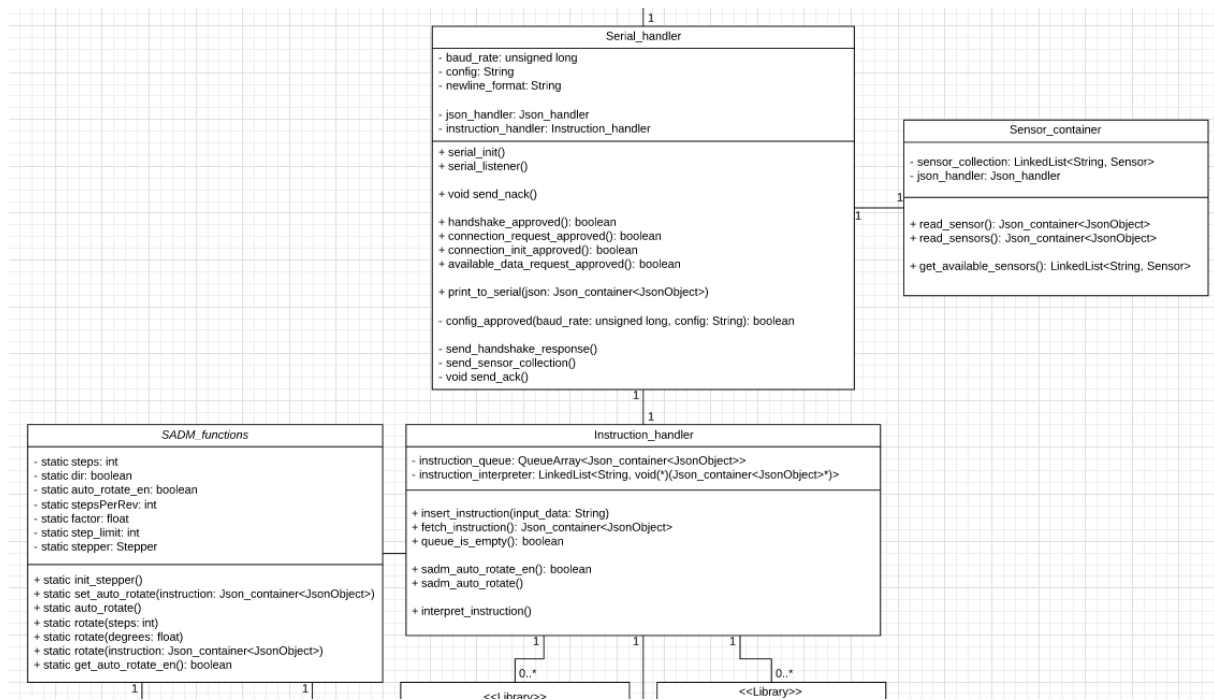


Figure 8: Serial_handler, Instruction_handler, Sensor_container and SADM.functions.

7 Removal of type specific JSON containers and general handler

From Alpha 2 through Beta 1, the parts of the code base concerning the JSON handling were split among four classes. These classes were `Json_container`, `Json_object_container`, `Json_array_container` and `Json_handler` (Fig 9).

In Beta 2 the idea of overloading the class member access operator (`->`) [28] was discovered. This provided direct access to the `m.json` objects' methods by calling this operator. By implementing this concept in the `Json_container`, and replacing the `Json_object_container` and `Json_array_container` with overloads of the `create` and `parse` methods in the `Json_container`, all the classes complementary to the `Json_container` could be removed (Fig 10).

A consequence of this change was that all the classes that used the old JSON structure had to be changed to work with the new solution. This took a lot of time, but simplified the code a whole lot.

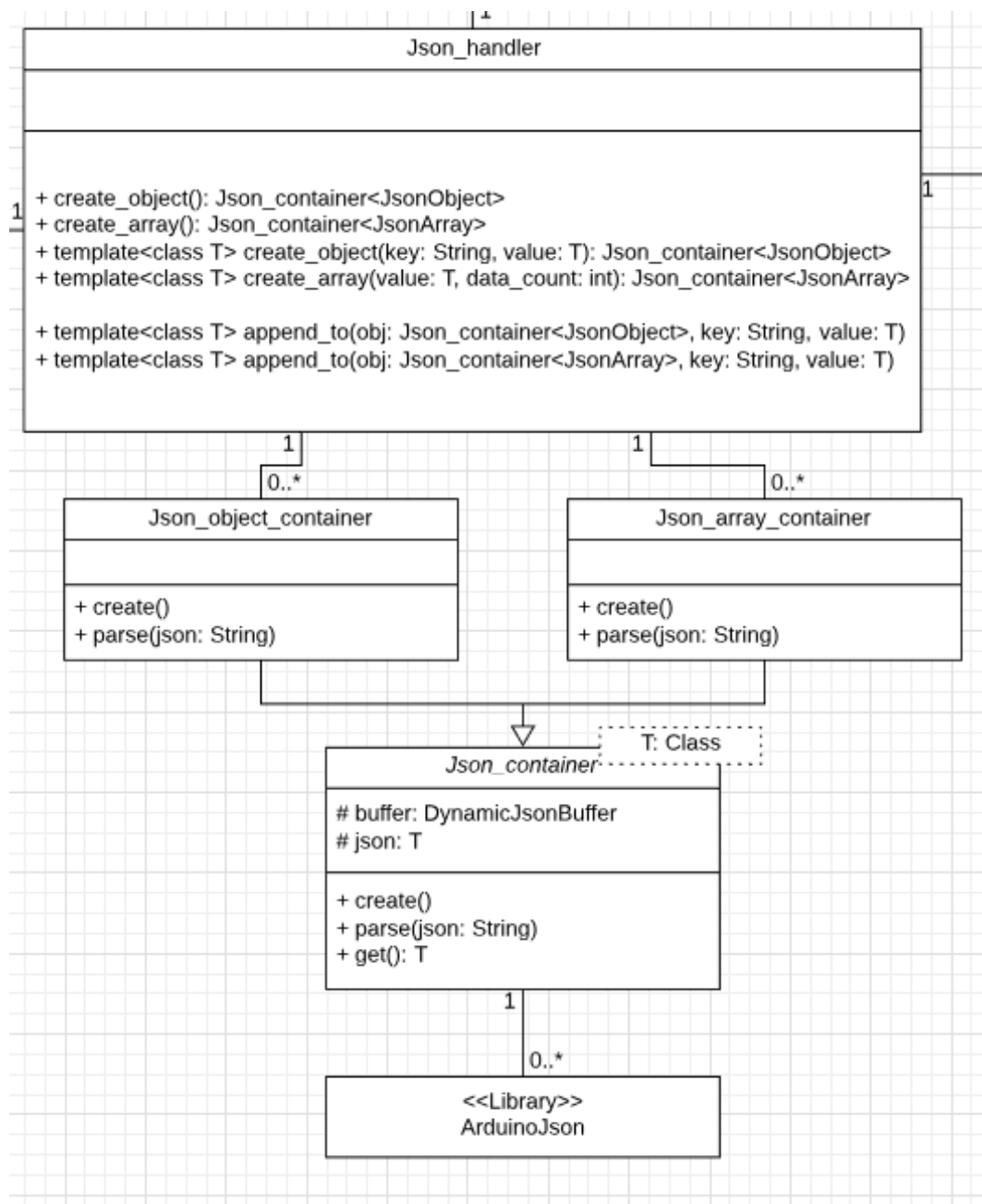


Figure 9: JSON classes from Alpha 2 through Beta 1.

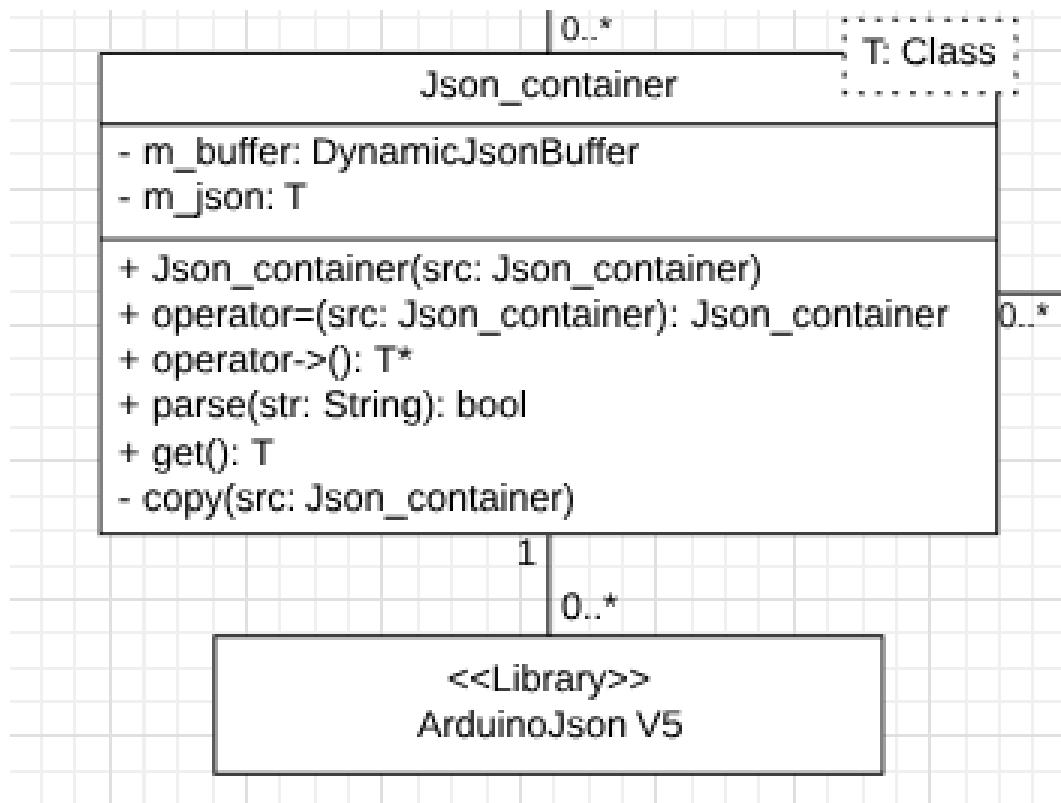


Figure 10: JSON class for Beta 2.

8 Replacing the Instruction_handler

As mentioned earlier, the Instruction_handler (Fig 11) was added in Alpha 2. It was present up until Beta 2 where it was replaced by the Message_handler and the complementary Instruction_container and Request_container (Fig 12). The reason for this replacement was that up until Beta 2, all data received through the serial port was considered instructions. This turned out to be a problem as the HWL now have to report back to the SWL when the execution of a SADM instruction is done. The data received concerning sensor data subscription or handshake related data do not have to report back when processed, and therefore data received through the serial port is split into two categories. These categories are instructions and requests. Both instructions and requests are considered messages when received, but when they are stored they get separated into different lists.

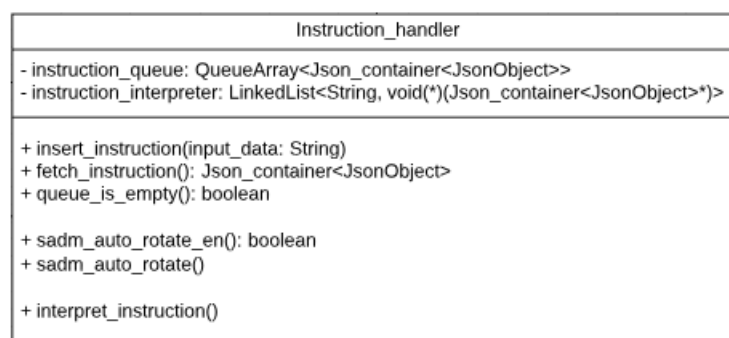


Figure 11: Instruction_handler.

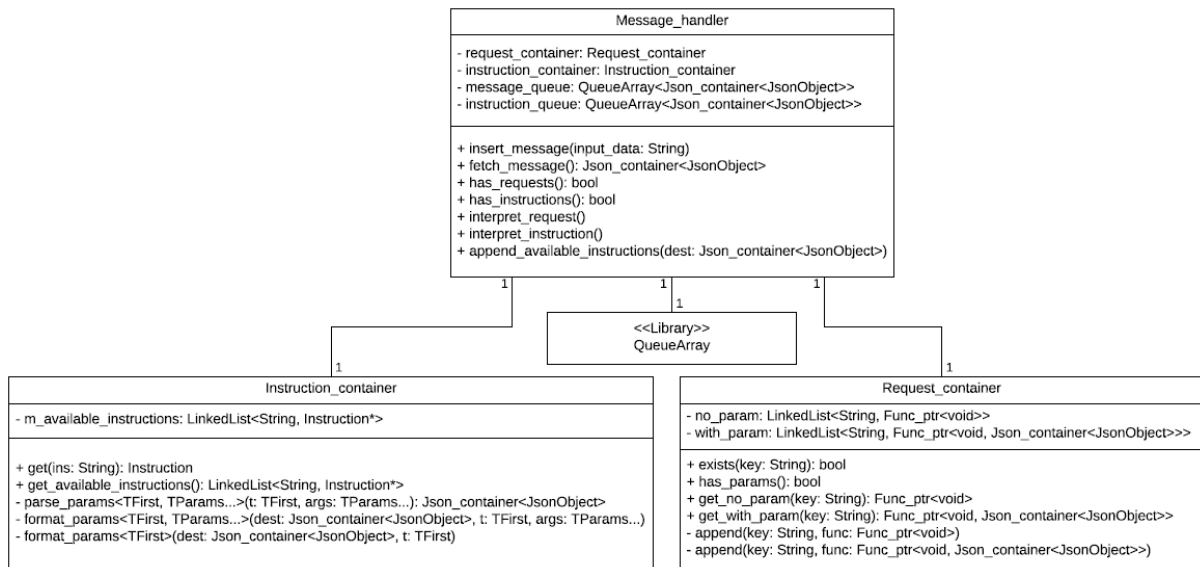


Figure 12: Message_handler, Instruction_container and Request_container.

9 References

- [1] A. Zibell, “What are fpgas and how do they work,” Online, May 2014, available: PDF, https://indico.cern.ch/event/283113/contributions/1632265/attachments/522019/720041/Zibell_How_FPGAs_work.pdf [Accessed: April 21, 2019].
- [2] P. Alley, “Introductory microcontroller programming,” May 2011, pages 1–3, Available: PDF, <https://web.wpi.edu/Pubs/ETD/Available/etd-042811-095908/unrestricted/alley.pdf> [Accessed: April 21, 2019].
- [3] Arduino, “Arduino introduction,” Online, available: Arduino, <https://www.arduino.cc/en/Guide/Introduction> [Accessed: April 21, 2019].
- [4] —, “Arduino mega 2560 rev3,” Online, available: Arduino, <https://store.arduino.cc/mega-2560-r3> [Accessed: April 21, 2019].
- [5] —, “Arduino - software,” Online, available: <https://www.arduino.cc/en/Main/Software> [Accessed: May 09, 2019].
- [6] Microsoft, “Visual studio community,” Online, available: <https://visualstudio.microsoft.com/vs/community/> [Accessed: May 09, 2019].
- [7] Visual Micro, “Arduino ide for visual studio,” Online, available: <https://www.visualmicro.com/> [Accessed: May 09, 2019].
- [8] ECMA International, “The json data interchange syntax,” <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf> [Accessed: March 15, 2019].
- [9] Newtonsoft, “Popular high-performance json framework for .net,” 2019, <https://www.newtonsoft.com/json> [Accessed: March 04, 2019].
- [10] B. Blanchon, “Arduinjson: Efficient json serialization for embedded c++,” Online, 2018, available: ArduinoJson, <https://arduinojson.org/> [Accessed: April 21, 2019].
- [11] The C++ Resources Network, “Standard c++ library reference,” Online, available: <http://www.cplusplus.com/reference/> [Accessed: May 09, 2019].
- [12] —, “Map,” Online, available: <http://www.cplusplus.com/reference/map/map/> [Accessed: May 09, 2019].
- [13] —, “Default hash function object class,” Online, available: <http://www.cplusplus.com/reference/functional/hash/> [Accessed: May 09, 2019].
- [14] E. Chatzikyriakidis, “Queuearray library for arduino,” Online, available: <https://playground.arduino.cc/Code/QueueArray/> [Accessed: May 09, 2019].
- [15] L. Fried, “Dht11, dht22 and am2302 sensors,” Online, August 2018, available: PDF, <https://www.mouser.com/ds/2/737/dht-932870.pdf> [Accessed: April 21, 2019].
- [16] R. Tillaart, “Dht library,” Online, September 2018, available: Github, <https://github.com/RobTillaart/Arduino/tree/master/libraries/DHTlib> [Accessed: April 21, 2019].

- [17] Adafruit Industries, “Adafruit dht humidity & temperature sensor library,” Online, April 2019, available: Github, <https://github.com/adafruit/DHT-sensor-library> [Accessed: April 21, 2019].
- [18] —, “Adafruit unified sensor driver,” Online, March 2019, available: Github, https://github.com/adafruit/Adafruit_Sensor [Accessed: April 21, 2019].
- [19] Kiatronics, “28byj-48 – 5v stepper motor,” Online, available: <http://robocraft.ru/files/datasheet/28BYJ-48.pdf> [Accessed: May 09, 2019].
- [20] —, “4 phase uln2003 stepper motor driver pcb,” Online, available: <https://www.electronicoscaldas.com/datasheet/ULN2003A-PCB.pdf> [Accessed: May 09, 2019].
- [21] Arduino, “Stepper library,” Online, available: <https://www.arduino.cc/en/reference/stepper> [Accessed: May 09, 2019].
- [22] SANYO DENKI, “2-phase stepping motor 42 mm square by 11.6 thin,” Online, available: http://www.farnell.com/datasheets/356617.pdf?_ga=2.66716649.692565864.1556612087-2062706234.1556612087 [Accessed: May 09, 2019].
- [23] Electromen, “Em-314 stepper motor driver 6a 12-24v,” Online, available: <http://www.zilvertron.com/media/electromen/EM-314.pdf> [Accessed: May 09, 2019].
- [24] B. Blanchon, “Documentation — arduinojson 5,” Online, available: ArduinoJson, <https://arduinojson.org/v5/doc/> [Accessed: May 09, 2019].
- [25] —, “Documentation — arduinojson 6,” Online, available: ArduinoJson, <https://arduinojson.org/v6/doc/> [Accessed: May 09, 2019].
- [26] —, “Jsonbuffer — arduinojson 5,” Online, available: ArduinoJson, <https://arduinojson.org/v5/api/jsonbuffer/> [Accessed: May 09, 2019].
- [27] —, “Jsondocument — arduinojson 6,” Online, available: ArduinoJson, <https://arduinojson.org/v6/api/jsondocument/> [Accessed: May 09, 2019].
- [28] tutorialspoint, “Class member access operator (– >) overloading in c++,” Online, available: https://www.tutorialspoint.com/cplusplus/class_member_access_operator_overloading.htm [Accessed: May 10, 2019].

Bachelor of Engineering

Project appendix

Winter semester 2019

SatStat Hardware Layer Alpha 1 - Requirements and verification

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

The hardware layer (HWL) of the Diagnostics System has two main responsibilities. One of these is to store test instructions received from the Software Layer (SWL), and translate these instructions to the proper format in order to execute them on the device corresponding to the initial instruction. The SWL also expect a continuous stream of sensor readings coming from the HWL. This requires the HWL to be able to read and transmit the prompted sensor data. This documents describes the requirements the HWL has to fulfill, and the various tests/verifications that has to be carried out in order to ensure that these requirements are met.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date	Signature
Kongsberg, May 20, 2019	

Diagnostics System - Hardware Layer version Alpha1

Target release	ALPHA 1
Epic	USN5-95 - Hardware layer of the Diagnostics System TO DO
Document status	V 1.0
Document owner	Jon Skjelsbek
Designer	Jon Skjelsbek
Tech lead	Jon Skjelsbek

Objective

The hardware layer(HWL) of the Diagnostics System has two main responsibilities. One of these is to store test instructions received from the Software Layer(SWL), and translate these instructions to the proper format in order to execute them on the device corresponding to the initial instruction. The SWL also expect a continuous stream of sensor readings coming from the HWL. This requires the HWL to be able to read and transmit the prompted sensor data. This documents describes the requirements the HWL has to fulfill, and the various tests/verifications that has to be carried out in order to ensure that these requirements are met.

Success metrics

Goal	Metric
Receive and store instructions from SWL.	Received instructions match the sent one.
Fetch instructions from input handler and pass to output handler.	The controller knows which functions to use for fetching and outputting instructions of different types.
Translate and execute fetched instructions.	The execution result replicate the initial instruction.
Provide prompted sensor data	SWL continuously receive the sensor data it asked for, and the corresponding values makes sense.

Roadmap

The HWL and SWL follow the same roadmap as when functionality is added to one layer, the other layer has to support it.

ID	Task Name	Start	Finish	Duration	10 feb 2019					17 feb 2019					24 feb 2019					3 mar 2019								
					11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	1	2	3	4	5	6
1	Design	11.02.2019	13.02.2019	3d	█																							
2	Implementation	13.02.2019	16.02.2019	4d						█																		
3	Verification	15.02.2019	17.02.2019	3d						█																		
4	Validation	16.02.2019	17.02.2019	2d						█																		
5	Alpha 1	18.02.2019	18.02.2019	0d						◆																		
6	Design	19.02.2019	22.02.2019	4d						█																		
7	Implementation	21.02.2019	24.02.2019	4d											█													
8	Verification	23.02.2019	24.02.2019	2d											█													
9	Validation	24.02.2019	24.02.2019	1d											█													
10	Alpha 2	25.02.2019	25.02.2019	0d											◆													
11	Design	26.02.2019	03.03.2019	6d											█													
12	Implementation	28.02.2019	06.03.2019	7d																█								
13	Verification	05.03.2019	10.03.2019	6d																█								
14	Validation	08.03.2019	10.03.2019	3d																█								
15	Alpha 3	11.03.2019	11.03.2019	0d																◆								

Requirements

ID	Description	Importance	Version
R2-30	The DS Hardware layer must be able to control the SADM for testing purposes.	HIGH	PENDING
Date	Discussion and decisions		
06.02.2019			
User Story			
<div style="border: 1px solid gray; padding: 5px;"> <p>USN5-96 - As an embedded developer, I need my hardware to be able to control the SADM to be able to measure the result.</p> <p>DONE</p> </div>			
Verification ID	Verification method	Status	
TV-39	Functional test: Run as many instructions as see fit, and confirm that the desired result is achieved.	PENDING	

ID	Description	Importance	Version
R2-31	The DS Hardware layer must be able to transmit sensor data to the Software layer for comparison and visualization purposes.	HIGH	ALPHA 1

Date	Discussion and decisions	
06.02.2019		
User Story		
USN5-97 - As an embedded developer, I need to provide sensible sensor data to verify that the systems behaves as expected.		
DONE		
Verification ID	Verification method	Status
TV-40	Functional test: Transmit as much sensor data as see fit, and confirm that the receiver is able to properly read the data.	PASSED

ID	Description	Importance	Version
R2-36	The DS Hardware layer shall be able to interpret and execute instructions received from the software layer.	HIGH	PENDING
Date	Discussion and decisions		
22.02.2019			
User Story			
USN5-131 - As a test engineer, I need the ability to calibrate position and speed of the SADM drive shaft so that I can set correct initial conditions for operation			
DONE			
Verification ID	Verification method	Status	
TV-44	Functional test: Run as many instructions as see fit, and confirm that the desired result is achieved.	PENDING	

ID	Description	Importance	Version
R4-17	DS Hardware layer shall have a standardized expandable input handler, buffering input data according to input type. Rev. A) DS Hardware layer shall have a general input handler responsible for every kind of input.	HIGH	ALPHA 1
Date	Discussion and decisions		

06.02.19 Rev. A) 21.02.19		
User Story		
USN5-98 - As an embedded developer, I need to make sure that instructions can be received even though another instruction is currently executing. DONE		
Verification ID	Verification method	Status
IV-08	Confirm that the code replicates the class diagrams and follow related standards.	PASSED

ID	Description	Importance	Version
R4-18	DS Hardware layer shall have a standardized expandable output handler translating the input data passed from a specific controller, and output the translated information to the device to be controlled. Rev. A) DS Hardware layer shall have a general output handler supporting the different protocols/formats needed in order to communicate with the different connected devices.	HIGH	ALPHA 1

Date	Discussion and decisions	
06.02.19 Rev. A) 21.02.19		
User Story		
USN5-99 - As an embedded developer, I need to adapt the instructions to the proper format for the receiving device to understand. DONE		
Verification ID	Verification method	Status
IV-09	Confirm that the code replicates the class diagrams and follow related standards.	PASSED

ID	Description	Importance	Version
----	-------------	------------	---------

R4-19	There shall be an interface between the input handler and the output handler (controller). Preferably one for each device to be controlled. Rev. A) There shall be an interface responsible for fetching from the input handler and passing to the output handler.	HIGH	ALPHA 1
Date	Discussion and decisions		
06.02.19 Rev. A) 21.02.19			
User Story			
USN5-100 - As an embedded developer, I need to make sure that the correct instructions are executed at the correct device for the test results to be correct. DONE			
Verification ID	Verification method	Status	
IV-10	Confirm that the code replicates the class diagrams and follow related standards.	PASSED	

ID	Description	Importance	Version
R4-17.01	The input handler of the DS Hardware layer must be compatible with the software layer of the diagnostic system. Rev. A) The input handler must support JSON input received from the software layer.	HIGH	PENDING
Date	Discussion and decisions		
06.02.19 Rev. A) 21.02.19			
User Story			
USN5-101 - As an embedded developer, I need the receiving end of my hardware to speak the same language as the software layer in order to properly interpret the received instructions. DONE			
Verification ID	Verification method	Status	
TV-41	Functional test: Send as many instructions as see fit from the software layer, and confirm that the data received is properly handled.	PENDING	

ID	Description	Importance	Version
R4-18.01	The output handler of the DS Hardware layer shall support the different protocols needed in order to communicate with the different connected devices. Rev. A) The output handler shall incorporate serial printing of JSON formatted data.	HIGH	ALPHA 1
Date	Discussion and decisions		
06.02.19 Rev. A) 21.02.19			
User Story			
USN5-102 - As an embedded developer, I need the transmitting end of my hardware to speak languages that the receiving devices understand in order to transmit data in the correct format. DONE			
Verification ID	Verification method	Status	
TV-42	Confirm that the code replicates the class diagrams and related standards. Addition in Rev. A) Check that the software layer receives the data as expected.	PASSED	

ID	Description	Importance	Version
R5-22	The Diagnostics Systems shall not interfere with the SADM.	MEDIUM	PENDING
Date	Discussion and decisions		
06.02.2019			
User Story			
USN5-103 - As an embedded developer, I need to make sure my hardware doesn't affect the SADM as they are two separate systems. OPEN			
Verification ID	Verification method	Status	
IV-12	Confirm that the DS and SADM is indeed two separate systems.	PENDING	

Links to external sources

Alpha 1 class diagram: <https://goo.gl/r4PfjG>



Bachelor of Engineering

Project appendix

Winter semester 2019

SatStat Hardware Layer Alpha 1 - Class Diagram

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

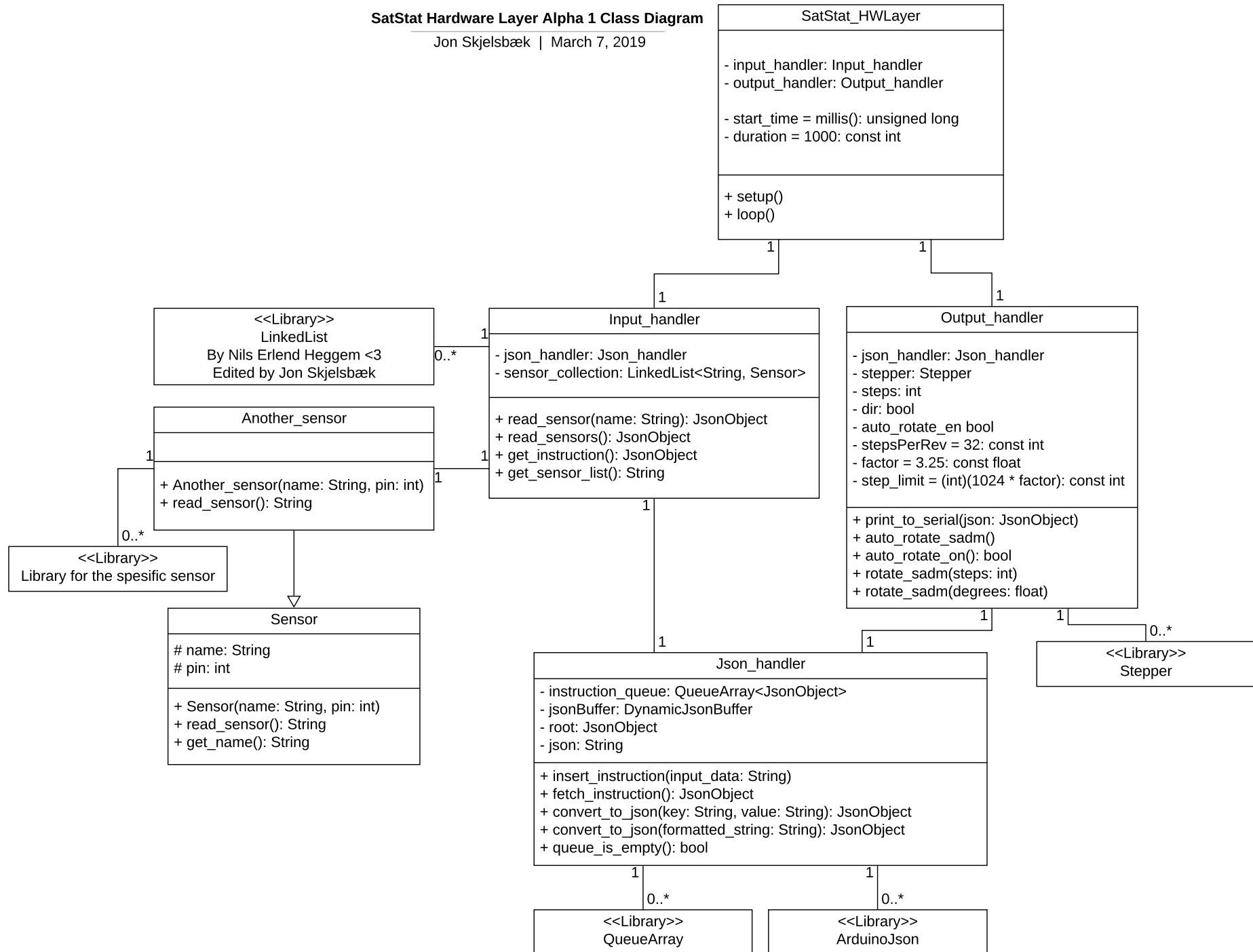
The main goal of this document is to describe the code visually in the form of a class diagram. This class diagram in particular is for version Alpha 1.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

SatStat Hardware Layer Alpha 1 Class Diagram

Jon Skjelsbæk | March 7, 2019



Bachelor of Engineering**Project appendix****Winter semester 2019****VR-HWL-A1**

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

Verification report for requirements related to the Hardware layer of the diagnostics system (SatStat), version Alpha 1.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

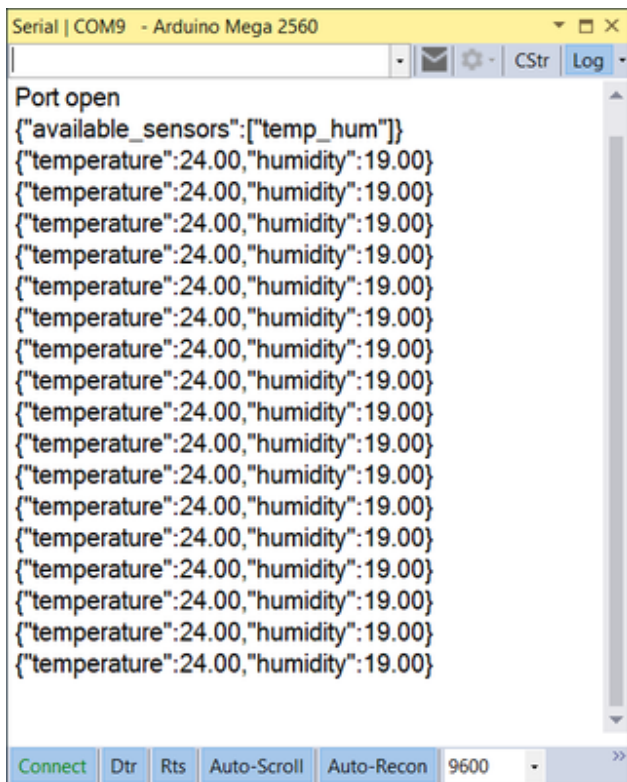
City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

VR-HWL-A1 07.03.2019

Date performed	07.03.2019
Performed by	Jon Skjelsbek
Tests performed (5)	TV-40, IV-08, IV-09, IV-10, TV-42
Considered requirements (2)	R4-17.02, R4-18.02
System version	Alpha 1

TV-40, TV-42

The image below shows a dump from the Serial Monitor with data received from the Arduino. The HWL is at this point expected to provide a list of available sensors (As it does at the second line) followed by continuous transmission of sensor reading which is displayed in the following lines.

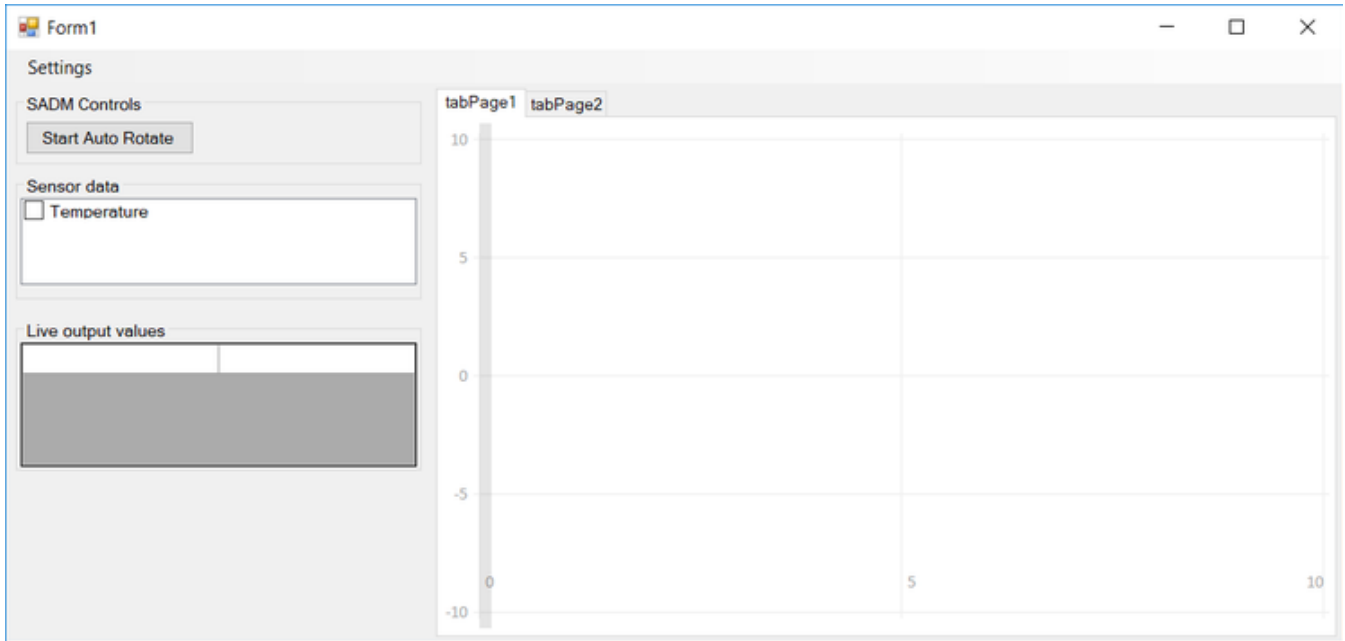


```

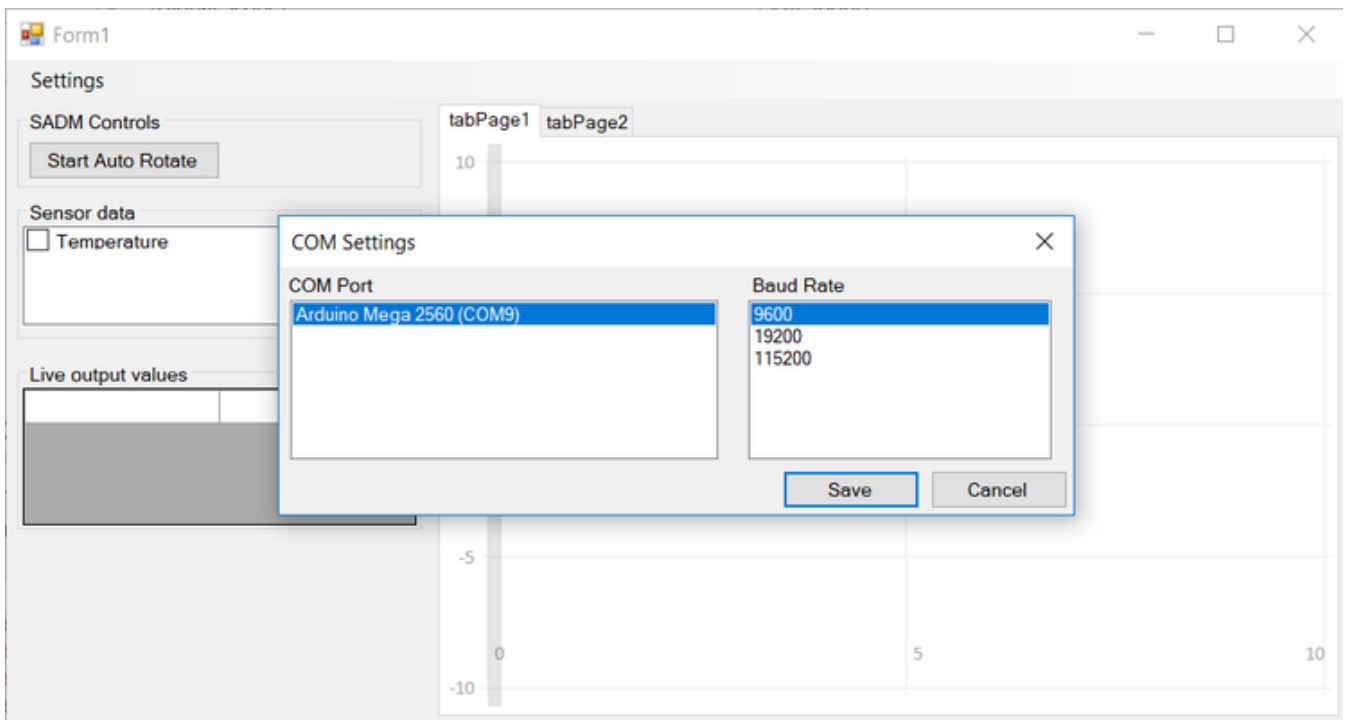
Serial | COM9 - Arduino Mega 2560
Port open
{"available_sensors":["temp_hum"]}
{"temperature":24.00,"humidity":19.00}
{"temperature":24.00,"humidity":19.00}
{"temperature":24.00,"humidity":19.00}
{"temperature":24.00,"humidity":19.00}
{"temperature":24.00,"humidity":19.00}
{"temperature":24.00,"humidity":19.00}
{"temperature":24.00,"humidity":19.00}
{"temperature":24.00,"humidity":19.00}
{"temperature":24.00,"humidity":19.00}
{"temperature":24.00,"humidity":19.00}
{"temperature":24.00,"humidity":19.00}
{"temperature":24.00,"humidity":19.00}
{"temperature":24.00,"humidity":19.00}
{"temperature":24.00,"humidity":19.00}
{"temperature":24.00,"humidity":19.00}
{"temperature":24.00,"humidity":19.00}
{"temperature":24.00,"humidity":19.00}
{"temperature":24.00,"humidity":19.00}

```

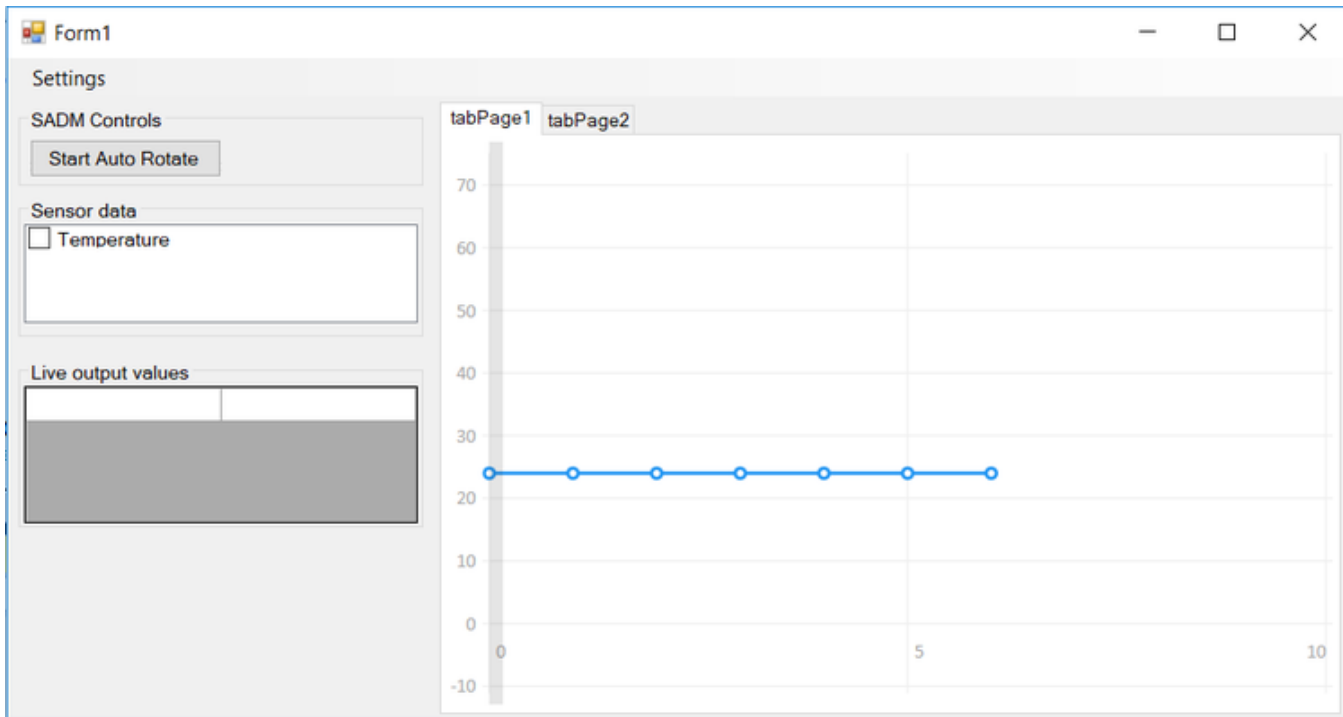
SWL before connecting:



SWL connection:



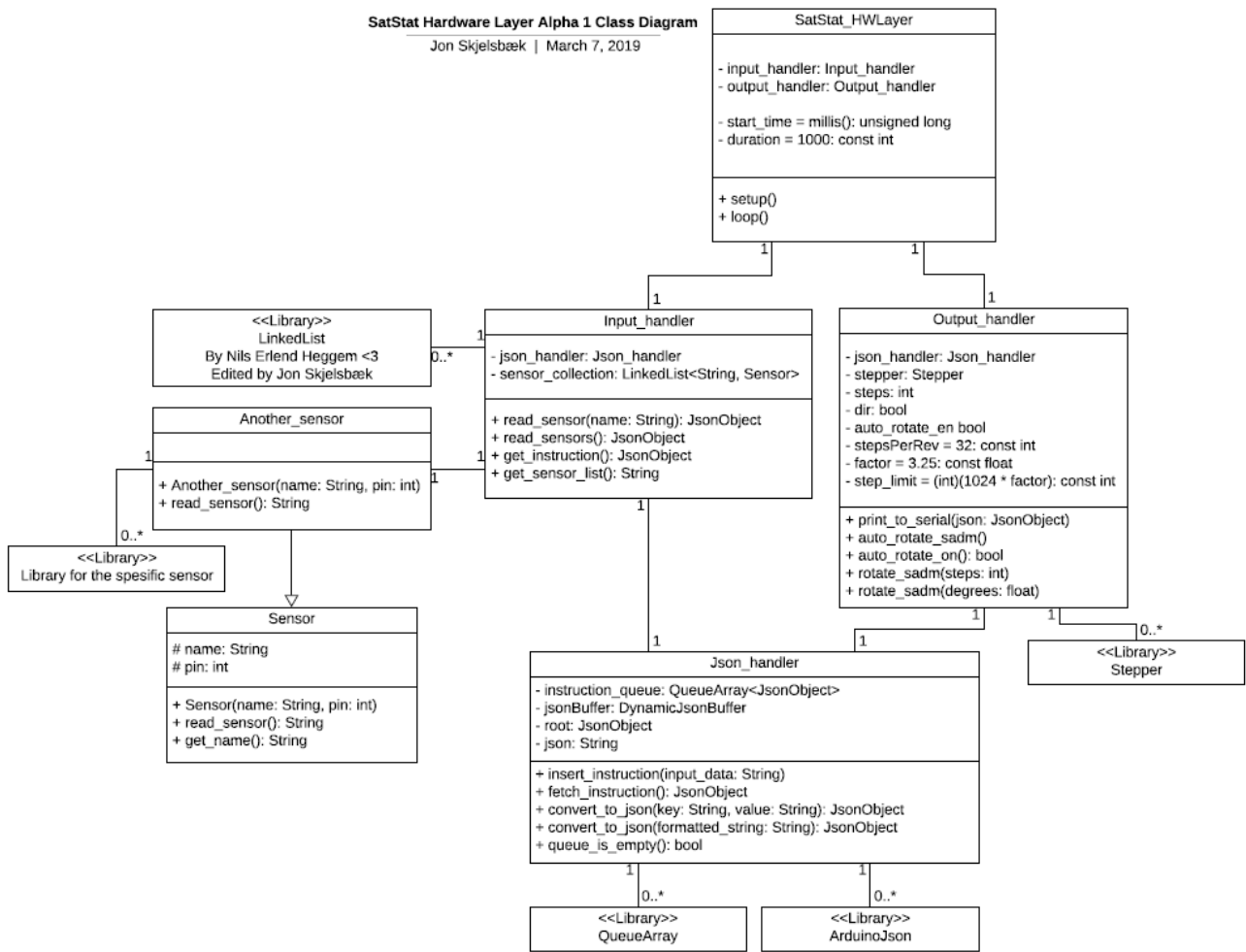
SWL after connection:



As you can see in the figures above, the software layer receives and visualizes the data received as expected, which means that it actually receives, and is able to understand what is sent from the HWL.

IV-08, IV-09, IV-10

IV-08, IV-09 and IV-10 requires the HWL to incorporate an input handler, output handler and a corresponding interface between these two. In the HWL Alpha 2 class diagram below, you can see that the HWL do indeed have the specified handlers, and the SatStat_HWLayer acts as an interface between these two.



Considered requirements

The requirements R4-17.02 and R4-18.02 was considered when transitioning from Alpha 1 to Alpha 2. They were removed as requirements R2-30 and R2-31, which already were defined, are basically the same requirements, but a more general general version at that.

Bachelor of Engineering

Project appendix

Winter semester 2019

SatStat Hardware Layer Alpha 2 - Requirements and verification

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

The hardware layer (HWL) of the Diagnostics System has two main responsibilities. One of these is to store test instructions received from the Software Layer (SWL), and translate these instructions to the proper format in order to execute them on the device corresponding to the initial instruction. The SWL also expect a continuous stream of sensor readings coming from the HWL. This requires the HWL to be able to read and transmit the prompted sensor data. This documents describes the requirements the HWL has to fulfill, and the various tests/verifications that has to be carried out in order to ensure that these requirements are met.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date	Signature
Kongsberg, May 20, 2019	

Diagnostics System - Hardware Layer version Alpha2

Target release	ALPHA 2
Epic	USN5-95 - Hardware layer of the Diagnostics System TO DO
Document status	V 1.0
Document owner	Jon Skjelsbek
Designer	Jon Skjelsbek
Tech lead	Jon Skjelsbek

Objective

The hardware layer(HWL) of the Diagnostics System has two main responsibilities. One of these is to store test instructions received from the Software Layer(SWL), and translate these instructions to the proper format in order to execute them on the device corresponding to the initial instruction. The SWL also expect a continuous stream of sensor readings coming from the HWL. This requires the HWL to be able to read and transmit the prompted sensor data. This documents describes the requirements the HWL has to fulfill, and the various tests/verifications that has to be carried out in order to ensure that these requirements are met.

Success metrics

Goal	Metric
Receive and store instructions from SWL.	Received instructions match the sent one.
Translate and execute fetched instructions.	The execution result replicate the initial instruction.
Provide prompted sensor data	SWL continuously receive the sensor data it asked for, and the corresponding values makes sense.

Roadmap

The HWL and SWL follow the same roadmap as when functionality is added to one layer, the other layer has to support it.

ID	Task Name	Start	Finish	Duration	10 feb 2019					17 feb 2019					24 feb 2019					3 mar 2019								
					11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	1	2	3	4	5	6
1	Design	11.02.2019	13.02.2019	3d	█																							
2	Implementation	13.02.2019	16.02.2019	4d						█																		
3	Verification	15.02.2019	17.02.2019	3d						█																		
4	Validation	16.02.2019	17.02.2019	2d						█																		
5	Alpha1	18.02.2019	18.02.2019	0d						◆																		
6	Design	19.02.2019	22.02.2019	4d						█																		
7	Implementation	21.02.2019	01.03.2019	9d											█													
8	Verification	23.02.2019	02.03.2019	8d											█													
9	Validation	24.02.2019	02.03.2019	7d											█													
10	Alpha2	03.03.2019	03.03.2019	0d																◆								
11	Design	01.03.2019	06.03.2019	6d											█													
12	Implementation	04.03.2019	10.03.2019	7d																█								
13	Verification	07.03.2019	10.03.2019	4d																█								
14	Validation	09.03.2019	10.03.2019	2d																█								
15	Alpha3	11.03.2019	11.03.2019	0d																◆								

Requirements

ID	Description	Importance	Version
R2-30	The DS Hardware layer must be able to control the SADM for testing purposes.	HIGH	ALPHA 2
Date	Discussion and decisions		
06.02.2019			
User Story			
<div style="border: 1px solid #ccc; padding: 5px;"> <p>USN5-96 - As an embedded developer, I need my hardware to be able to control the SADM to be able to measure the result.</p> <p>DONE</p> </div>			
Verification ID	Verification method	Status	
TV-39	Functional test: Run as many instructions as see fit, and confirm that the desired result is achieved.	PASSED	

ID	Description	Importance	Version
R2-31	The DS Hardware layer must be able to transmit sensor data to the Software layer for comparison and visualization purposes.	HIGH	ALPHA 1
Date	Discussion and decisions		
06.02.2019			
User Story			

USN5-97 - As an embedded developer, I need to provide sensible sensor data to verify that the systems behaves as expected.

DONE

Verification ID	Verification method	Status
TV-40	Functional test: Transmit as much sensor data as see fit, and confirm that the receiver is able to properly read the data.	PASSED

ID	Description	Importance	Version
R2-36	The DS Hardware layer shall be able to interpret and execute instructions received from the software layer.	HIGH	ALPHA 2

Date	Discussion and decisions
------	--------------------------

22.02.2019

User Story

USN5-131 - As a test engineer, I need the ability to calibrate position and speed of the SADM drive shaft so that I can set correct initial conditions for operation **DONE**

Verification ID	Verification method	Status
TV-44	Functional test: Run as many instructions as see fit, and confirm that the desired result is achieved.	PASSED

ID	Description	Importance	Version
R4-17	DS Hardware layer shall have a standardized expandable input handler, buffering input data according to input type. Rev. A) DS Hardware layer shall have a general input handler responsible for every kind of input.	HIGH	ALPHA 2

Date	Discussion and decisions
------	--------------------------

~~06.02.19~~

Rev. A) 21.02.19

Discarded for Alpha 2 as the input and output handler has been replaced by the serial handler.

User Story

USN5-98 - As an embedded developer, I need to make sure that instructions can be received even though another instruction is currently executing. **DONE**

Verification ID	Verification method	Status
IV-08	Confirm that the code replicates the class diagrams and follow related standards.	DISCARDED

ID	Description	Importance	Version
R4-18	DS Hardware layer shall have a standardized expandable output handler translating the input data passed from a specific controller, and output the translated information to the device to be controlled. Rev. A) DS Hardware layer shall have a general output handler supporting the different protocols/formats needed in order to communicate with the different connected devices.	HIGH	ALPHA 2

Date	Discussion and decisions
06.02.19 Rev. A) 21.02.19	Discarded for Alpha 2 as the input and output handler has been replaced by the serial handler.

User Story

USN5-99 - As an embedded developer, I need to adapt the instructions to the proper format for the receiving device to understand. **DONE**

Verification ID	Verification method	Status
IV-09	Confirm that the code replicates the class diagrams and follow related standards.	DISCARDED

ID	Description	Importance	Version
R4-19	There shall be an interface between the input handler and the output handler (controller). Preferably one for each device to be controlled. Rev. A) There shall be an interface responsible for fetching from the input handler and passing to the output handler.	HIGH	ALPHA 2

Date	Discussion and decisions

06.02.19 Rev. A) 21.02.19	Discarded for Alpha 2 as the input and output handler has been replaced by the serial handler.		
User Story			
USN5-100 - As an embedded developer, I need to make sure that the correct instructions are executed at the correct device for the test results to be correct. DONE			
Verification ID	Verification method	Status	
IV-10	Confirm that the code replicates the class diagrams and follow related standards.	DISCARDED	

ID	Description	Importance	Version
R4-17.01	The input handler of the DS Hardware layer must be compatible with the software layer of the diagnostic system. Rev. A) The input handler must support JSON input received from the software layer. Rev. B) The HWL must support JSON input received from the software layer.	HIGH	ALPHA 2
Date	Discussion and decisions		
06.02.19 Rev. A) JS 21.02.19 Rev. B) JS 07.03.19	The input and output handler has been replaced by the serial handler in Alpha 2.		
User Story			
USN5-101 - As an embedded developer, I need the receiving end of my hardware to speak the same language as the software layer in order to properly interpret the received instructions. DONE			
Verification ID	Verification method	Status	
TV-41	Functional test: Send as many instructions as see fit from the software layer, and confirm that the data received is properly handled.	PASSED	

ID	Description	Importance	Version
----	-------------	------------	---------

R4-18.01	The output handler of the DS Hardware layer shall support the different protocols needed in order to communicate with the different connected devices. Rev. A) The output handler shall incorporate serial printing of JSON formatted data. Rev. B) The HWL shall incorporate serial printing of JSON formatted data.	HIGH	ALPHA 2
----------	---	-------------	----------------

Date	Discussion and decisions
06.02.19 Rev. A) JS 21.02.19 Rev. B) JS 07.03.19	The input and output handler has been replaced by the serial handler in Alpha 2.

User Story

USN5-102 - As an embedded developer, I need the transmitting end of my hardware to speak languages that the receiving devices understand in order to transmit data in the correct format. **DONE**

Verification ID	Verification method	Status
TV-42	Confirm that the code replicates the class diagrams and related standards. Rev. A) Check that the software layer receives the data as expected.	PASSED

ID	Description	Importance	Version
R4-25	Shall follow a unified communication protocol.	HIGH	ALPHA 2

Date	Discussion and decisions
04.03.2019	

User Story

USN5-226 - As a computer engineer, I need to define a communication protocol so that both hardware- and software-layer can communicate correctly **DONE**

Verification ID	Verification method	Status
TV-43	Functional test: System behaves as expected and follows defined protocol	PASSED

ID	Description	Importance	Version
R5-22	The Diagnostics Systems shall not interfere with the SADM.	MEDIUM	PENDING
Date	Discussion and decisions		
06.02.2019			
User Story			
USN5-103 - As an embedded developer, I need to make sure my hardware doesn't affect the SADM as they are two separate systems.			
<input type="button" value="OPEN"/>			
Verification ID	Verification method	Status	
IV-12	Confirm that the DS and SADM is indeed two separate systems.	PENDING	

ID	Description	Importance	Version
R7-06	The DS HWL must incorporate timeouts in the handshake protocol.	MEDIUM	ALPHA 2
Date	Discussion and decisions		
06.03.19			
User Story			
USN5-226 - As a computer engineer, I need to define a communication protocol so that both hardware- and software-layer can communicate correctly			
<input type="button" value="DONE"/>			
Verification ID	Verification method	Status	
TV-45	Functional test: Force timeout and see that it works properly.	PASSED	

ID	Description	Importance	Version
----	-------------	------------	---------

R7-07	The DS HWL must have functionality for Negative Acknowledgement (NACK) in case of error in communication.	MEDIUM	ALPHA 2
Date	Discussion and decisions		
06.03.19			
User Story			
USN5-226 - As a computer engineer, I need to define a communication protocol so that both hardware- and software-layer can communicate correctly DONE			
Verification ID	Verification method	Status	
TV-46	Functional test: Force timeout and send faulty instructions to verify that a nack is sent.	PASSED	

Links to external sources

Alpha 1 class diagram: <https://goo.gl/r4Pfg>

Alpha 2 class diagram: <https://goo.gl/uWCcxU>

SatStat communication protocol: <https://usnbachelor.atlassian.net/wiki/x/S4AEBQ>

Bachelor of Engineering

Project appendix

Winter semester 2019

SatStat Hardware Layer Alpha 2 and 3 - Class Diagram

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

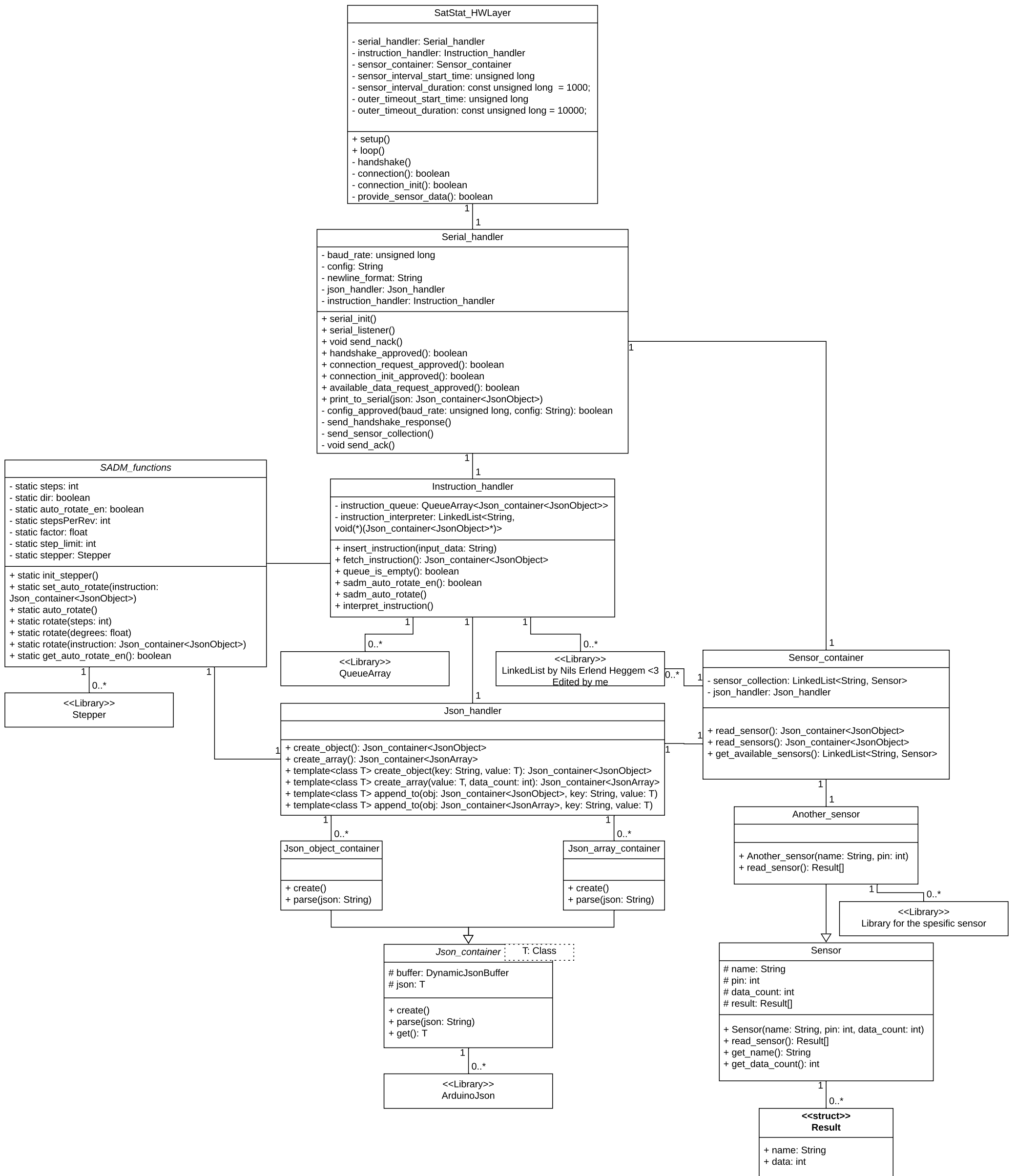
The main goal of this document is to describe the code visually in the form of a class diagram. This class diagram in particular is for versions Alpha 2 and 3.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

SatStat Hardware Layer Alpha 2 and 3 Class Diagram

Jon Skjelsbæk | May 13, 2019



Bachelor of Engineering**Project appendix****Winter semester 2019****VR-HWL-A2**

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

Verification report for requirements related to the Hardware layer of the diagnostics system (SatStat), version Alpha 2.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

VR-HWL-A2 07.03.2019

Date performed	07.03.2019
Performed by	Jon Skjelsbek
Tests performed (6)	TV-39 , TV-41 , TV-42 , TV-43 , TV-44 , TV-45 , TV-46
Tests discarded (3)	IV-08 , IV-09 , IV-10
System version	Alpha 2

Discarded tests

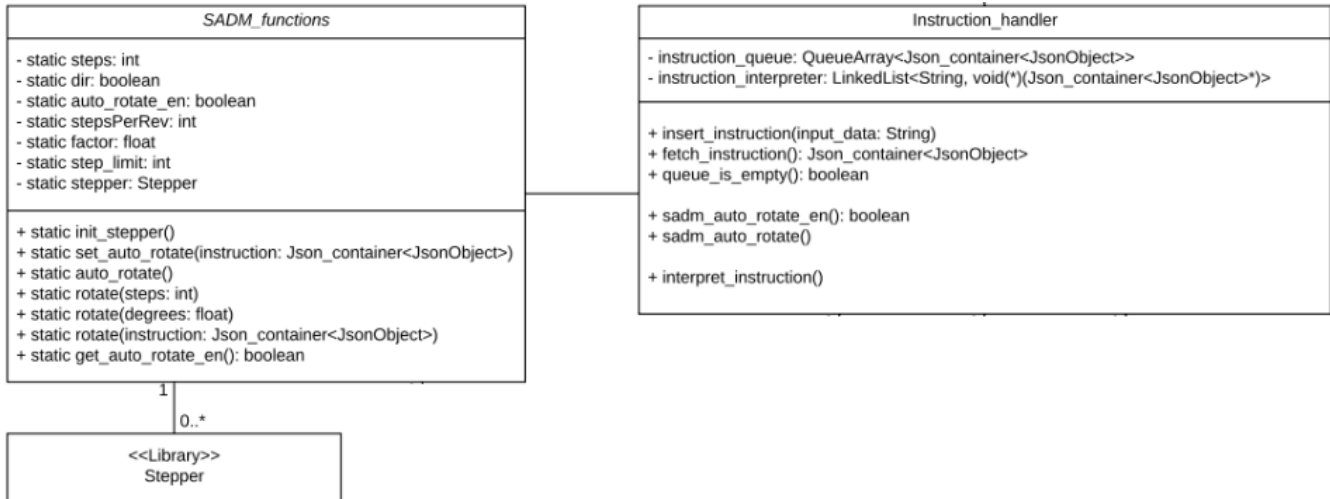
IV-08, IV-09 and IV-10 were verifying the requirements related to the HWL having separate handlers for input and output and a interface connecting these two. For Alpha 2, the input- and output handler has been merged into a serial handler as it's proven redundant having one for each purpose. TV-41 and TV-42, which are discussed at a later point in this report, verifies that the system does indeed handle input and output according to standards for Alpha 2 as well, even though the input- and output handler has been merged.

TV-39 and TV-44 Instruction handling and controlling the SADM

The figure below is an excerpt from the [Alpha 2 class diagram](#) that shows the "Instruction_handler"- and the abstract "SADM_functions" class. HWL Alpha 2 has to be compliant with the stepper motor used for the Mark 2 prototype, which is why "SADM_functions" utilize the "Stepper" library. Functional tests has been performed by sending instructions from SWL, and observing the outcome/result when executed at HWL. When an instruction was received, the "interpret_instruction" function found the corresponding SADM function in the "instruction_interpreter" list as expected. The function was executed, and the SADM was rotated according to the instruction sent.

Functional tests has been done to all the functions in the "SADM_functions" class, and the observations prove that the HWL does indeed handle instructions properly, and are able to control the SADM according to instructions received from SWL.





TV-41, TV-42 and TV-43 JSON formatted serial I/O

TV-41 verifies that the HWL is able to receive and interpret JSON formatted serial input from the SWL. As TV-39 and TV-44 prove that the HWL is able to do just that, TV-41 is also considered passed.

TV-42 was conducted by checking if sensor data was properly sent from the HWL to the SWL by observing the outputs in both the serial monitor, and the SWL visualization. The figures below show the test results, and proves that the format of the transmitted data are as defined in the [Sat Stat communication protocol](#) document, and that the SWL is able to interpret the received data.

TV-43 concerns the communication protocol in general, where as TV-41 and TV-42 concerns the input and output parts separately. Since this is the case, TV-43 is considered passed as both TV-41 and TV-42 are.

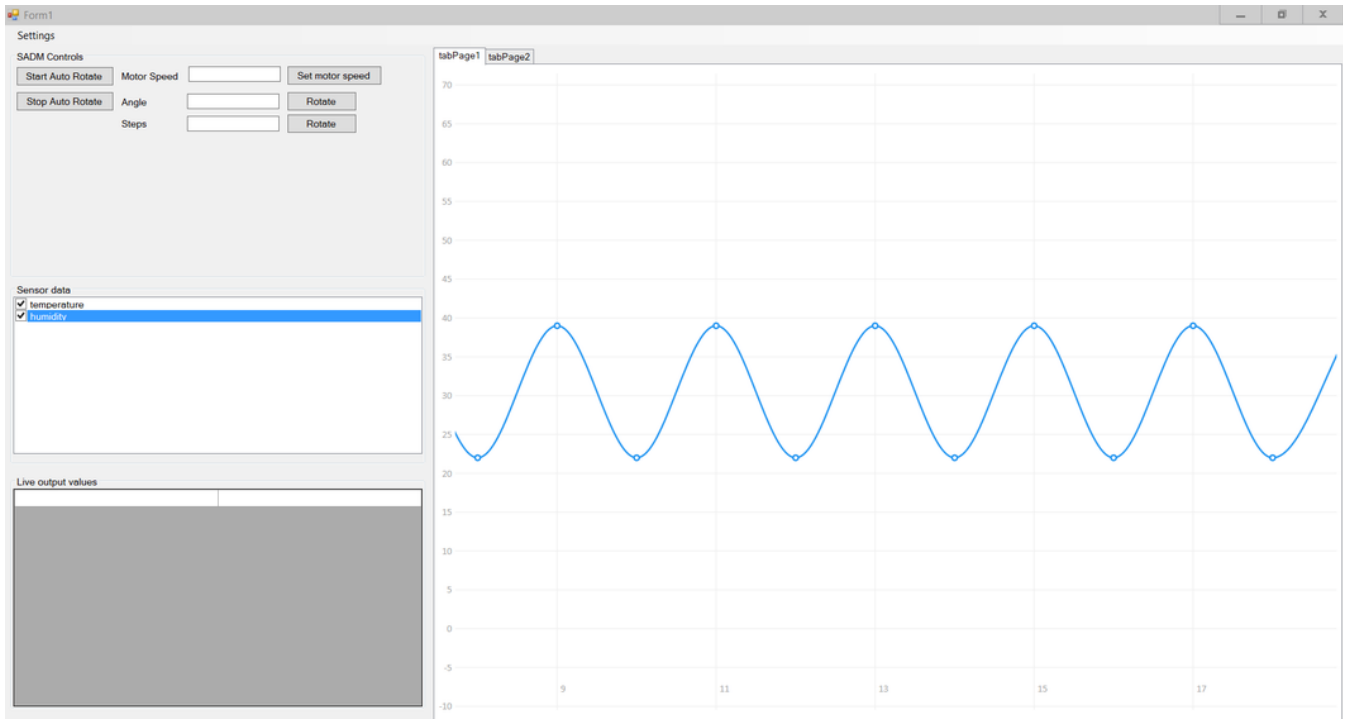
Serial monitor screen dump (Temperature and humidity are consistently transmitted after connection is established):

```

Serial | COM11 - Arduino Mega 2560
request:"available_data"
Opening port
Port open
{"serial_handshake":{"baud_rates":[9600,19200,38400,14400,28800,57600,115200],"configs":
["5N1","5N2","5O1","5O2","5E1","5E2","6N1","6N2","6O1","6O2","6E1","6E2","7N1","7N2","7O1","7O2","7E1","7E2","8N1","8N2","8O1","8O2","8E1","8E2"],"newline":["\n","\n"]}}
{"connect":"init"}
{"connect":"init"}
{"connect":"init"}
{"connect":"init"}
{"available_data":{"temperature":"int","humidity":"int"}}
{"temperature":22,"humidity":39}
{"temperature":22,"humidity":39}
{"temperature":22,"humidity":39}
{"temperature":22,"humidity":39}
{"temperature":22,"humidity":39}
{"temperature":22,"humidity":39}
{"temperature":22,"humidity":39}
{"temperature":22,"humidity":39}
{"temperature":22,"humidity":39}
{"temperature":22,"humidity":39}
{"temperature":22,"humidity":39}
{"temperature":22,"humidity":39}
{"temperature":22,"humidity":39}
{"temperature":22,"humidity":39}

```

SWL visualization of data received from HWL (Temperature and humidity are both displayed in the same graph. Values a bit above 20 are temperature, and values close to 40 are humidity):



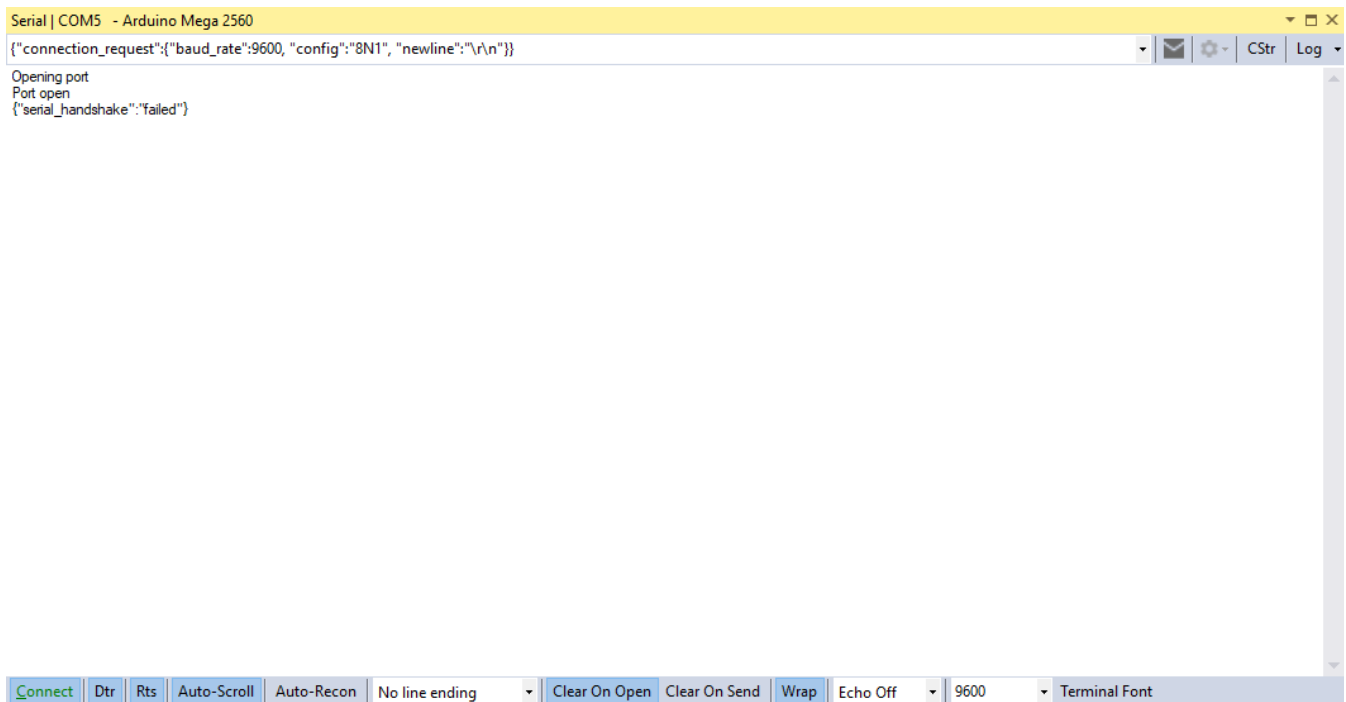
TV-45 and TV-46 Timeouts and negative acknowledgement

The [SatStat communication protocol](#) document defines four steps in the handshake protocol, and what's expected to be received on HWL for each step is:

1. Serial handshake request.
2. Connection request.
3. Connection acknowledgement with new configuration.
4. Available data request.

If an unexpected message, or a timeout occur for any of these steps, the whole process has to start over in order to establish the connection properly. The figures below show screen dumps of the serial monitor where each of these steps are tested sequentially. The first figure prove that when an unexpected message is received, a NACK is sent. The following figures show that the timeout system do work for every single one of the previously mentioned steps in the handshake protocol. The last figure show a proper execution of the handshake protocol.

NACK sent when wrong message received:



The screenshot shows a serial monitor window titled "Serial | COM5 - Arduino Mega 2560". The main area displays the following text:

```
["connection_request":{"baud_rate":9600, "config":"8N1", "newline":"\r\n"}]  
Opening port  
Port open  
{"serial_handshake":"failed"}
```

The bottom of the window features a control bar with various settings: Connect, Dtr, Rts, Auto-Scroll, Auto-Recon, No line ending, Clear On Open, Clear On Send, Wrap, Echo Off, 9600, and Terminal Font.

NACK sent after timeout waiting for connection request:

```
Serial | COM5 - Arduino Mega 2560
{"serial_handshake":"init"}
Opening port
Port open
{"serial_handshake":"failed"}
{"serial_handshake":{"baud_rates":[9600,19200,38400,14400,28800,57600,115200],"configs":
["5N1","5N2","5O1","5O2","5E1","5E2","6N1","6N2","6O1","6O2","6E1","6E2","7N1","7N2","7O1","7O2","7E1","7E2","8N1","8N2","8O1","8O2","8E1","8E2"],"newlines":["\r\n","\n"]}}
{"serial_handshake":"failed"}
```

NACK sent after timeout waiting for connection response:

```
Serial | COM5 - Arduino Mega 2560
{"connection_request":{"baud_rate":9600,"config":"8N1","newline":"\r\n"}}
Opening port
Port open
{"serial_handshake":"failed"}
{"serial_handshake":{"baud_rates":[9600,19200,38400,14400,28800,57600,115200],"configs":
["5N1","5N2","5O1","5O2","5E1","5E2","6N1","6N2","6O1","6O2","6E1","6E2","7N1","7N2","7O1","7O2","7E1","7E2","8N1","8N2","8O1","8O2","8E1","8E2"],"newlines":["\r\n","\n"]}}
{"serial_handshake":"failed"}
{"serial_handshake":{"baud_rates":[9600,19200,38400,14400,28800,57600,115200],"configs":
["5N1","5N2","5O1","5O2","5E1","5E2","6N1","6N2","6O1","6O2","6E1","6E2","7N1","7N2","7O1","7O2","7E1","7E2","8N1","8N2","8O1","8O2","8E1","8E2"],"newlines":["\r\n","\n"]}}
{"connect":"init"}
{"connect":"init"}
{"connect":"init"}
{"connect":"init"}
{"connect":"init"}
{"connect":"init"}
{"serial_handshake":"failed"}⌘
```

NACK sent after timeout waiting for available data request:

```
Serial | COM5 - Arduino Mega 2560
{"connect": "ok"}
Opening port
Port open
{"serial_handshake": "failed"}
{"serial_handshake": {"baud_rates": [9600, 19200, 38400, 14400, 28800, 57600, 115200], "configs": [{"5N1": "5N2", "5O1": "5O2", "5E1": "5E2", "6N1": "6N2", "6O1": "6O2", "6E1": "6E2", "7N1": "7N2", "7O1": "7O2", "7E1": "7E2", "8N1": "8N2", "8O1": "8O2", "8E1": "8E2"], "newlines": [{"\r\n", "\n"}]}}
{"serial_handshake": "failed"}
{"serial_handshake": {"baud_rates": [9600, 19200, 38400, 14400, 28800, 57600, 115200], "configs": [{"5N1": "5N2", "5O1": "5O2", "5E1": "5E2", "6N1": "6N2", "6O1": "6O2", "6E1": "6E2", "7N1": "7N2", "7O1": "7O2", "7E1": "7E2", "8N1": "8N2", "8O1": "8O2", "8E1": "8E2"], "newlines": [{"\r\n", "\n"}]}}
{"connect": "int"}
{"connect": "int"}
{"connect": "int"}
{"connect": "int"}
{"connect": "int"}
{"serial_handshake": "failed"}
{"serial_handshake": {"baud_rates": [9600, 19200, 38400, 14400, 28800, 57600, 115200], "configs": [{"5N1": "5N2", "5O1": "5O2", "5E1": "5E2", "6N1": "6N2", "6O1": "6O2", "6E1": "6E2", "7N1": "7N2", "7O1": "7O2", "7E1": "7E2", "8N1": "8N2", "8O1": "8O2", "8E1": "8E2"], "newlines": [{"\r\n", "\n"}]}}
{"connect": "int"}
{"connect": "int"}
{"serial_handshake": "failed"}

```

Connection properly established:

```
Serial | COM5 - Arduino Mega 2560
{"request": "available_data"}
Opening port
Port open
{"serial_handshake": "failed"}
{"serial_handshake": {"baud_rates": [9600, 19200, 38400, 14400, 28800, 57600, 115200], "configs": [{"5N1": "5N2", "5O1": "5O2", "5E1": "5E2", "6N1": "6N2", "6O1": "6O2", "6E1": "6E2", "7N1": "7N2", "7O1": "7O2", "7E1": "7E2", "8N1": "8N2", "8O1": "8O2", "8E1": "8E2"], "newlines": [{"\r\n", "\n"}]}}
{"serial_handshake": "failed"}
{"serial_handshake": {"baud_rates": [9600, 19200, 38400, 14400, 28800, 57600, 115200], "configs": [{"5N1": "5N2", "5O1": "5O2", "5E1": "5E2", "6N1": "6N2", "6O1": "6O2", "6E1": "6E2", "7N1": "7N2", "7O1": "7O2", "7E1": "7E2", "8N1": "8N2", "8O1": "8O2", "8E1": "8E2"], "newlines": [{"\r\n", "\n"}]}}
{"connect": "int"}
{"connect": "int"}
{"connect": "int"}
{"connect": "int"}
{"connect": "int"}
{"serial_handshake": "failed"}
{"serial_handshake": {"baud_rates": [9600, 19200, 38400, 14400, 28800, 57600, 115200], "configs": [{"5N1": "5N2", "5O1": "5O2", "5E1": "5E2", "6N1": "6N2", "6O1": "6O2", "6E1": "6E2", "7N1": "7N2", "7O1": "7O2", "7E1": "7E2", "8N1": "8N2", "8O1": "8O2", "8E1": "8E2"], "newlines": [{"\r\n", "\n"}]}}
{"connect": "int"}
{"serial_handshake": "failed"}
{"serial_handshake": {"baud_rates": [9600, 19200, 38400, 14400, 28800, 57600, 115200], "configs": [{"5N1": "5N2", "5O1": "5O2", "5E1": "5E2", "6N1": "6N2", "6O1": "6O2", "6E1": "6E2", "7N1": "7N2", "7O1": "7O2", "7E1": "7E2", "8N1": "8N2", "8O1": "8O2", "8E1": "8E2"], "newlines": [{"\r\n", "\n"}]}}
{"connect": "int"}
{"connect": "int"}
{"available_data": {"temperature": "int", "humidity": "int"}}
{"temperature": 0, "humidity": 0}
{"temperature": 22, "humidity": 36}
{"temperature": 22, "humidity": 36}
{"temperature": 22, "humidity": 36}
{"temperature": 22, "humidity": 36}
{"temperature": 23, "humidity": 36}
{"temperature": 23, "humidity": 36}
{"temperature": 23, "humidity": 36}
{"temperature": 23, "humidity": 36}
{"temperature": 23, "humidity": 36}
{"temperature": 23, "humidity": 36}
{"temperature": 23, "humidity": 36}
{"temperature": 23, "humidity": 36}
{"temperature": 23, "humidity": 36}
{"temperature": 23, "humidity": 36}

```

Bachelor of Engineering

Project appendix

Winter semester 2019

SatStat Hardware Layer Alpha 3 - Requirements and verification

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

The hardware layer (HWL) of the Diagnostics System has two main responsibilities. One of these is to store test instructions received from the Software Layer (SWL), and translate these instructions to the proper format in order to execute them on the device corresponding to the initial instruction. The SWL also expect a continuous stream of sensor readings coming from the HWL. This requires the HWL to be able to read and transmit the prompted sensor data. This documents describes the requirements the HWL has to fulfill, and the various tests/verifications that has to be carried out in order to ensure that these requirements are met.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date	Signature
Kongsberg, May 20, 2019	

Diagnostics System - Hardware Layer version Alpha3

Target release	ALPHA 3
Epic	USN5-95 - Hardware layer of the Diagnostics System IN PROGRESS
Document status	V 1.0
Document owner	Jon Skjelsbek
Designer	Jon Skjelsbek
Tech lead	Jon Skjelsbek

Objective

The hardware layer(HWL) of the Diagnostics System has two main responsibilities. One of these is to store test instructions received from the Software Layer(SWL), and translate these instructions to the proper format in order to execute them on the device corresponding to the initial instruction. The SWL also expect a continuous stream of sensor readings coming from the HWL. This requires the HWL to be able to read and transmit the prompted sensor data. This documents describes the requirements the HWL has to fulfill, and the various tests/verifications that has to be carried out in order to ensure that these requirements are met.

Success metrics

Goal	Metric
Receive and store instructions from SWL.	Received instructions match the sent one.
Translate and execute fetched instructions.	The execution result replicate the initial instruction.
Provide prompted sensor data	SWL continuously receive the sensor data it asked for, and the corresponding values makes sense.

Roadmap

The HWL and SWL follow the same roadmap as when functionality is added to one layer, the other layer has to support it.

ID	Task Name	Start	Finish	Duration	10 feb 2019					17 feb 2019					24 feb 2019					3 mar 2019								
					11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	1	2	3	4	5	6
1	Design	11.02.2019	13.02.2019	3d	█																							
2	Implementation	13.02.2019	16.02.2019	4d						█																		
3	Verification	15.02.2019	17.02.2019	3d						█																		
4	Validation	16.02.2019	17.02.2019	2d						█																		
5	Alpha1	18.02.2019	18.02.2019	0d						◆																		
6	Design	19.02.2019	22.02.2019	4d						█																		
7	Implementation	21.02.2019	01.03.2019	9d											█													
8	Verification	23.02.2019	02.03.2019	8d											█													
9	Validation	24.02.2019	02.03.2019	7d											█													
10	Alpha2	03.03.2019	03.03.2019	0d																◆								
11	Design	01.03.2019	06.03.2019	6d											█													
12	Implementation	04.03.2019	10.03.2019	7d																█								
13	Verification	07.03.2019	10.03.2019	4d																█								
14	Validation	09.03.2019	10.03.2019	2d																█								
15	Alpha3	11.03.2019	11.03.2019	0d																◆								

Requirements

ID	Description	Importance	Version
R7-08	The DS HWL must incorporate error handling where it's needed.	HIGH	ALPHA 3
Date	Discussion and decisions		
17.03.2019			
User Story			
<p>USN5-257 - As the HWL has to function at all times, error handling must be implemented to ensure that the system doesn't crash.</p> <p>DONE</p>			
Verification ID	Verification method	Status	
TV-48	Functional test: Provoke errors to check if it is properly handled.	PASSED	

Bachelor of Engineering**Project appendix****Winter semester 2019****VR-HWL-A3**

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

Verification report for requirements related to the Hardware layer of the diagnostics system (SatStat), version Alpha 3.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

VR-HWL-A3

Date performed	15.04.2019
Performed by	Jon Skjelsbek
Tests performed (1)	TV-48
Tests discarded (0)	
System version	Alpha 3

Tests

TV-48

This test checks if the proper exception handling routines are executed when receiving faulty requests from SWL.

Parsing error:

Receiving data that's not JSON formatted will prompt a parsing error message.

```
Serial | COM11 - Arduino Mega 2560
Invalid JSON format
Opening port
Port open
{"Error": "Could not parse received data!"}
```

Errors provoked in handshake protocol:

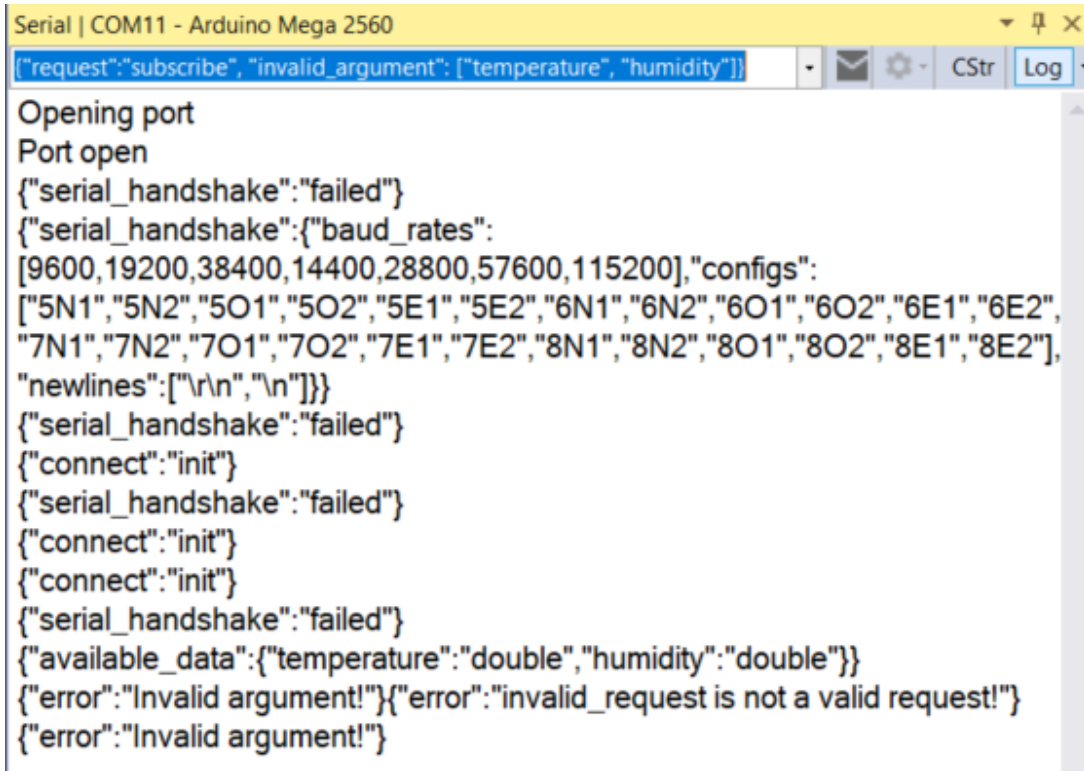
The figure below show that errors have been provoked for every stage in the handshake protocol, and an error message has been provided.

```
Serial | COM11 - Arduino Mega 2560
{"request":"available_data"}
Opening port
Port open
{"serial_handshake":"failed"}
{"serial_handshake":{"baud_rates":
[9600,19200,38400,14400,28800,57600,115200],"configs":
["5N1","5N2","5O1","5O2","5E1","5E2","6N1","6N2","6O1","6O2","6E1","6E2",
"7N1","7N2","7O1","7O2","7E1","7E2","8N1","8N2","8O1","8O2","8E1","8E2"],
"newlines":["\r\n","\n"]}}
{"serial_handshake":"failed"}
{"connect":"init"}
{"serial_handshake":"failed"}
{"connect":"init"}
{"connect":"init"}
{"serial_handshake":"failed"}
{"available_data":{"temperature":"double","humidity":"double"}}
```

Invalid argument:

If the system is unable to interpret the first, or any of the consecutive arguments, an Invalid argument error is prompted.

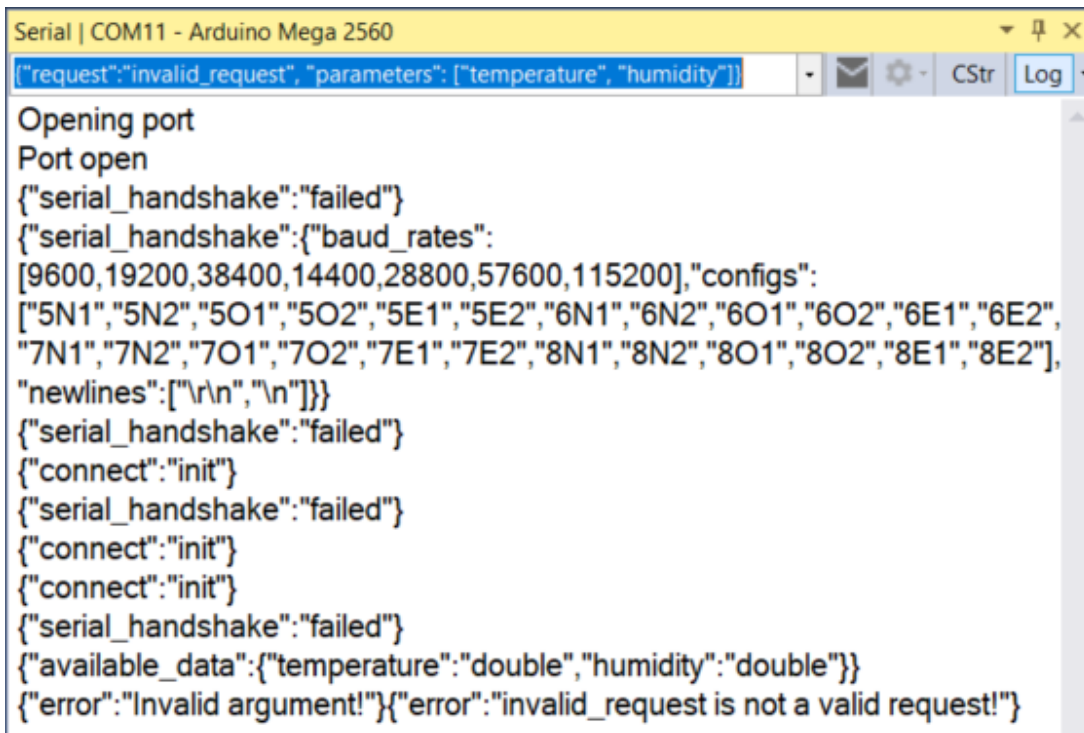
```
Serial | COM11 - Arduino Mega 2560
{"not_a_request":"subscribe","parameters":["temperature","humidity"]}
Opening port
Port open
{"serial_handshake":"failed"}
{"serial_handshake":{"baud_rates":
[9600,19200,38400,14400,28800,57600,115200],"configs":
["5N1","5N2","5O1","5O2","5E1","5E2","6N1","6N2","6O1","6O2","6E1","6E2",
"7N1","7N2","7O1","7O2","7E1","7E2","8N1","8N2","8O1","8O2","8E1","8E2"],
"newlines":["\r\n","\n"]}}
{"serial_handshake":"failed"}
{"connect":"init"}
{"serial_handshake":"failed"}
{"connect":"init"}
{"connect":"init"}
{"serial_handshake":"failed"}
{"available_data":{"temperature":"double","humidity":"double"}}
{"error":"Invalid argument!"}
```



```
Serial | COM11 - Arduino Mega 2560
["request":"subscribe", "invalid_argument": ["temperature", "humidity"]]
Opening port
Port open
{"serial_handshake":"failed"}
{"serial_handshake":{"baud_rates":
[9600,19200,38400,14400,28800,57600,115200],"configs":
["5N1","5N2","5O1","5O2","5E1","5E2","6N1","6N2","6O1","6O2","6E1","6E2",
"7N1","7N2","7O1","7O2","7E1","7E2","8N1","8N2","8O1","8O2","8E1","8E2"],
"newlines":["\r\n","\n"]}}
{"serial_handshake":"failed"}
{"connect":"init"}
{"serial_handshake":"failed"}
{"connect":"init"}
{"connect":"init"}
{"serial_handshake":"failed"}
{"available_data":{"temperature":"double","humidity":"double"}}
{"error":"Invalid argument!"}{"error":"invalid_request is not a valid request!"}
{"error":"Invalid argument!"}
```

Invalid request:

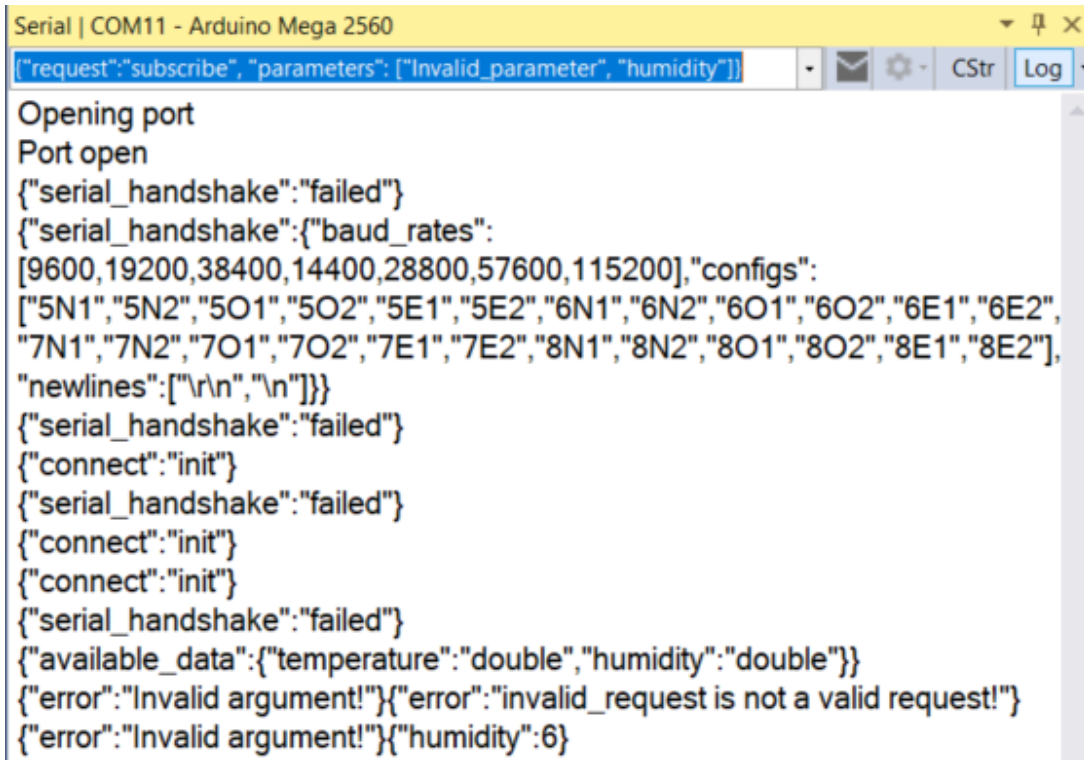
When an invalid request is received, an error message stating the cause of the error will be sent.



```
Serial | COM11 - Arduino Mega 2560
["request":"invalid_request", "parameters": ["temperature", "humidity"]]
Opening port
Port open
{"serial_handshake":"failed"}
{"serial_handshake":{"baud_rates":
[9600,19200,38400,14400,28800,57600,115200],"configs":
["5N1","5N2","5O1","5O2","5E1","5E2","6N1","6N2","6O1","6O2","6E1","6E2",
"7N1","7N2","7O1","7O2","7E1","7E2","8N1","8N2","8O1","8O2","8E1","8E2"],
"newlines":["\r\n","\n"]}}
{"serial_handshake":"failed"}
{"connect":"init"}
{"serial_handshake":"failed"}
{"connect":"init"}
{"connect":"init"}
{"serial_handshake":"failed"}
{"available_data":{"temperature":"double","humidity":"double"}}
{"error":"Invalid argument!"}{"error":"invalid_request is not a valid request!"}
```

Invalid parameter:

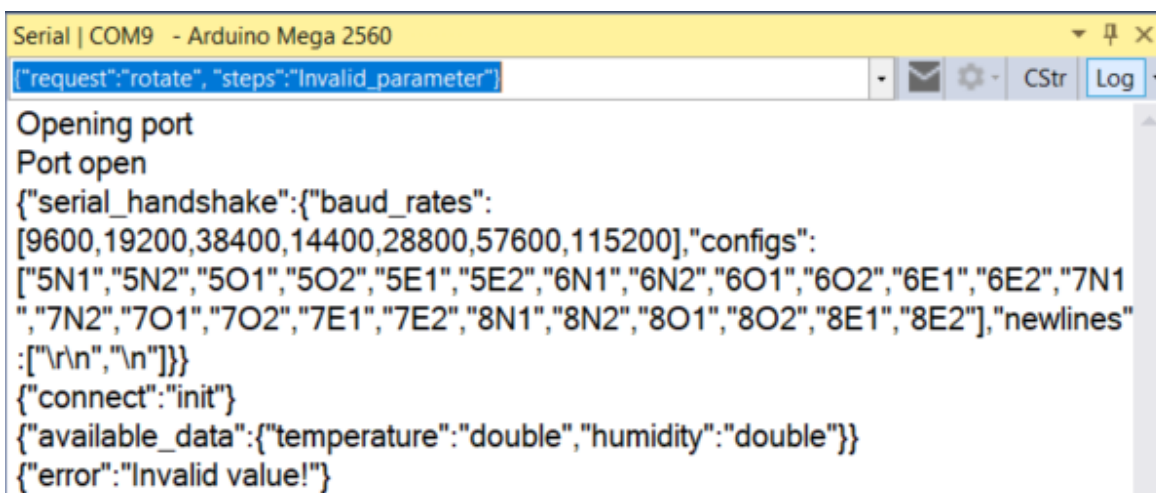
Invalid parameters will be ignored, and measures have been taken to prevent this from affecting consecutive arguments.



```
Serial | COM11 - Arduino Mega 2560
{"request": "subscribe", "parameters": ["Invalid_parameter", "humidity"]}
Opening port
Port open
{"serial_handshake": "failed"}
{"serial_handshake": {"baud_rates": [9600, 19200, 38400, 14400, 28800, 57600, 115200], "configs": ["5N1", "5N2", "5O1", "5O2", "5E1", "5E2", "6N1", "6N2", "6O1", "6O2", "6E1", "6E2", "7N1", "7N2", "7O1", "7O2", "7E1", "7E2", "8N1", "8N2", "8O1", "8O2", "8E1", "8E2"], "newlines": ["\r\n", "\n"]}}
{"serial_handshake": "failed"}
{"connect": "init"}
{"serial_handshake": "failed"}
{"connect": "init"}
{"connect": "init"}
{"serial_handshake": "failed"}
{"available_data": {"temperature": "double", "humidity": "double"}}
{"error": "Invalid argument!"} {"error": "invalid_request is not a valid request!"}
{"error": "Invalid argument!"} {"humidity": 6}
```

Invalid parameter type:

For rotate- steps and degrees, steps has to be provided as an int, where as degrees has to be provided as a float. If the parameter following either is of a different type, an error message will be prompted stating that the provided value is invalid.



```
Serial | COM9 - Arduino Mega 2560
{"request": "rotate", "steps": "Invalid_parameter"}
Opening port
Port open
{"serial_handshake": {"baud_rates": [9600, 19200, 38400, 14400, 28800, 57600, 115200], "configs": ["5N1", "5N2", "5O1", "5O2", "5E1", "5E2", "6N1", "6N2", "6O1", "6O2", "6E1", "6E2", "7N1", "7N2", "7O1", "7O2", "7E1", "7E2", "8N1", "8N2", "8O1", "8O2", "8E1", "8E2"], "newlines": ["\r\n", "\n"]}}
{"connect": "init"}
{"available_data": {"temperature": "double", "humidity": "double"}}
{"error": "Invalid value!"}
```

Bachelor of Engineering

Project appendix

Winter semester 2019

SatStat Hardware Layer Beta 1 - Requirements and verification

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

The hardware layer (HWL) of the Diagnostics System has two main responsibilities. One of these is to store test instructions received from the Software Layer (SWL), and translate these instructions to the proper format in order to execute them on the device corresponding to the initial instruction. The SWL also expect a continuous stream of sensor readings coming from the HWL. This requires the HWL to be able to read and transmit the prompted sensor data. This documents describes the requirements the HWL has to fulfill, and the various tests/verifications that has to be carried out in order to ensure that these requirements are met.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date	Signature
Kongsberg, May 20, 2019	

Diagnostics System - Hardware Layer version Beta1

Target release	BETA 1
Epic	USN5-95 - Hardware layer of the Diagnostics System IN PROGRESS
Document status	V 1.0
Document owner	Jon Skjelsbek
Designer	Jon Skjelsbek
Tech lead	Jon Skjelsbek

Objective

The hardware layer(HWL) of the Diagnostics System has two main responsibilities. One of these is to store test instructions received from the Software Layer(SWL), and translate these instructions to the proper format in order to execute them on the device corresponding to the initial instruction. The SWL also expect a continuous stream of sensor readings coming from the HWL. This requires the HWL to be able to read and transmit the prompted sensor data. This documents describes the requirements the HWL has to fulfill, and the various tests/verifications that has to be carried out in order to ensure that these requirements are met.

Success metrics

Goal	Metric
Receive and store instructions from SWL.	Received instructions match the sent one.
Translate and execute fetched instructions.	The execution result replicate the initial instruction.
Provide prompted sensor data	SWL continuously receive the sensor data it asked for, and the corresponding values makes sense.

Roadmap

The HWL and SWL follow the same roadmap as when functionality is added to one layer, the other layer has to support it.

ID	Task Name	Start	Finish	Duration	24 mar 2019						31 mar 2019						7 apr 2019						14 apr 2019			
					25	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	Save COM (R2-33)	25.03.2019	25.03.2019	1d	[Gantt bar]																					
2	Plot optimizations (R2-27, R4-15)	25.03.2019	30.03.2019	6d	[Gantt bar]																					
3	UI diag settings (R2-34)	28.03.2019	31.03.2019	4d	[Gantt bar]																					
4	UI live values status indicator (R2-34)	29.03.2019	31.03.2019	3d	[Gantt bar]																					
5	Code structure review and optimization (R2-28)	01.04.2019	07.04.2019	7d	[Gantt bar]																					
6	UI optimization and design overhaul	01.04.2019	07.04.2019	7d	[Gantt bar]																					
7	Automated tests (R2-34, R2-35)	08.04.2019	14.04.2019	7d	[Gantt bar]																					
8	UI diag settings	08.04.2019	09.04.2019	2d	[Gantt bar]																					
9	Diag settings template saving	09.04.2019	11.04.2019	3d	[Gantt bar]																					
10	Run tests from template	08.04.2019	14.04.2019	7d	[Gantt bar]																					
11	Beta1	15.04.2019	15.04.2019	0d	◆																					
12																										

Revised roadmap

ID	Task Name	Start	Finish	Duration	31 mar 2019						7 apr 2019						14 apr 2019						21 apr 2019			
					31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	Save COM (R2-33)	31.03.2019	31.03.2019	1d	[Gantt bar]																					
2	Plot optimizations (R2-27, R4-15)	31.03.2019	05.04.2019	6d	[Gantt bar]																					
3	UI diag settings (R2-34)	02.04.2019	05.04.2019	4d	[Gantt bar]																					
4	UI live values status indicator (R2-34)	06.04.2019	08.04.2019	3d	[Gantt bar]																					
5	Diag settings template saving	09.04.2019	14.04.2019	6d	[Gantt bar]																					
6	Beta1	15.04.2019	15.04.2019	0d	◆																					
7	Run tests from template	15.04.2019	21.04.2019	7d	[Gantt bar]																					
8	Automated tests (R2-34, R2-35)	15.04.2019	21.04.2019	7d	[Gantt bar]																					
9	Beta2	22.04.2019	22.04.2019	0d	◆																					
10	UI optimization and design overhaul	22.04.2019	22.04.2019	0d	◆																					
11	Code structure review and optimization (R2-28)	22.04.2019	22.04.2019	0d	◆																					

Requirements

ID	Description	Importance	Version
R4-26	The HWL should be able to enable or disable different sensors according to received instructions.	MEDIUM	BETA 1
Date	Discussion and decisions		
15.03.2019			
User Story			
USN5-251 - As an embedded developer, I need to implement functionality supporting a subscriber system for sensor data to comply with the SatStat communication protocol. DONE			

Verification ID	Verification method	Status
TV-47	Functional test: Run as many instructions as see fit, and confirm that the desired result is achieved.	PASSED

Bachelor of Engineering

Project appendix

Winter semester 2019

SatStat Hardware Layer Beta 1 - Class Diagram

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

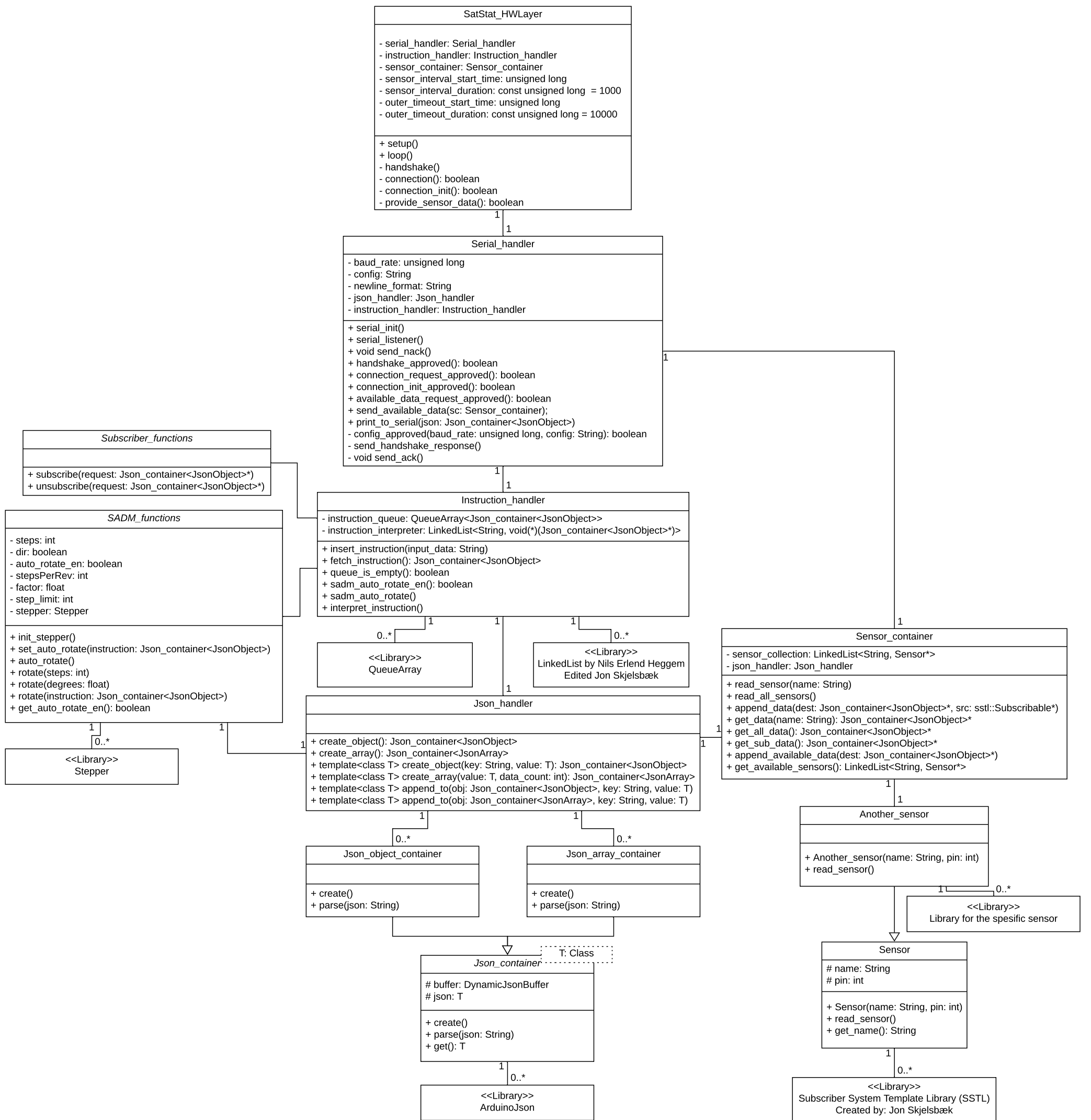
The main goal of this document is to describe the code visually in the form of a class diagram. This class diagram in particular is for version Beta 1.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

SatStat Hardware Layer Beta 1 Class Diagram

Jon Skjelsbæk | May 13, 2019



Bachelor of Engineering

Project appendix

Winter semester 2019

VR-HWL-B1

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

Verification report for requirements related to the Hardware layer of the diagnostics system (SatStat), version Beta 1.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

VR-HWL-B1

Date performed	15.04.2019
Performed by	Jon Skjelsbek
Tests performed (1)	TV-47
Tests discarded (0)	
System version	Beta 1

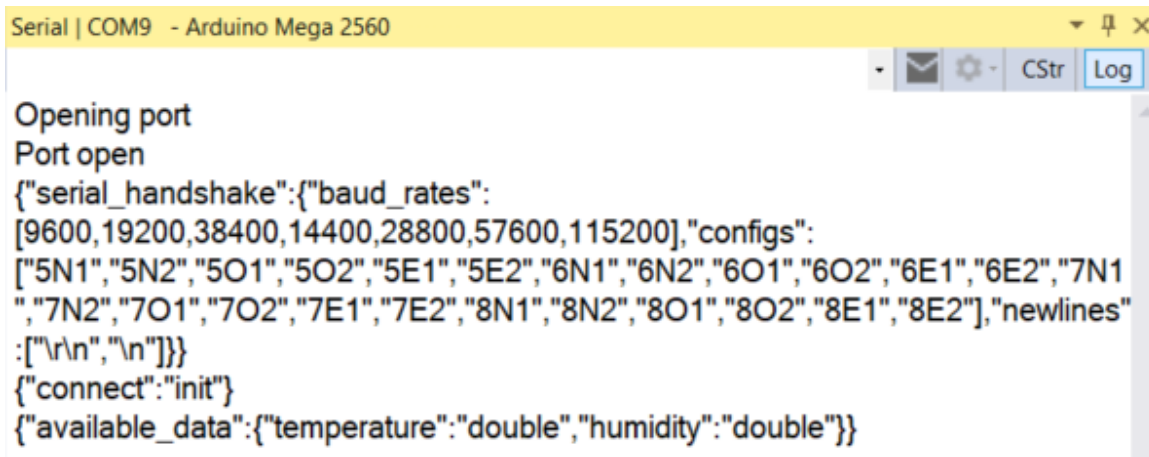
Tests

TV-47

This test is carried out by subscribing and unsubscribing to different sensor data one at a time, as well as all at the same time. Observations define if the test is considered passed or failed.

Before subscription:

The HWL will wait for a request after the handshake protocol as illustrated in the image below.



```
Serial | COM9 - Arduino Mega 2560
Opening port
Port open
{"serial_handshake":{"baud_rates":
[9600,19200,38400,14400,28800,57600,115200],"configs":
["5N1","5N2","5O1","5O2","5E1","5E2","6N1","6N2","6O1","6O2","6E1","6E2","7N1
","7N2","7O1","7O2","7E1","7E2","8N1","8N2","8O1","8O2","8E1","8E2"],"newlines"
:["\r\n","\n"]}}
{"connect":"init"}
{"available_data":{"temperature":"double","humidity":"double"}}
```

Subscribe temperature:

Passed: Checking if temperature is prompted when subscribed to.

```
Serial | COM9 - Arduino Mega 2560
["request":"subscribe", "parameters": ["temperature"]]
Opening port
Port open
{"serial_handshake":{"baud_rates":
[9600,19200,38400,14400,28800,57600,115200],"configs":
["5N1","5N2","5O1","5O2","5E1","5E2","6N1","6N2","6O1","6O2","6E1","6E2","7N1
","7N2","7O1","7O2","7E1","7E2","8N1","8N2","8O1","8O2","8E1","8E2"],"newlines"
:["\r\n","\n"]}}
{"connect":"init"}
{"available_data":{"temperature":"double","humidity":"double"}}
{"temperature":25}
```

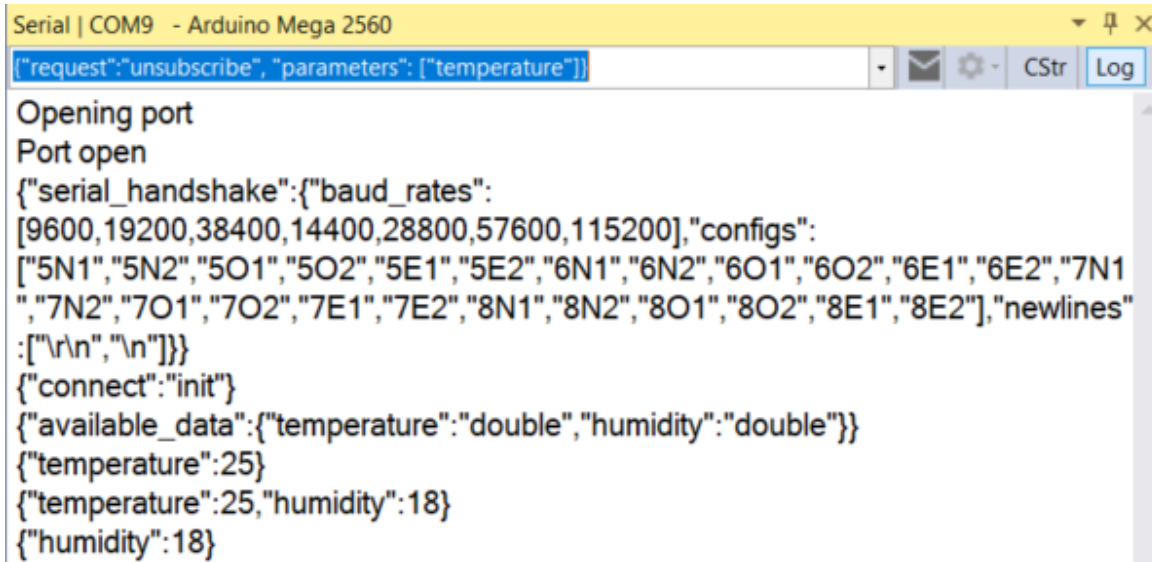
Subscribe humidity:

Passed: Checking if humidity is prompted when subscribed to.

```
Serial | COM9 - Arduino Mega 2560
["request":"subscribe", "parameters": ["humidity"]]
Opening port
Port open
{"serial_handshake":{"baud_rates":
[9600,19200,38400,14400,28800,57600,115200],"configs":
["5N1","5N2","5O1","5O2","5E1","5E2","6N1","6N2","6O1","6O2","6E1","6E2","7N1
","7N2","7O1","7O2","7E1","7E2","8N1","8N2","8O1","8O2","8E1","8E2"],"newlines"
:["\r\n","\n"]}}
{"connect":"init"}
{"available_data":{"temperature":"double","humidity":"double"}}
{"temperature":25}
{"temperature":25,"humidity":18}
```

Unsubscribe temperature:

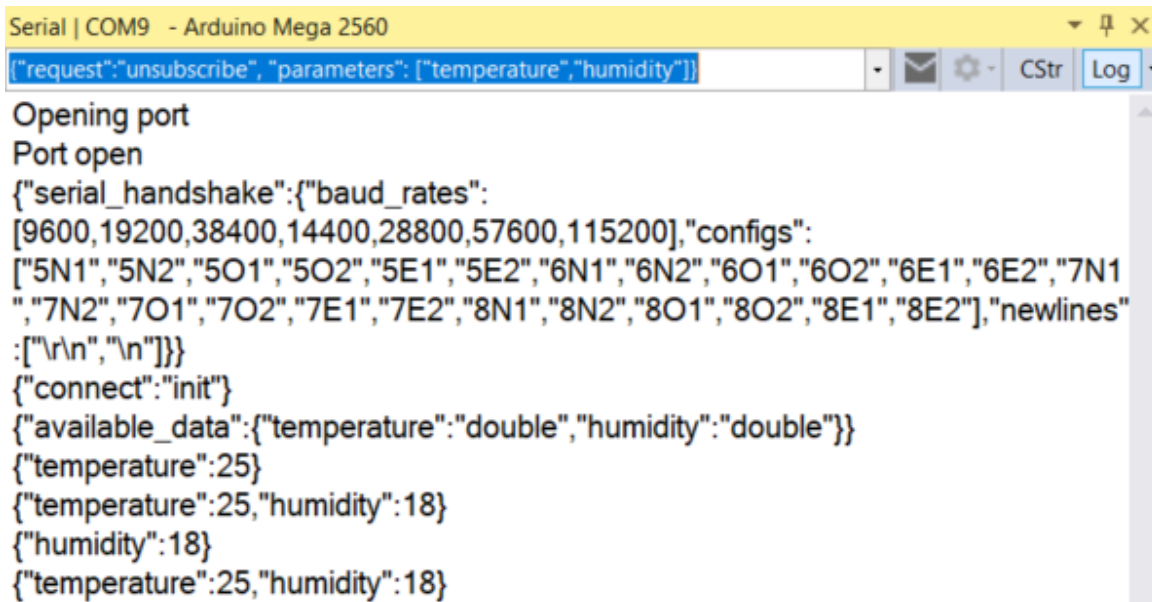
Passed: Checking if only humidity is prompted when unsubscribing from temperature.



```
Serial | COM9 - Arduino Mega 2560
[{"request": "unsubscribe", "parameters": ["temperature"]}
Opening port
Port open
{"serial_handshake":{"baud_rates":
[9600,19200,38400,14400,28800,57600,115200],"configs":
["5N1","5N2","5O1","5O2","5E1","5E2","6N1","6N2","6O1","6O2","6E1","6E2","7N1
","7N2","7O1","7O2","7E1","7E2","8N1","8N2","8O1","8O2","8E1","8E2"],"newlines"
:["\r\n","\n"]}}
{"connect": "init"}
{"available_data":{"temperature": "double", "humidity": "double"}}
{"temperature": 25}
{"temperature": 25, "humidity": 18}
{"humidity": 18}
```

Unsubscribe humidity, "subscribe to all" and "unsubscribe from all":

Passed: Unsubscribing to the last entry in the subscription list, as well as all at once, is hard to visualize in an image. In the image below, it's possible to see that the unsubscription of humidity worked, as temperature appears first in the consecutive message. This also indicates that "subscribe to all" works properly. "unsubscribe from all" isn't visualized in the image below, as the console simply won't receive anything before sending another request. Observations confirms that "unsubscribe from all" complies with the expected outcome.



```
Serial | COM9 - Arduino Mega 2560
[{"request": "unsubscribe", "parameters": ["temperature", "humidity"]}
Opening port
Port open
{"serial_handshake":{"baud_rates":
[9600,19200,38400,14400,28800,57600,115200],"configs":
["5N1","5N2","5O1","5O2","5E1","5E2","6N1","6N2","6O1","6O2","6E1","6E2","7N1
","7N2","7O1","7O2","7E1","7E2","8N1","8N2","8O1","8O2","8E1","8E2"],"newlines"
:["\r\n","\n"]}}
{"connect": "init"}
{"available_data":{"temperature": "double", "humidity": "double"}}
{"temperature": 25}
{"temperature": 25, "humidity": 18}
{"humidity": 18}
{"temperature": 25, "humidity": 18}
```

Bachelor of Engineering

Project appendix

Winter semester 2019

SatStat Hardware Layer Beta 2 - Requirements and verification

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

The hardware layer (HWL) of the Diagnostics System has two main responsibilities. One of these is to store test instructions received from the Software Layer (SWL), and translate these instructions to the proper format in order to execute them on the device corresponding to the initial instruction. The SWL also expect a continuous stream of sensor readings coming from the HWL. This requires the HWL to be able to read and transmit the prompted sensor data. This documents describes the requirements the HWL has to fulfill, and the various tests/verifications that has to be carried out in order to ensure that these requirements are met.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

Diagnostics System - Hardware Layer version Beta2

Target release	BETA 2
Epic	USN5-95 - Hardware layer of the Diagnostics System IN PROGRESS
Document status	V 1.0
Document owner	Jon Skjelsbek
Designer	Jon Skjelsbek
Tech lead	Jon Skjelsbek

Objective

The hardware layer(HWL) of the Diagnostics System has two main responsibilities. One of these is to store test instructions received from the Software Layer(SWL), and translate these instructions to the proper format in order to execute them on the device corresponding to the initial instruction. The SWL also expect a continuous stream of sensor readings coming from the HWL. This requires the HWL to be able to read and transmit the prompted sensor data. This documents describes the requirements the HWL has to fulfill, and the various tests/verifications that has to be carried out in order to ensure that these requirements are met.

Success metrics

Goal	Metric
Receive and store instructions from SWL.	Received instructions match the sent one.
Translate and execute fetched instructions.	The execution result replicate the initial instruction.
Provide prompted sensor data	SWL continuously receive the sensor data it asked for, and the corresponding values makes sense.

Roadmap

The HWL and SWL follow the same roadmap as when functionality is added to one layer, the other layer has to support it.

ID	Task Name	Start	Finish	Duration	31 mar 2019							7 apr 2019							14 apr 2019							21 apr 2019						
					31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22					
1	Save COM (R2-33)	31.03.2019	31.03.2019	1d	[Gantt bar]																											
2	Plot optimizations (R2-27, R4-15)	31.03.2019	05.04.2019	6d	[Gantt bar]																											
3	UI diag settings (R2-34)	02.04.2019	05.04.2019	4d	[Gantt bar]																											
4	UI live values status indicator (R2-34)	06.04.2019	08.04.2019	3d	[Gantt bar]																											
5	Diag settings template saving	09.04.2019	14.04.2019	6d	[Gantt bar]																											
6	Beta1	15.04.2019	15.04.2019	0d	[Milestone diamond]																											
7	Run tests from template	15.04.2019	21.04.2019	7d	[Gantt bar]																											
8	Automated tests (R2-34, R2-35)	15.04.2019	21.04.2019	7d	[Gantt bar]																											
9	Beta2	22.04.2019	22.04.2019	0d	[Milestone diamond]																											
10	UI optimization and design overhaul	22.04.2019	22.04.2019	0d	[Milestone diamond]																											
11	Code structure review and optimization (R2-28)	22.04.2019	22.04.2019	0d	[Milestone diamond]																											

Requirements

ID	Description	Importance	Version
R7-09	The final DS HWL shall be as effective as possible in the sens of proper design and having no redundant code.	MEDIUM	BETA 2
Date	Discussion and decisions		
17.03.2019			
User Story			
USN5-250 - As an embedded developer, I need to optimize the current version of the HWL to prepare for further development.			
DONE			
Verification ID	Verification method	Status	
IV-18	Inspect the class diagram to check for redundant code and changes that could be made to improve performance. If it looks good, the verification can be considered passed.	PASSED	

ID	Description	Importance	Version
R4-27	The HWL should provide a list of available instructions to the SWL when prompted.	HIGH	BETA 2
Date	Discussion and decisions		

24.04.2019		
User Story		
USN5-311 - As a user of the HMI I want a list of available instructions so that I don't have to remember all of them		
DONE		
Verification ID	Verification method	Status
TV-52	Functional test: Request a list of available instructions and check if what's sent reflects the expectations.	PASSED

Bachelor of Engineering

Project appendix

Winter semester 2019

SatStat Hardware Layer Beta 2 - Class Diagram

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

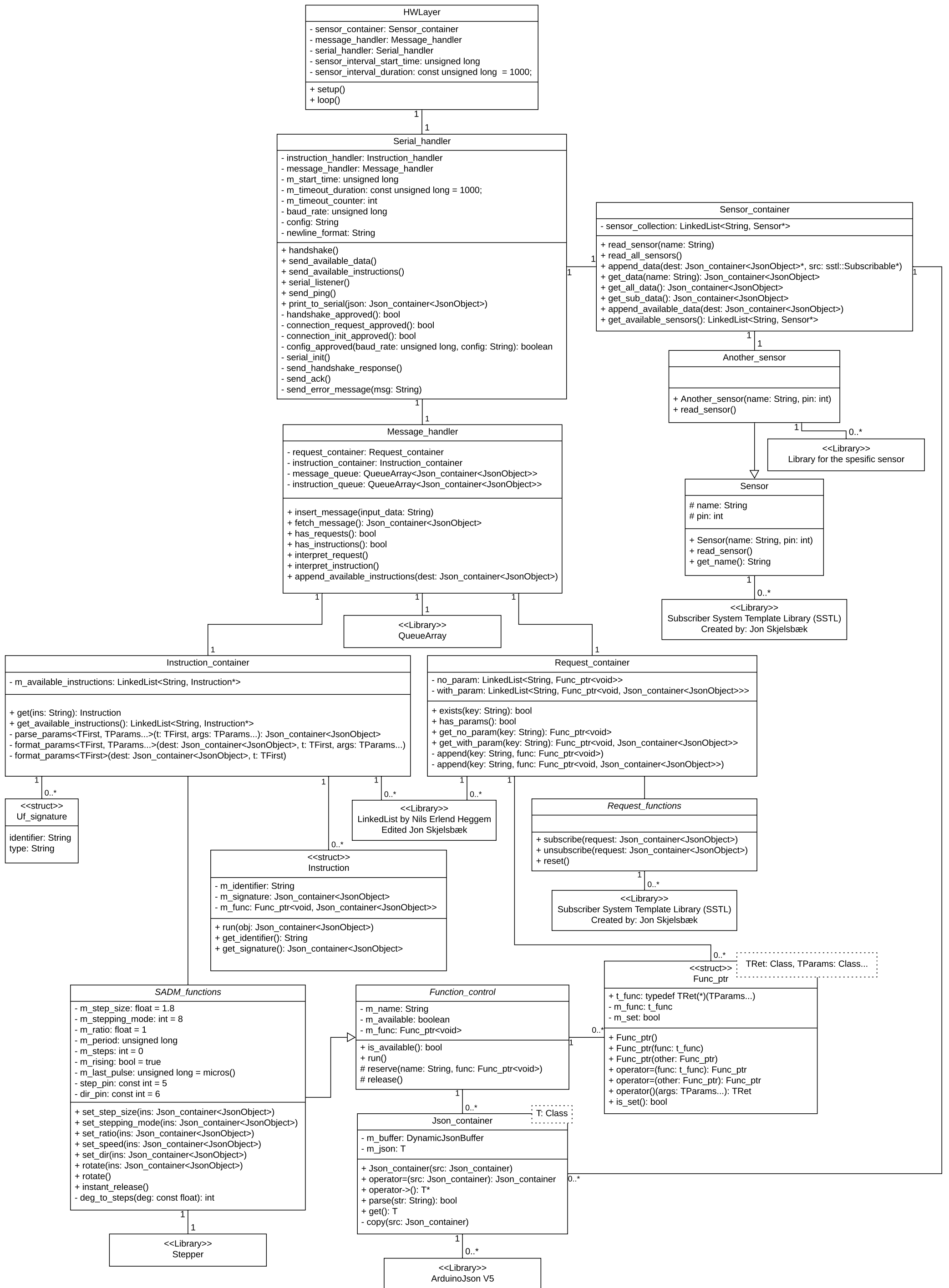
The main goal of this document is to describe the code visually in the form of a class diagram. This class diagram in particular is for version Beta 2.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

SatStat Hardware Layer Beta 2 Class Diagram

Jon Skjelsbæk | May 13, 2019



Bachelor of Engineering

Project appendix

Winter semester 2019

VR-HWL-B2

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

Verification report for requirements related to the Hardware layer of the diagnostics system (SatStat), version Beta 2.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

VR-HWL-B2

Date performed	06.05.2019
Performed by	Jon Skjelsbek
Tests performed (0)	IV-18, TV-52
Tests discarded (0)	
System version	Beta 2

Tests

TV-52

Functional test: Request a list of available instructions and check if what's sent reflects the expectations.

Available instructions in raw format observed in the serial monitor:

```
Serial | COM9 - Arduino Mega 2560
["connect": "ok"]
Opening port
Port open
{"device_name": "SatLight SADM V1.0"}
{"serial_handshake": {"baud_rates": [9600, 19200, 38400, 14400, 28800, 57600, 115200], "configs":
["5N1", "5N2", "5O1", "5O2", "5E1", "5E2", "6N1", "6N2", "6O1", "6O2", "6E1", "6E2", "7N1", "7N2", "7O1", "7O2", "7E1", "7E2", "8N1", "8N2", "8O1", "8O2", "8E1", "8E2"], "newlines": ["\r\n", "\n"]}}
{"connect": "init"}
{"available_data": {"temperature": "float", "humidity": "float"}}
{"available_instructions": {"set_gear_ratio": {"ratio": "float"}, "auto_rotate": {"enable": "bool"}, "rotate_steps": {"steps": "int"}, "rotate_degrees": {"degrees": "float"}}}
{"device_name": "SatLight SADM V1.0"}
```

Available instructions captured and selectable from SWL:

Plot view | Parameter control | Test configuration

Test configuration

Test name:

Test description:

Parameter control template:

Use current parameter control configuration

Save test configuration:

Run test: Abort test:

Test device:

Saved test configurations:

Add Instruction

Instruction name:

set_gear_ratio
auto_rotate
rotate_steps
rotate_degrees

Add instruction:

Intruction parameters

Parameter key	Parameter value

Output parameters

temperature

humidity

Instruction list

Instruction name	Parameters	State

↑

↓

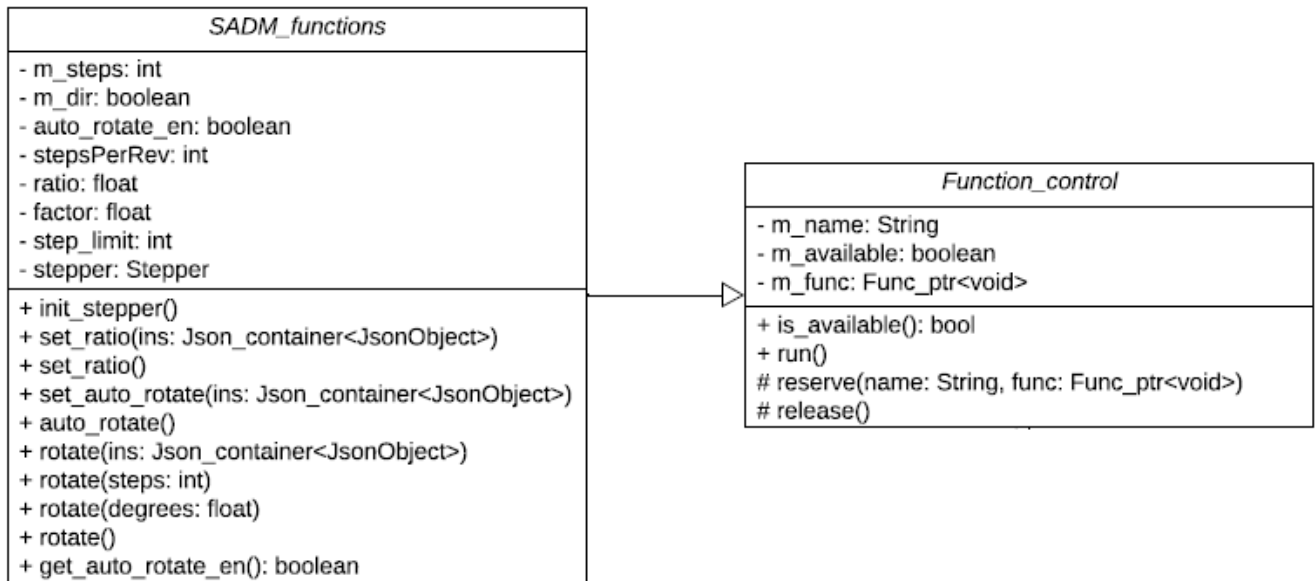
X

IV-18

Inspect the class diagram to check for redundant code and changes that could be made to improve performance. If it looks good, the verification can be considered passed.

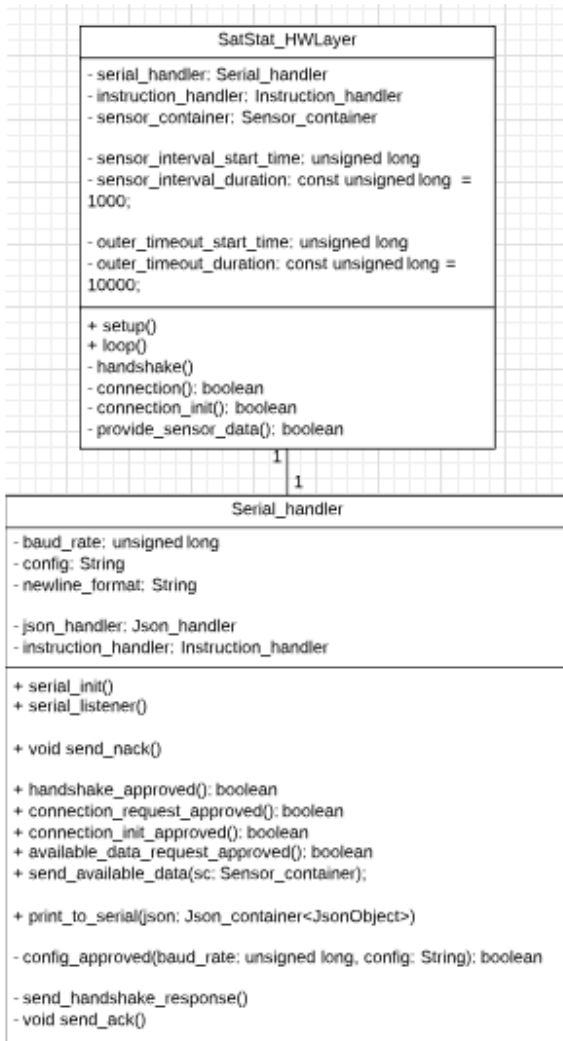
Function_control class added to simplify protothreading of functions. Works like a semaphore where only one function can enter at a time, and the reserving function itself has to lift the lock before other functions are allowed to take it's place.

Function_control inherited in SADM_functions:

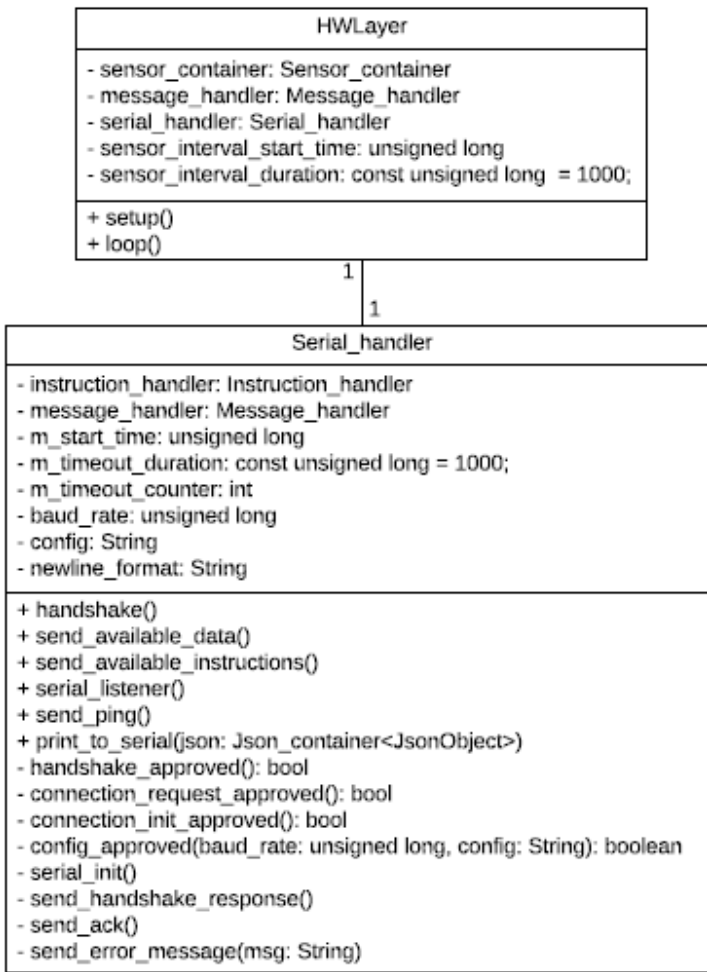


Up until Beta 2, the "main class" had functions handling parts of the handshake protocol. These functions are in Beta 2 moved to the Serial_handler class to improve the general structure.

Beta 1 "main class" and Serial_handler:

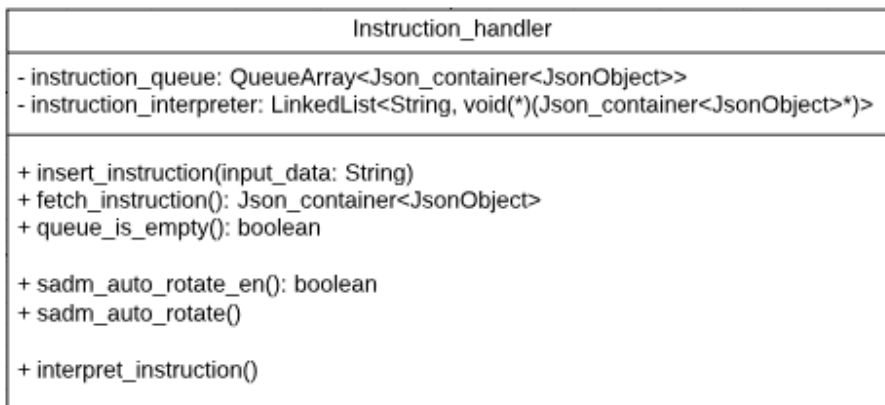


Beta 2 "Main class" and Serial_handler:

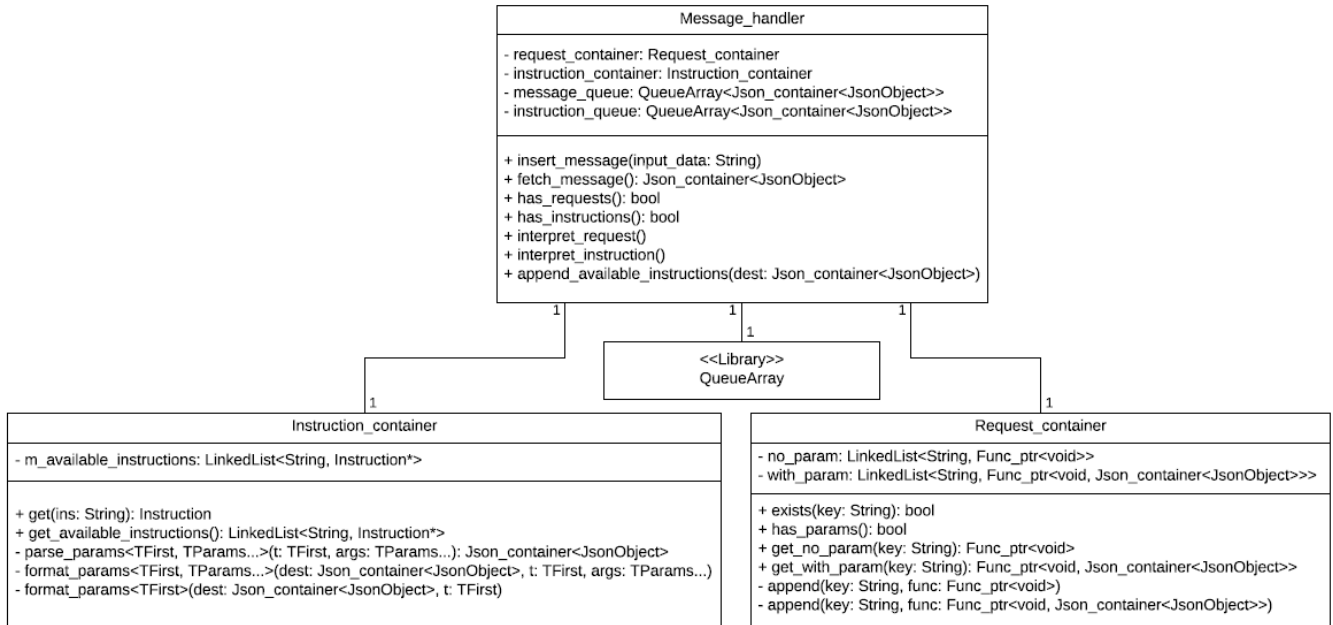


The Instruction_handler has been replaced by the Message_handler in Beta 2. The reason for this is to separate instructions and request while still having a common handler for both of them. This requires the addition of the Instruction_container and the Request_container.

Beta 1 Instruction_handler:

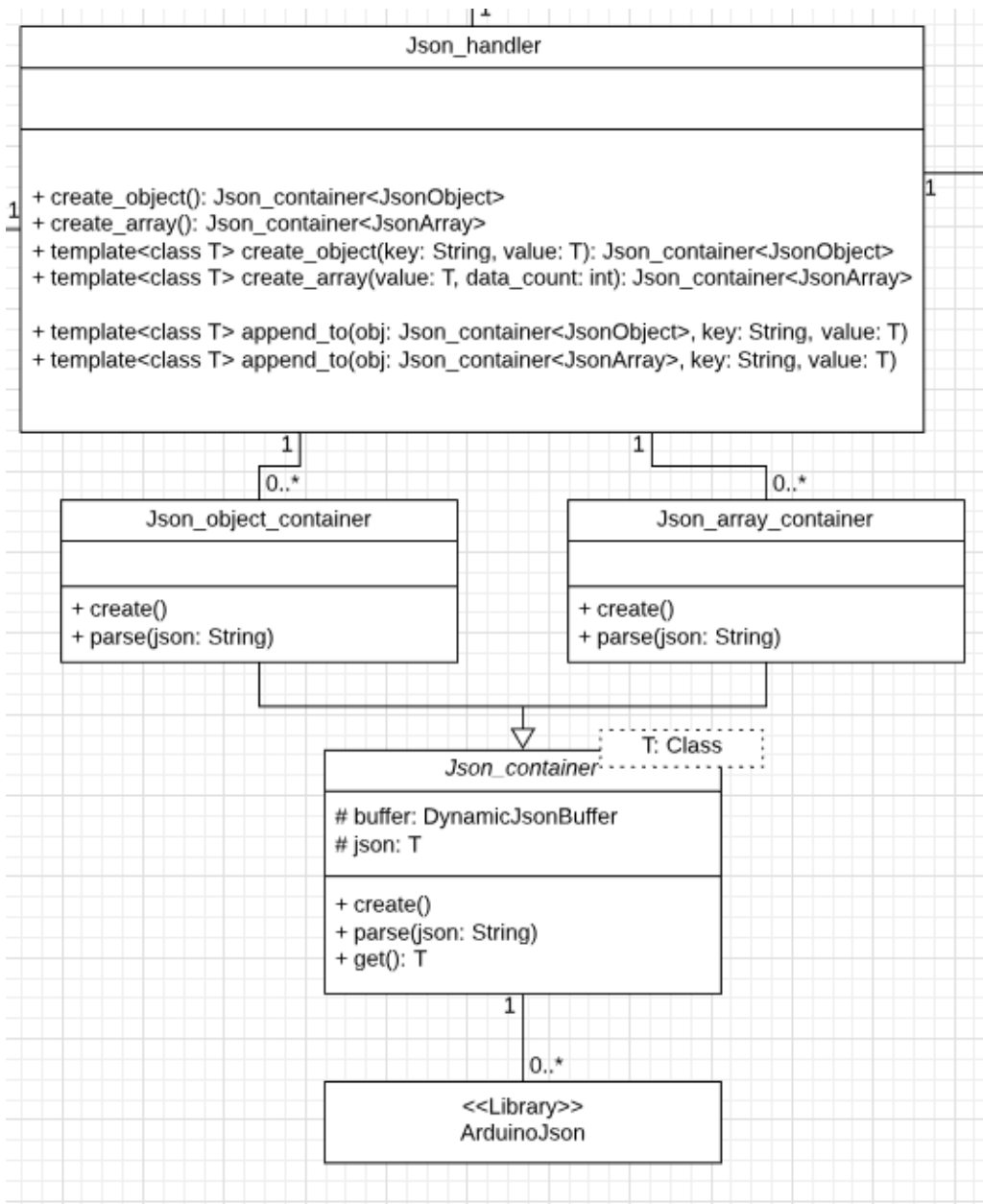


Beta 2 Instruction_handler equivalent:

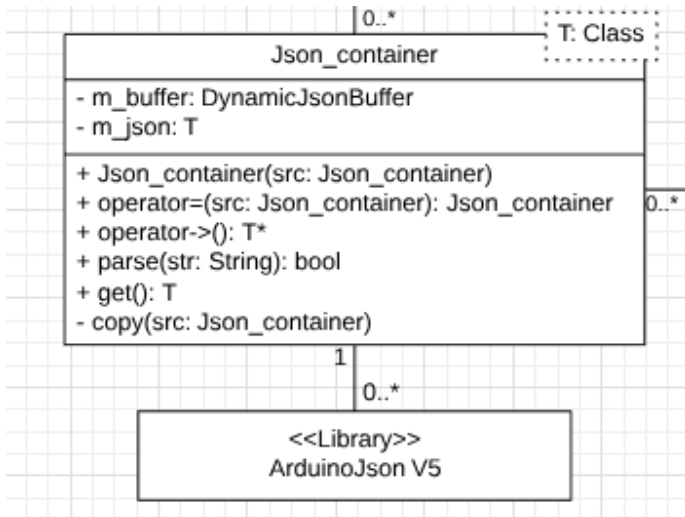


The three classes complementing the Json_container in Beta 1 was there mainly to access member functions of the json member variable in the Json_container. In Beta 2, these members can be accessed by utilizing the arrow operator overloaded in the Json_container class. This eliminates the need of the complementary classes, therefore they're removed in Beta 2.

Beta 1 Json_handler:



Beta 2 Json_container:



External resources

SatStat HWL Beta 1 Class Diagram: <https://www.lucidchart.com/invitations/accept/97613a83-6026-4e5f-b188-d95e7ab36f23>

SatStat HWL Beta 2 Class Diagram: <https://www.lucidchart.com/invitations/accept/df4347de-68c5-494b-9629-b516a0fe8167>



Bachelor of Engineering

Project appendix

Winter semester 2019

LinkedList - Class Diagram

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

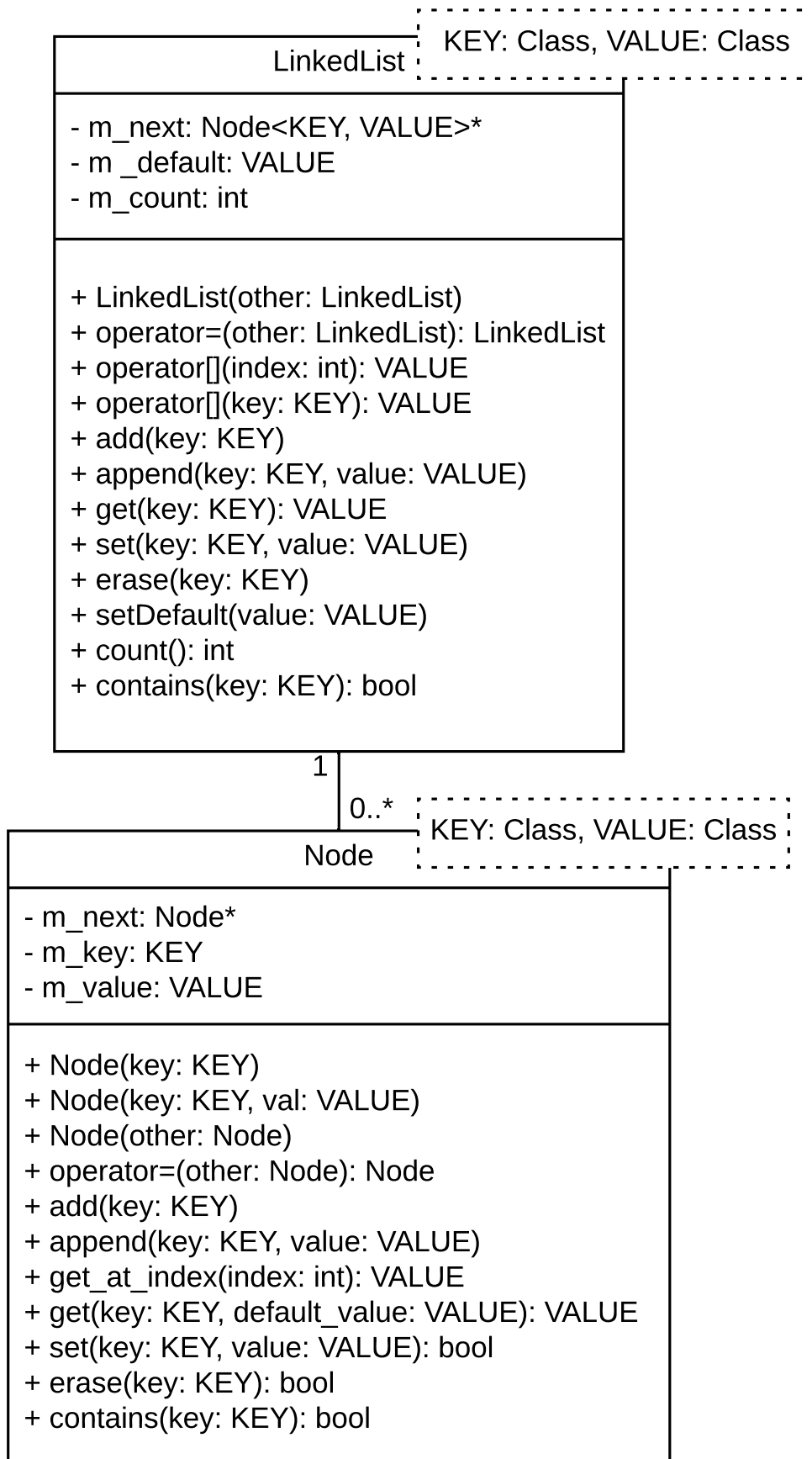
The main goal of this document is to describe the LinkedList library code visually in the form of a class diagram.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

LinkedList Class Diagram

Nils Erlend Heggem & Jon Skjelsbæk | May 7, 2019



Bachelor of Engineering

Project appendix

Winter semester 2019

SSTL - Class Diagram

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

The main goal of this document is to describe the code for the Subscriber System Template Library (SSTL) visually in the form of a class diagram.

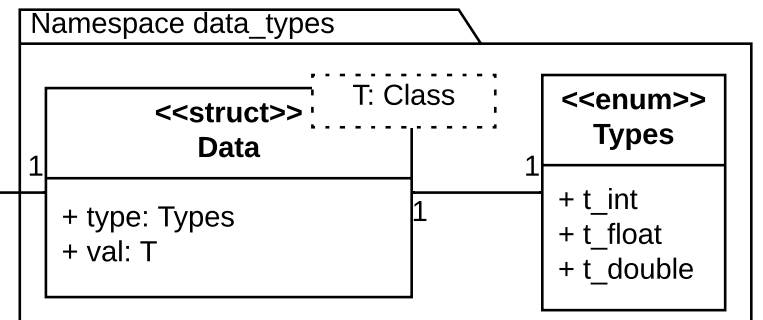
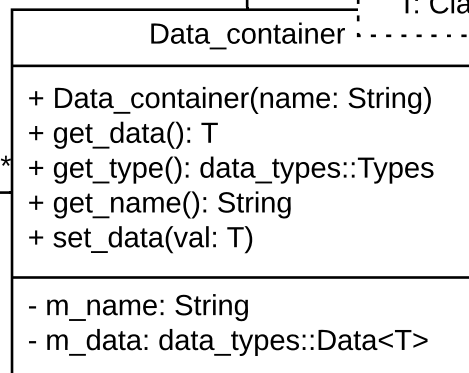
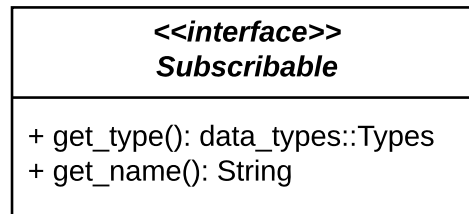
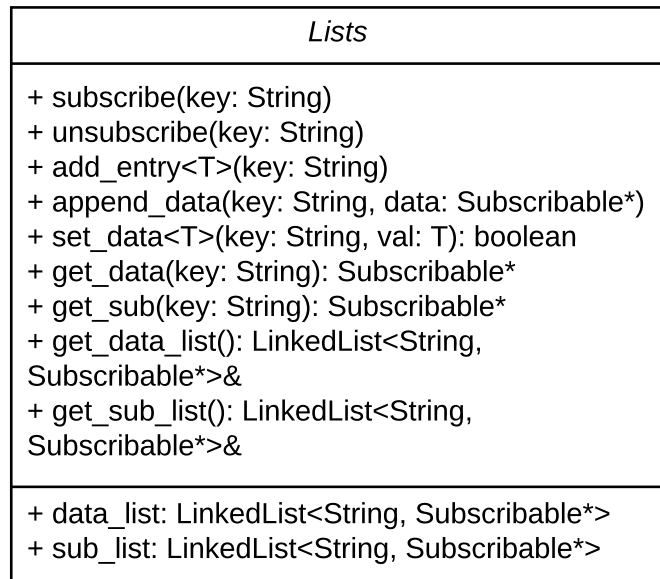
I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

Subscriber System Template Library Class Diagram

Jon Skjelsbæk | May 7, 2019

Namespace sstl



1

2

1

0..*

1

1

1

1



KONGSBERG

**Bachelor of Engineering****Project appendix****Winter semester 2019**

LinkedList - API Documentation

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document describes the code for the LinkedList library in great detail. It's recommended that it's used as support material for the LinkedList - Class Diagram, as it's large and complex on it's own.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------



KONGSBERG

LinkedList

V1.0

Generated by Doxygen 1.8.15

1 Class Index	1
<hr/>	
1 Class Index	1
1.1 Class List	1
2 Class Documentation	1
2.1 LinkedList< KEY, VALUE > Class Template Reference	1
2.1.1 Detailed Description	2
2.1.2 Constructor & Destructor Documentation	2
2.1.3 Member Function Documentation	2
2.2 Node< KEY, VALUE > Class Template Reference	4
2.2.1 Detailed Description	5
2.2.2 Constructor & Destructor Documentation	5
2.2.3 Member Function Documentation	6
Index	9

1 Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

LinkedList< KEY, VALUE >	1
Node< KEY, VALUE >	4

2 Class Documentation

2.1 LinkedList< KEY, VALUE > Class Template Reference

```
#include <LinkedList.h>
```

Public Member Functions

- **LinkedList** ()
- **LinkedList** (const **LinkedList**< KEY, VALUE > &other)
- **~LinkedList** ()
- **LinkedList**< KEY, VALUE > & **operator=** (const **LinkedList**< KEY, VALUE > &other)
- VALUE & **operator[]** (int index)
- VALUE & **operator[]** (const KEY &key)
- void **add** (const KEY &key)
- void **append** (const KEY &key, const VALUE &value)
- VALUE & **get** (const KEY &key)
- void **set** (const KEY &key, const VALUE &value)
- void **erase** (const KEY &key)
- void **setDefault** (const VALUE &value)
- const int & **count** () const
- bool **contains** (const KEY &key) const

Generated by Doxygen

Private Attributes

- **Node**< KEY, VALUE > * **m_next**
- VALUE **m_default**
- int **m_count**

2.1.1 Detailed Description

```
template<class KEY, class VALUE>
class LinkedList< KEY, VALUE >
```

LinkedList (p. 1) is a generic linked list class implementation. It contains the first **Node** (p. 4) in the list, and has methods to access and modify these nodes.

2.1.2 Constructor & Destructor Documentation**2.1.2.1 LinkedList()** [1/2]

```
template<class KEY , class VALUE >
LinkedList< KEY, VALUE >:: LinkedList ( ) [inline]
```

Default constructor. Setting **m_next** to nullptr and **m_count** to zero.

2.1.2.2 LinkedList() [2/2]

```
template<class KEY , class VALUE >
LinkedList< KEY, VALUE >:: LinkedList (
    const LinkedList< KEY, VALUE > & other ) [inline]
```

Copy constructor. Using assignment operator to perform a deep copy from *other* to this.

2.1.2.3 ~LinkedList()

```
template<class KEY , class VALUE >
LinkedList< KEY, VALUE >::~ LinkedList ( ) [inline]
```

Destructor. Deleting **m_next** which will call **Node** (p. 4)'s destructor recursively deleting every **Node** (p. 4) in the list.

2.1.3 Member Function Documentation**2.1.3.1 add()**

```
template<class KEY , class VALUE >
void LinkedList< KEY, VALUE >::add (
    const KEY & key ) [inline]
```

Adds an entry to the list with no predefined value.

2.1 LinkedList< KEY, VALUE > Class Template Reference**3****2.1.3.2 append()**

```
template<class KEY , class VALUE >
void LinkedList< KEY, VALUE >::append (
    const KEY & key,
    const VALUE & value ) [inline]
```

Appends an entry to the list with predefined value.

2.1.3.3 contains()

```
template<class KEY , class VALUE >
bool LinkedList< KEY, VALUE >::contains (
    const KEY & key ) const [inline]
```

Checks if an object with the given key is present in the list. Returns true if it is, false if not.

2.1.3.4 count()

```
template<class KEY , class VALUE >
const int & LinkedList< KEY, VALUE >::count ( ) const [inline]
```

Returns the number of objects in the list.

2.1.3.5 erase()

```
template<class KEY , class VALUE >
void LinkedList< KEY, VALUE >::erase (
    const KEY & key ) [inline]
```

Erases an object from the list.

2.1.3.6 get()

```
template<class KEY , class VALUE >
VALUE & LinkedList< KEY, VALUE >::get (
    const KEY & key ) [inline]
```

Looks for an object with the given key, returns it's value if found, returns default value if not.

2.1.3.7 operator=()

```
template<class KEY , class VALUE >
LinkedList< KEY, VALUE > & LinkedList< KEY, VALUE >::operator= (
    const LinkedList< KEY, VALUE > & other ) [inline]
```

Assignment operator overload. Deletes m_next if not nullptr, and performs a deep copy from other to this. Returns a reference to this to allow chaining.

2.1.3.8 operator[]() [1/2]

```
template<class KEY , class VALUE >
VALUE & LinkedList< KEY, VALUE >::operator[] (
    int index ) [inline]
```

Subscript member access operator overload. Returns the value at the given index. Returns default value if index out of bounds.

Generated by Doxygen

2.1.3.9 operator[]() [2/2]

```
template<class KEY , class VALUE >
VALUE & LinkedList< KEY, VALUE >::operator[] (
    const KEY & key ) [inline]
```

Subscript member access operator overload. Returns the result of the get method.

2.1.3.10 set()

```
template<class KEY , class VALUE >
void LinkedList< KEY, VALUE >::set (
    const KEY & key,
    const VALUE & value ) [inline]
```

Sets the value of an existing object. Appends a new object if no object with the given already key exists.

2.1.3.11 setDefault()

```
template<class KEY , class VALUE >
void LinkedList< KEY, VALUE >::setDefault (
    const VALUE & value ) [inline]
```

Sets the default value.

2.2 Node< KEY, VALUE > Class Template Reference

```
#include <Node.h>
```

Public Member Functions

- **Node** ()
- **Node** (const KEY &key)
- **Node** (const KEY &key, const VALUE &val)
- **Node** (const **Node**< KEY, VALUE > &other)
- **~Node** ()
- **Node**< KEY, VALUE > & **operator=** (const **Node**< KEY, VALUE > &other)
- void **add** (const KEY &key)
- void **append** (const KEY &key, const VALUE &value)
- VALUE & **get_at_index** (int &index)
- VALUE & **get** (const KEY &key, VALUE &default_value)
- bool **set** (const KEY &key, const VALUE &value)
- bool **erase** (const KEY &key, **Node** *&parent_m_next)
- bool **contains** (const KEY &key) const

Private Attributes

- **Node** * **m_next**
- KEY **m_key**
- VALUE **m_value**

2.2 Node< KEY, VALUE > Class Template Reference**5****2.2.1 Detailed Description**

```
template<class KEY, class VALUE>
class Node< KEY, VALUE >
```

Node (p. 4) is a generic class implementation for the nodes contained in a linked list. This class is complementary to the **LinkedList** (p. 1) class, and contains the key-value pairs.

2.2.2 Constructor & Destructor Documentation**2.2.2.1 Node()** [1/4]

```
template<class KEY , class VALUE >
Node< KEY, VALUE >:: Node ( ) [inline]
```

Default constructor. Setting m_next to nullptr.

2.2.2.2 Node() [2/4]

```
template<class KEY , class VALUE >
Node< KEY, VALUE >:: Node (
    const KEY & key ) [inline]
```

Constructor. Setting m_next to nullptr, and m_key to the given key.

2.2.2.3 Node() [3/4]

```
template<class KEY , class VALUE >
Node< KEY, VALUE >:: Node (
    const KEY & key,
    const VALUE & val ) [inline]
```

Constructor. Sets m_next to nullptr, and sets both the key and value.

2.2.2.4 Node() [4/4]

```
template<class KEY , class VALUE >
Node< KEY, VALUE >:: Node (
    const Node< KEY, VALUE > & other ) [inline]
```

Copy constructor. Using assignment operator to perform a deep copy from other to this.

2.2.2.5 ~Node()

```
template<class KEY , class VALUE >
Node< KEY, VALUE >::~ Node ( ) [inline]
```

Destructor. Recursively deleting every node in the list.

Generated by Doxygen

2.2.3 Member Function Documentation

2.2.3.1 add()

```
template<class KEY , class VALUE >
void Node< KEY, VALUE >::add (
    const KEY & key ) [inline]
```

Adds an entry to the list with no predefined value.

2.2.3.2 append()

```
template<class KEY , class VALUE >
void Node< KEY, VALUE >::append (
    const KEY & key,
    const VALUE & value ) [inline]
```

Appends an entry to the list with predefined value.

2.2.3.3 contains()

```
template<class KEY , class VALUE >
bool Node< KEY, VALUE >::contains (
    const KEY & key ) const [inline]
```

Checks if an object with the given key is present in the list. Returns true if it is, false if not.

2.2.3.4 erase()

```
template<class KEY , class VALUE >
bool Node< KEY, VALUE >::erase (
    const KEY & key,
    Node< KEY, VALUE > *& parent_m_next ) [inline]
```

Erases an object from the list.

2.2.3.5 get()

```
template<class KEY , class VALUE >
VALUE & Node< KEY, VALUE >::get (
    const KEY & key,
    VALUE & default_value ) [inline]
```

Looks for an object with the given key, returns it's value if found, returns default value if not.

2.2.3.6 get_at_index()

```
template<class KEY , class VALUE >
VALUE & Node< KEY, VALUE >::get_at_index (
    int & index ) [inline]
```

Iterates through the objects in the list and decrement the index for each element. Returns the value of the current object when the index hits zero.

2.2 Node< KEY, VALUE > Class Template Reference**7****2.2.3.7 operator=()**

```
template<class KEY , class VALUE >
Node< KEY, VALUE > & Node< KEY, VALUE >::operator= (
    const Node< KEY, VALUE > & other ) [inline]
```

Assignment operator overload. Deletes `m_next` if not `nullptr`, and performs a deep copy from `other` to this. Returns a reference to this to allow chaining.

2.2.3.8 set()

```
template<class KEY , class VALUE >
bool Node< KEY, VALUE >::set (
    const KEY & key,
    const VALUE & value ) [inline]
```

Sets the value of an existing object. Appends a new object if no object with the given key already exists.

Index

~LinkedList
 LinkedList< KEY, VALUE >, 2
 ~Node
 Node< KEY, VALUE >, 5
 add
 LinkedList< KEY, VALUE >, 2
 Node< KEY, VALUE >, 6
 append
 LinkedList< KEY, VALUE >, 2
 Node< KEY, VALUE >, 6
 contains
 LinkedList< KEY, VALUE >, 3
 Node< KEY, VALUE >, 6
 count
 LinkedList< KEY, VALUE >, 3
 erase
 LinkedList< KEY, VALUE >, 3
 Node< KEY, VALUE >, 6
 get
 LinkedList< KEY, VALUE >, 3
 Node< KEY, VALUE >, 6
 get_at_index
 Node< KEY, VALUE >, 6
 LinkedList
 LinkedList< KEY, VALUE >, 2
 LinkedList< KEY, VALUE >, 1
 ~LinkedList, 2
 add, 2
 append, 2
 contains, 3
 count, 3
 erase, 3
 get, 3
 LinkedList, 2
 operator=, 3
 operator[], 3
 set, 4
 setDefault, 4
 Node
 Node< KEY, VALUE >, 5
 Node< KEY, VALUE >, 4
 ~Node, 5
 add, 6
 append, 6
 contains, 6
 erase, 6
 get, 6
 get_at_index, 6
 Node, 5
 operator=, 6
 set, 7
 operator=
 LinkedList< KEY, VALUE >, 3
 Node< KEY, VALUE >, 6
 operator[]
 LinkedList< KEY, VALUE >, 3
 set
 LinkedList< KEY, VALUE >, 4
 Node< KEY, VALUE >, 7
 setDefault
 LinkedList< KEY, VALUE >, 4



KONGSBERG

**Bachelor of Engineering****Project appendix****Winter semester 2019****SSSL - API Documentation**

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document describes the code for the Subscriber System Template Library (SSSL) in great detail. It's recommended that it's used as support material for the SSSL - Class Diagram, as it's large and complex on it's own.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------



KONGSBERG

Subscriber System Template Library (SSTL)
V1.0

Generated by Doxygen 1.8.15

1 Hierarchical Index	1
<hr/>	
1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	2
2.1 Class List	2
3 Class Documentation	2
3.1 sstl::data_types::Data< T > Struct Template Reference	2
3.1.1 Detailed Description	2
3.2 sstl::data_types::Data< double > Struct Template Reference	2
3.2.1 Detailed Description	3
3.3 sstl::data_types::Data< float > Struct Template Reference	3
3.3.1 Detailed Description	3
3.4 sstl::data_types::Data< int > Struct Template Reference	3
3.4.1 Detailed Description	3
3.5 sstl::Data_container< T > Class Template Reference	4
3.5.1 Detailed Description	4
3.5.2 Constructor & Destructor Documentation	4
3.5.3 Member Function Documentation	4
3.6 sstl::Lists Class Reference	5
3.6.1 Detailed Description	6
3.6.2 Constructor & Destructor Documentation	6
3.6.3 Member Function Documentation	6
3.7 sstl::Subscribable Class Reference	8
3.7.1 Detailed Description	8
3.7.2 Constructor & Destructor Documentation	8
3.7.3 Member Function Documentation	8
Index	9

1 Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

sstl::data_types::Data< T >	2
sstl::data_types::Data< double >	2
sstl::data_types::Data< float >	3
sstl::data_types::Data< int >	3
sstl::Lists	5
sstl::Subscribable	8

Generated by Doxygen

2

<code>sstl::Data_container< T ></code>	4
--	---

2 Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<code>sstl::data_types::Data< T ></code>	2
<code>sstl::data_types::Data< double ></code>	2
<code>sstl::data_types::Data< float ></code>	3
<code>sstl::data_types::Data< int ></code>	3
<code>sstl::Data_container< T ></code>	4
<code>sstl::Lists</code>	5
<code>sstl::Subscribable</code>	8

3 Class Documentation

3.1 `sstl::data_types::Data< T >` Struct Template Reference

```
#include <Data_types.h>
```

3.1.1 Detailed Description

```
template<typename T>
struct sstl::data_types::Data< T >
```

Declaration of the generic **Data** (p. 2) struct. The different definitions of this struct holds the data type as well as the value.

3.2 `sstl::data_types::Data< double >` Struct Template Reference

```
#include <Data_types.h>
```

Public Attributes

- const Types **type** = Types::t_double
- double **val**

Generated by Doxygen

3.3 sstl::data_types::Data< float > Struct Template Reference

3

3.2.1 Detailed Description

```
template<>
struct sstl::data_types::Data< double >
```

Data (p. 2) struct for doubles.

3.3 sstl::data_types::Data< float > Struct Template Reference

```
#include <Data_types.h>
```

Public Attributes

- const Types **type** = Types::t_float
- float **val**

3.3.1 Detailed Description

```
template<>
struct sstl::data_types::Data< float >
```

Data (p. 2) struct for floats.

3.4 sstl::data_types::Data< int > Struct Template Reference

```
#include <Data_types.h>
```

Public Attributes

- const Types **type** = Types::t_int
- int **val**

3.4.1 Detailed Description

```
template<>
struct sstl::data_types::Data< int >
```

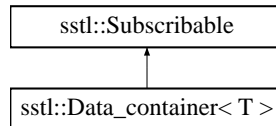
Data (p. 2) struct for integers.

Generated by Doxygen

3.5 sstl::Data_container< T > Class Template Reference

```
#include <Data_container.h>
```

Inheritance diagram for sstl::Data_container< T >:



Public Member Functions

- **Data_container** (const String &name)
- T & **get_data** ()
- const data_types::Types & **get_type** () const override
- const String & **get_name** () const override
- void **set_data** (const T &val)

Private Attributes

- String **m_name**
- data_types::Data< T > **m_data**

3.5.1 Detailed Description

```
template<typename T>
class sstl::Data_container< T >
```

Data_container (p. 4) is a generic class extending the **Subscribable** (p. 8) class. It contains the name of a specific kind of data a sensor can read, as well as the actual data.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 Data_container()

```
template<typename T>
sstl::Data_container< T >::Data_container (
    const String & name ) [inline]
```

Constructor. Setting m_name.

3.5.3 Member Function Documentation

3.6 sstl::Lists Class Reference**5****3.5.3.1 get_data()**

```
template<typename T>
T& sstl::Data_container< T >::get_data ( ) [inline]
```

Returns the value of the m_data object.

3.5.3.2 get_name()

```
template<typename T>
const String& sstl::Data_container< T >::get_name ( ) const [inline], [override], [virtual]
```

Override of **Subscribable** (p. 8)'s get_name method. Returns the data identifier.

Implements **sstl::Subscribable** (p. 8).

3.5.3.3 get_type()

```
template<typename T>
const data_types::Types& sstl::Data_container< T >::get_type ( ) const [inline], [override],
[virtual]
```

Override of **Subscribable** (p. 8)'s get_type method. Returns the type of the m_data object.

Implements **sstl::Subscribable** (p. 8).

3.5.3.4 set_data()

```
template<typename T>
void sstl::Data_container< T >::set_data (
    const T & val ) [inline]
```

Sets the value of m_data.

3.6 sstl::Lists Class Reference

```
#include <Lists.h>
```

Public Member Functions

- virtual **~Lists** ()=0

Generated by Doxygen

Static Public Member Functions

- static bool **subscribe** (const String &key)
- static void **unsubscribe** (const String &key)
- template<typename T >
static void **add_entry** (const String &key)
- static void **append_data** (const String &key, **Subscribable** *data)
- template<typename T >
static bool **set_data** (const String &key, const T &val)
- static **Subscribable** * **get_sub** (const String &key)
- static **Subscribable** * **get_data** (const String &key)
- static LinkedList< String, **Subscribable** * > & **get_sub_list** ()
- static LinkedList< String, **Subscribable** * > & **get_data_list** ()

Static Private Attributes

- static LinkedList< String, **Subscribable** * > **sub_list**
- static LinkedList< String, **Subscribable** * > **data_list**

3.6.1 Detailed Description

This class holds two lists, both containing pointers to a **Subscribable** (p. 8) object. Upon creation of a specific sensor object, the data it's able to read is added to the `data_list`. When data is subscribed to, it get's copied from the `data_list` into the `sub_list`. When unsubscribing to data, it get's erased from the `sub_list`, but remains in the `data_list` to allow resubscription.

3.6.2 Constructor & Destructor Documentation**3.6.2.1 ~Lists()**

```
virtual sstl::Lists::~Lists ( ) [pure virtual]
```

Pure virtual destructor to make the class abstract.

3.6.3 Member Function Documentation**3.6.3.1 add_entry()**

```
template<typename T >  
void sstl::Lists::add_entry (  
    const String & key ) [inline], [static]
```

Adds an empty **Data_container** (p. 4) of the given type to the `data_list` with the given key.

3.6 sstl::Lists Class Reference**7****3.6.3.2 append_data()**

```
void sstl::Lists::append_data (
    const String & key,
    Subscribable * data ) [static]
```

Appends an object to the data_list.

3.6.3.3 get_data()

```
Subscribable * sstl::Lists::get_data (
    const String & key ) [static]
```

Returns a specific object from the data_list.

3.6.3.4 get_data_list()

```
LinkedList< String, Subscribable * > & sstl::Lists::get_data_list ( ) [static]
```

Returns the entire data_list.

3.6.3.5 get_sub()

```
Subscribable * sstl::Lists::get_sub (
    const String & key ) [static]
```

Returns a specific object from the sub_list.

3.6.3.6 get_sub_list()

```
LinkedList< String, Subscribable * > & sstl::Lists::get_sub_list ( ) [static]
```

Returns the entire sub_list.

3.6.3.7 set_data()

```
template<typename T >
bool sstl::Lists::set_data (
    const String & key,
    const T & val ) [inline], [static]
```

Sets the value of an existing entry in the list to the given value of generic type.

3.6.3.8 subscribe()

```
bool sstl::Lists::subscribe (
    const String & key ) [static]
```

Copies an object from the data_list to the sub_list.

Generated by Doxygen

3.6.3.9 unsubscribe()

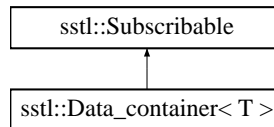
```
void sstl::Lists::unsubscribe (
    const String & key ) [static]
```

Erases an object from the sub_list. The object is still present in the data_list.

3.7 sstl::Subscribable Class Reference

```
#include <Subscribable.h>
```

Inheritance diagram for sstl::Subscribable:



Public Member Functions

- virtual **~Subscribable** ()
- virtual const data_types::Types & **get_type** () const =0
- virtual const String & **get_name** () const =0

3.7.1 Detailed Description

Subscribable (p. 8) is a sort of interface that allow us to store objects of generic type that inherits/extends this interface in e.g. lists.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 ~Subscribable()

```
virtual sstl::Subscribable::~~Subscribable ( ) [inline], [virtual]
```

As other classes will inherit this one, we need a virtual destructor to let the compiler know that a polymorphic object instantiated through a **Subscribable** (p. 8) type might have it's own destructor that has to be called upon deletion.

3.7.3 Member Function Documentation

3.7.3.1 get_name()

```
virtual const String& sstl::Subscribable::get_name ( ) const [pure virtual]
```

Returns the name of the polymorphed object.

Implemented in **sstl::Data_container< T >** (p. 5).

3.7.3.2 get_type()

```
virtual const data_types::Types& sstl::Subscribable::get_type ( ) const [pure virtual]
```

Returns the type of the polymorphed object to be able to downcast to the correct type when needed.

Implemented in **sstl::Data_container< T >** (p. 5).

Index

~Lists
 sstl::Lists, 6
 ~Subscribable
 sstl::Subscribable, 8

 add_entry
 sstl::Lists, 6
 append_data
 sstl::Lists, 6

 Data_container
 sstl::Data_container< T >, 4

 get_data
 sstl::Data_container< T >, 4
 sstl::Lists, 7
 get_data_list
 sstl::Lists, 7
 get_name
 sstl::Data_container< T >, 5
 sstl::Subscribable, 8
 get_sub
 sstl::Lists, 7
 get_sub_list
 sstl::Lists, 7
 get_type
 sstl::Data_container< T >, 5
 sstl::Subscribable, 8

 set_data
 sstl::Data_container< T >, 5
 sstl::Lists, 7
 sstl::Data_container< T >, 4
 Data_container, 4
 get_data, 4
 get_name, 5
 get_type, 5
 set_data, 5
 sstl::data_types::Data< double >, 2
 sstl::data_types::Data< float >, 3
 sstl::data_types::Data< int >, 3
 sstl::data_types::Data< T >, 2
 sstl::Lists, 5
 ~Lists, 6
 add_entry, 6
 append_data, 6
 get_data, 7
 get_data_list, 7
 get_sub, 7
 get_sub_list, 7
 set_data, 7
 subscribe, 7
 unsubscribe, 7
 sstl::Subscribable, 8
 ~Subscribable, 8
 get_name, 8
 get_type, 8
 subscribe
 sstl::Lists, 7
 unsubscribe
 sstl::Lists, 7

Bachelor of Engineering**Project appendix****Winter semester 2019****SatStat Hardware Layer Alpha 1 - API documentation**

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document describes the code for version Alpha 1 in great detail. It's recommended that it's used as support material for the SatStat Hardware Layer Alpha 1 - Class Diagram, as it's large and complex on it's own.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

SatStat Hardware Layer
Alpha 1

Generated by Doxygen 1.8.15

1 Hierarchical Index	1
<hr/>	
1 Hierarchical Index	1
1.1 Class Hierarchy	2
2 Class Index	2
2.1 Class List	2
3 Class Documentation	2
3.1 Input_handler Class Reference	2
3.1.1 Detailed Description	3
3.1.2 Constructor & Destructor Documentation	3
3.1.3 Member Function Documentation	3
3.1.4 Member Data Documentation	4
3.2 Json_handler Class Reference	4
3.2.1 Detailed Description	5
3.2.2 Constructor & Destructor Documentation	5
3.2.3 Member Function Documentation	5
3.2.4 Member Data Documentation	6
3.3 Output_handler Class Reference	6
3.3.1 Detailed Description	7
3.3.2 Constructor & Destructor Documentation	7
3.3.3 Member Function Documentation	7
3.3.4 Member Data Documentation	8
3.4 SatStat_HWLayer Class Reference	9
3.4.1 Detailed Description	9
3.4.2 Constructor & Destructor Documentation	10
3.4.3 Member Function Documentation	10
3.4.4 Member Data Documentation	10
3.5 Sensor Class Reference	11
3.5.1 Detailed Description	11
3.5.2 Constructor & Destructor Documentation	11
3.5.3 Member Function Documentation	12
3.5.4 Member Data Documentation	12
3.6 Temp_hum_sensor Class Reference	12
3.6.1 Detailed Description	13
3.6.2 Constructor & Destructor Documentation	13
3.6.3 Member Function Documentation	13
3.6.4 Member Data Documentation	13
Index	15

1 Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Input_handler	2
Json_handler	4
Output_handler	6
SatStat_HWLayer	9
Sensor	11
Temp_hum_sensor	12

2 Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Input_handler	2
Json_handler	4
Output_handler	6
SatStat_HWLayer	9
Sensor	11
Temp_hum_sensor	12

3 Class Documentation

3.1 Input_handler Class Reference

```
#include <Input_handler.h>
```

Public Member Functions

- **Input_handler** ()
- void **serial_listener** ()
- JsonObject * **read_sensor** (String name)
- JsonObject * **read_sensors** ()
- JsonObject * **get_instruction** ()
- String **get_sensor_list** ()

3.1 Input_handler Class Reference**3**

Private Attributes

- **Json_handler json_handler**
- `LinkedList< String, Sensor * >` **sensor_collection**

3.1.1 Detailed Description

The **Input_handler** (p.2) class is responsible for handling both the serial input, and the input from the different sensors. It has methods for listening to the serial port, reading sensors, inserting and fetching instructions and providing a list of available sensor data.

3.1.2 Constructor & Destructor Documentation**3.1.2.1 Input_handler()**

```
Input_handler::Input_handler ( )
```

Instantiate sensors and append to collection.

3.1.3 Member Function Documentation**3.1.3.1 serial_listener()**

```
void Input_handler::serial_listener ( )
```

Listens on inputs from software layer, appends to instruction queue when data received.

3.1.3.2 read_sensor()

```
JsonObject * Input_handler::read_sensor (
    String name )
```

Returns the read value of the given sensor as a JsonObject.

3.1.3.3 read_sensors()

```
JsonObject * Input_handler::read_sensors ( )
```

Returns the read value of every sensor in the collection as a JsonObject.

3.1.3.4 get_instruction()

```
JsonObject * Input_handler::get_instruction ( )
```

Returns the first instruction in the queue as JsonObject.

Generated by Doxygen

3.1.3.5 get_sensor_list()

```
String Input_handler::get_sensor_list ( )
```

Returns a list of every available sensor as a String.

3.1.4 Member Data Documentation

3.1.4.1 json_handler

```
Json_handler Input_handler::json_handler [private]
```

3.1.4.2 sensor_collection

```
LinkedList<String, Sensor*> Input_handler::sensor_collection [private]
```

The documentation for this class was generated from the following files:

- handlers/Input_handler.h
- handlers/Input_handler.cpp

3.2 Json_handler Class Reference

```
#include <Json_handler.h>
```

Public Member Functions

- **Json_handler** ()
- **~Json_handler** ()
- void **insert_instruction** (String input_data)
- JsonObject * **fetch_instruction** ()
- JsonObject * **convert_to_json** (String key, String value)
- JsonObject * **convert_to_json** (String formatted_string)
- bool **queue_is_empty** ()

Private Attributes

- QueueArray< JsonObject * > **instruction_queue**
- DynamicJsonBuffer * **jsonBuffer**
- JsonObject * **root**
- String **json**

3.2 **Json_handler Class Reference**

5

3.2.1 Detailed Description

The **Json_handler** (p. 4) class is the class that holds the instruction queue, as well as providing the functionality for converting data to JSON format. The class has methods for inserting and fetching instructions from the queue, checking if the queue is empty, and converting to JSON format.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 **Json_handler()**

```
Json_handler::Json_handler ( )
```

Instantiating jsonBuffer and the root JsonObject.

3.2.2.2 **~Json_handler()**

```
Json_handler::~~Json_handler ( )
```

Deletes member variables of pointer type.

3.2.3 Member Function Documentation

3.2.3.1 **insert_instruction()**

```
void Json_handler::insert_instruction (
    String input_data )
```

Converts the passed String argument to a JsonObject, and inserts it in the instruction queue.

3.2.3.2 **fetch_instruction()**

```
JsonObject * Json_handler::fetch_instruction ( )
```

Returns the first instruction in the queue as a JsonObject pointer.

3.2.3.3 **convert_to_json()** [1/2]

```
JsonObject * Json_handler::convert_to_json (
    String key,
    String value )
```

Converts both arguments (key-value pair) to a JsonObject.

Generated by Doxygen

3.2.3.4 `convert_to_json()` [2/2]

```
JsonObject * Json_handler::convert_to_json (
    String formatted_string )
```

Parses the argument to a JsonObject.

3.2.3.5 `queue_is_empty()`

```
bool Json_handler::queue_is_empty ( )
```

Returns true if the instruction queue is empty, false if not.

3.2.4 Member Data Documentation

3.2.4.1 `instruction_queue`

```
QueueArray<JsonObject*> Json_handler::instruction_queue [private]
```

3.2.4.2 `jsonBuffer`

```
DynamicJsonBuffer* Json_handler::jsonBuffer [private]
```

3.2.4.3 `root`

```
JsonObject* Json_handler::root [private]
```

3.2.4.4 `json`

```
String Json_handler::json [private]
```

The documentation for this class was generated from the following files:

- handlers/Json_handler.h
- handlers/Json_handler.cpp

3.3 `Output_handler` Class Reference

```
#include <Output_handler.h>
```

3.3 Output_handler Class Reference

7

Public Member Functions

- **Output_handler** ()
- void **print_to_serial** (JsonObject *json)
- void **auto_rotate_sadm** ()
- bool **auto_rotate_on** ()
- void **rotate_sadm** (int steps)
- void **rotate_sadm** (float degrees)

Private Attributes

- **Json_handler json_handler**
- Stepper * **stepper**
- int **steps**
- bool **dir**
- bool **auto_rotate_en**
- const int **stepsPerRev** = 32
- const float **factor** = 3.25
- const int **step_limit** = (int)(1024 * factor)

3.3.1 Detailed Description

The **Output_handler** (p. 6) class is the one responsible for controlling the SADM and sending JSON formatted data through the serial port. The class has methods for checking if auto rotate is enabled, automatically rotating the SADM, rotate the SADM a given number of steps or degrees and printing to serial.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Output_handler()

```
Output_handler::Output_handler ( )
```

Instantiates the stepper, and initializes memeber variables.

3.3.3 Member Function Documentation

3.3.3.1 print_to_serial()

```
void Output_handler::print_to_serial (
    JsonObject * json )
```

Prints the JsonObject passed as an argument to the serial.

Generated by Doxygen

3.3.3.2 auto_rotate_sadm()

```
void Output_handler::auto_rotate_sadm ( )
```

Automatically rotates the SADM. Must be continuously called.

3.3.3.3 auto_rotate_on()

```
bool Output_handler::auto_rotate_on ( )
```

Returns true if auto rotate is on, false if not.

3.3.3.4 rotate_sadm() [1/2]

```
void Output_handler::rotate_sadm (
    int steps )
```

Rotates the SADM the passed number of steps.

3.3.3.5 rotate_sadm() [2/2]

```
void Output_handler::rotate_sadm (
    float degrees )
```

Converts from degrees to steps, and rotates the SADM.

3.3.4 Member Data Documentation**3.3.4.1 json_handler**

```
Json_handler Output_handler::json_handler [private]
```

3.3.4.2 stepper

```
Stepper* Output_handler::stepper [private]
```

3.3.4.3 steps

```
int Output_handler::steps [private]
```

3.3.4.4 dir

```
bool Output_handler::dir [private]
```

3.4 SatStat_HWLayer Class Reference**9****3.3.4.5 auto_rotate_en**

```
bool Output_handler::auto_rotate_en [private]
```

3.3.4.6 stepsPerRev

```
const int Output_handler::stepsPerRev = 32 [private]
```

3.3.4.7 factor

```
const float Output_handler::factor = 3.25 [private]
```

3.3.4.8 step_limit

```
const int Output_handler::step_limit = (int)(1024 * factor) [private]
```

The documentation for this class was generated from the following files:

- handlers/Output_handler.h
- handlers/Output_handler.cpp

3.4 SatStat_HWLayer Class Reference

```
#include <SatStat_HWLayer.h>
```

Public Member Functions

- **SatStat_HWLayer** ()
- **~SatStat_HWLayer** ()
- void **loop** ()

Private Attributes

- **Input_handler** * **input_handler**
- **Output_handler** * **output_handler**
- unsigned long **start_time** = millis()
- const int **duration** = 1000

3.4.1 Detailed Description

The **SatStat_HWLayer** (p. 9) class utilizes the input and output handler to read sensor data, input and output serial data, and control the SADM.

Generated by Doxygen

3.4.2 Constructor & Destructor Documentation

3.4.2.1 SatStat_HWLayer()

```
SatStat_HWLayer::SatStat_HWLayer ( ) [inline]
```

Initializes the serial port, instantiates input_handler and output_handler and prints the available sensors to serial.

3.4.2.2 ~SatStat_HWLayer()

```
SatStat_HWLayer::~~SatStat_HWLayer ( ) [inline]
```

Deletes every member variable of pointer type.

3.4.3 Member Function Documentation

3.4.3.1 loop()

```
void SatStat_HWLayer::loop ( ) [inline]
```

Automatically rotates the SADM if auto rotate is on, continuously listens to serial and prints sensor data to serial at a given interval.

3.4.4 Member Data Documentation

3.4.4.1 input_handler

```
Input_handler* SatStat_HWLayer::input_handler [private]
```

3.4.4.2 output_handler

```
Output_handler* SatStat_HWLayer::output_handler [private]
```

3.4.4.3 start_time

```
unsigned long SatStat_HWLayer::start_time = millis() [private]
```

3.5 Sensor Class Reference

11

3.4.4.4 duration

```
const int SatStat_HWLayer::duration = 1000 [private]
```

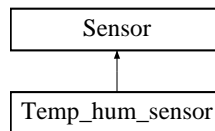
The documentation for this class was generated from the following file:

- SatStat_HWLayer.h

3.5 Sensor Class Reference

```
#include <Sensor.h>
```

Inheritance diagram for Sensor:

**Public Member Functions**

- **Sensor** (String **name**, int **pin**)
- virtual String **read_sensor** ()=0
- String **get_name** ()

Protected Attributes

- String **name**
- int **pin**

3.5.1 Detailed Description

The **Sensor** (p.11) class is the parent of every specific sensor added to the system. This class holds the name of the sensor and the pin it's connected to, and it has a pure virtual `read_sensor` method that has to be overridden for every child class. It also has a method for retrieving the name of the sensor.

3.5.2 Constructor & Destructor Documentation**3.5.2.1 Sensor()**

```
Sensor::Sensor (
    String name,
    int pin )
```

Setting member variables to the values passed as arguments.

Generated by Doxygen

3.5.3 Member Function Documentation

3.5.3.1 read_sensor()

```
virtual String Sensor::read_sensor ( ) [pure virtual]
```

Implemented in **Temp_hum_sensor** (p. 13).

3.5.3.2 get_name()

```
String Sensor::get_name ( )
```

Returns the name of the sensor as a String.

3.5.4 Member Data Documentation

3.5.4.1 name

```
String Sensor::name [protected]
```

3.5.4.2 pin

```
int Sensor::pin [protected]
```

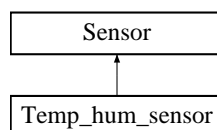
The documentation for this class was generated from the following files:

- sensors/Sensor.h
- sensors/Sensor.cpp

3.6 Temp_hum_sensor Class Reference

```
#include <Temp_hum_sensor.h>
```

Inheritance diagram for Temp_hum_sensor:



3.6 Temp_hum_sensor Class Reference

13

Public Member Functions

- **Temp_hum_sensor** (String *name*, int *pin*)
- String **read_sensor** ()

Private Attributes

- dht **DHT**

Additional Inherited Members

3.6.1 Detailed Description

The **Temp_hum_sensor** (p. 12) class is the "controller" for the DHT11 temperature and humidity sensor. This class is a child of the **Sensor** (p. 11) class, and therefor has to override the `read_sensor` method. `read_sensor` utilizes the DHT library to be able to read the specific sensor and return the result.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 Temp_hum_sensor()

```
Temp_hum_sensor::Temp_hum_sensor (
    String name,
    int pin )
```

Passing arguments to parent constructor.

3.6.3 Member Function Documentation

3.6.3.1 read_sensor()

```
String Temp_hum_sensor::read_sensor ( ) [virtual]
```

Reads the sensor, and return the result as a String.

Implements **Sensor** (p. 12).

3.6.4 Member Data Documentation

3.6.4.1 DHT

```
dht Temp_hum_sensor::DHT [private]
```

The documentation for this class was generated from the following files:

- sensors/Temp_hum_sensor.h
- sensors/Temp_hum_sensor.cpp

Generated by Doxygen

Index

- ~Json_handler
 - Json_handler, 5
- ~SatStat_HWLayer
 - SatStat_HWLayer, 10
- auto_rotate_en
 - Output_handler, 8
- auto_rotate_on
 - Output_handler, 8
- auto_rotate_sadm
 - Output_handler, 7
- convert_to_json
 - Json_handler, 5
- DHT
 - Temp_hum_sensor, 13
- dir
 - Output_handler, 8
- duration
 - SatStat_HWLayer, 10
- factor
 - Output_handler, 9
- fetch_instruction
 - Json_handler, 5
- get_instruction
 - Input_handler, 3
- get_name
 - Sensor, 12
- get_sensor_list
 - Input_handler, 3
- Input_handler, 2
 - get_instruction, 3
 - get_sensor_list, 3
 - Input_handler, 3
 - json_handler, 4
 - read_sensor, 3
 - read_sensors, 3
 - sensor_collection, 4
 - serial_listener, 3
- input_handler
 - SatStat_HWLayer, 10
- insert_instruction
 - Json_handler, 5
- instruction_queue
 - Json_handler, 6
- json
 - Json_handler, 6
- Json_handler, 4
 - ~Json_handler, 5
 - convert_to_json, 5
 - fetch_instruction, 5
 - insert_instruction, 5
 - instruction_queue, 6
 - json, 6
 - json_handler
 - Input_handler, 4
 - Output_handler, 8
 - jsonBuffer
 - Json_handler, 6
 - loop
 - SatStat_HWLayer, 10
 - name
 - Sensor, 12
 - Output_handler, 6
 - auto_rotate_en, 8
 - auto_rotate_on, 8
 - auto_rotate_sadm, 7
 - dir, 8
 - factor, 9
 - json_handler, 8
 - Output_handler, 7
 - print_to_serial, 7
 - rotate_sadm, 8
 - step_limit, 9
 - stepper, 8
 - steps, 8
 - stepsPerRev, 9
 - output_handler
 - SatStat_HWLayer, 10
 - pin
 - Sensor, 12
 - print_to_serial
 - Output_handler, 7
 - queue_is_empty
 - Json_handler, 6
 - read_sensor
 - Input_handler, 3
 - Sensor, 12
 - Temp_hum_sensor, 13
 - read_sensors
 - Input_handler, 3
 - root
 - Json_handler, 6
 - rotate_sadm
 - Output_handler, 8
 - SatStat_HWLayer, 9
 - ~SatStat_HWLayer, 10
 - duration, 10

- input_handler, 10
- loop, 10
- output_handler, 10
- SatStat_HWLayer, 10
- start_time, 10
- Sensor, 11
 - get_name, 12
 - name, 12
 - pin, 12
 - read_sensor, 12
 - Sensor, 11
- sensor_collection
 - Input_handler, 4
- serial_listener
 - Input_handler, 3
- start_time
 - SatStat_HWLayer, 10
- step_limit
 - Output_handler, 9
- stepper
 - Output_handler, 8
- steps
 - Output_handler, 8
- stepsPerRev
 - Output_handler, 9
- Temp_hum_sensor, 12
 - DHT, 13
 - read_sensor, 13
 - Temp_hum_sensor, 13

Bachelor of Engineering
Project appendix
Winter semester 2019

SatStat Hardware Layer Alpha 2 and 3 - API documentation

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document describes the code for versions Alpha 2 and 3 in great detail. It's recommended that it's used as support material for the SatStat Hardware Layer Alpha 2 and 3 - Class Diagram, as it's large and complex on it's own.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

SatStat Hardware Layer
Alpha 2

Generated by Doxygen 1.8.15

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	2
2.1 Class List	2
3 Class Documentation	2
3.1 Instruction_handler Class Reference	2
3.1.1 Detailed Description	3
3.1.2 Constructor & Destructor Documentation	3
3.1.3 Member Function Documentation	3
3.1.4 Member Data Documentation	4
3.2 Json_array_container Class Reference	5
3.2.1 Detailed Description	5
3.2.2 Constructor & Destructor Documentation	5
3.2.3 Member Function Documentation	5
3.3 Json_container< T > Class Template Reference	6
3.3.1 Detailed Description	6
3.3.2 Constructor & Destructor Documentation	6
3.3.3 Member Function Documentation	7
3.3.4 Member Data Documentation	7
3.4 Json_handler Class Reference	8
3.4.1 Detailed Description	8
3.4.2 Member Function Documentation	8
3.5 Json_object_container Class Reference	10
3.5.1 Detailed Description	10
3.5.2 Constructor & Destructor Documentation	10
3.5.3 Member Function Documentation	10
3.6 Result Struct Reference	11
3.6.1 Detailed Description	11
3.6.2 Member Data Documentation	11
3.7 SADM_functions Class Reference	12
3.7.1 Detailed Description	12
3.7.2 Constructor & Destructor Documentation	12
3.7.3 Member Function Documentation	13
3.7.4 Member Data Documentation	14
3.8 SatStat_HWLayer Class Reference	15
3.8.1 Detailed Description	15
3.8.2 Constructor & Destructor Documentation	15
3.8.3 Member Function Documentation	15
3.8.4 Member Data Documentation	16
3.9 Sensor Class Reference	17
3.9.1 Detailed Description	18

1 Hierarchical Index	1
3.9.2 Constructor & Destructor Documentation	18
3.9.3 Member Function Documentation	18
3.9.4 Member Data Documentation	19
3.10 Sensor_container Class Reference	19
3.10.1 Detailed Description	20
3.10.2 Constructor & Destructor Documentation	20
3.10.3 Member Function Documentation	20
3.10.4 Member Data Documentation	21
3.11 Serial_handler Class Reference	21
3.11.1 Detailed Description	22
3.11.2 Constructor & Destructor Documentation	22
3.11.3 Member Function Documentation	22
3.11.4 Member Data Documentation	24
3.12 Temp_hum_sensor Class Reference	25
3.12.1 Detailed Description	25
3.12.2 Constructor & Destructor Documentation	25
3.12.3 Member Function Documentation	26
3.12.4 Member Data Documentation	26
Index	27

1 Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Instruction_handler	2
Json_container< T >	6
Json_container< JSONArray >	6
Json_array_container	5
Json_container< JsonObject >	6
Json_object_container	10
Json_handler	8
Result	11
SADM_functions	12
SatStat_HWLayer	15
Sensor	17
Temp_hum_sensor	25

Generated by Doxygen

2

Sensor_container	19
Serial_handler	21

2 Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Instruction_handler	2
Json_array_container	5
Json_container< T >	6
Json_handler	8
Json_object_container	10
Result	11
SADM_functions	12
SatStat_HWLayer	15
Sensor	17
Sensor_container	19
Serial_handler	21
Temp_hum_sensor	25

3 Class Documentation

3.1 Instruction_handler Class Reference

```
#include <Instruction_handler.h>
```

Public Member Functions

- **Instruction_handler** ()
- **~Instruction_handler** ()
- void **insert_instruction** (const String &input_data)
- **Json_container**< JsonObject > * **fetch_instruction** ()
- bool **queue_is_empty** () const
- bool **sadm_auto_rotate_en** ()
- void **sadm_auto_rotate** ()
- void **interpret_instruction** ()

Generated by Doxygen

3.1 Instruction_handler Class Reference**3****Private Attributes**

- `LinkedList< String, void(*)(Json_container< JsonObject > *)>` **instruction_interpreter**

Static Private Attributes

- `static QueueArray< Json_container< JsonObject > * >` **instruction_queue**

3.1.1 Detailed Description

The **Instruction_handler** (p. 2) class holds the instructions received through the serial port in the `instruction_queue` member. It also has a linked list with supported instruction and a pointer to the function corresponding to the specific instruction. This list works as a lookup table where you look for a specific instruction (key), and receive a pointer to a function (value) responsible for handling the execution of that specific instruction. The main responsibility of this class is to insert, fetch and execute instructions.

3.1.2 Constructor & Destructor Documentation**3.1.2.1 Instruction_handler()**

```
Instruction_handler::Instruction_handler ( )
```

Constructor initializing the stepper, as well as appending instructions and the corresponding functions to the `instruction_interpreter`.

3.1.2.2 ~Instruction_handler()

```
Instruction_handler::~~Instruction_handler ( )
```

Delete every `JsonObject` in the `instruction_queue`.

3.1.3 Member Function Documentation**3.1.3.1 insert_instruction()**

```
void Instruction_handler::insert_instruction (
    const String & input_data )
```

Inserts passed instruction into the `instruction_queue`.

3.1.3.2 fetch_instruction()

```
Json_container< JsonObject > * Instruction_handler::fetch_instruction ( )
```

Fetches an instruction from the instruction queue. Returned as **Json_container**<**JsonObject**> (p. 6) pointer.

Generated by Doxygen

3.1.3.3 queue_is_empty()

```
bool Instruction_handler::queue_is_empty ( ) const
```

Returns true if instruction queue is empty, false if not.

3.1.3.4 sadm_auto_rotate_en()

```
bool Instruction_handler::sadm_auto_rotate_en ( )
```

Returns true if auto rotate is enabled, false if not.

3.1.3.5 sadm_auto_rotate()

```
void Instruction_handler::sadm_auto_rotate ( )
```

Calls the static auto_rotate function from the **SADM_functions** (p. 12) container.

3.1.3.6 interpret_instruction()

```
void Instruction_handler::interpret_instruction ( )
```

Fetches the first instruction in the instruction_queue, getting the corresponding function from the instruction_↔ interpreter and calling the retrieved function passing the instruction as argument.

3.1.4 Member Data Documentation**3.1.4.1 instruction_queue**

```
QueueArray< Json_container< JsonObject > * > Instruction_handler::instruction_queue [static],  
[private]
```

3.1.4.2 instruction_interpreter

```
LinkedList<String, void(*) ( Json_container<JsonObject>*) > Instruction_handler::instruction_↔  
interpreter [private]
```

The documentation for this class was generated from the following files:

- handlers/Instruction_handler.h
- handlers/Instruction_handler.cpp

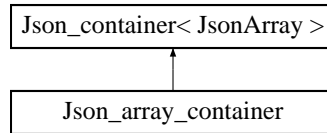
3.2 Json_array_container Class Reference

5

3.2 Json_array_container Class Reference

```
#include <Json_array_container.h>
```

Inheritance diagram for Json_array_container:

**Public Member Functions**

- **Json_array_container** ()
- void **create** ()
- void **parse** (const String & **json**)

Additional Inherited Members**3.2.1 Detailed Description**

Child class of **Json_container** (p. 6) instantiated as **Json_container<JSONArray>** (p. 6). Overrides the create and parse methods, and has a constructor for instantiating the json member variable in the parent class as a JSONArray type.

3.2.2 Constructor & Destructor Documentation**3.2.2.1 Json_array_container()**

```
Json_array_container::Json_array_container ( )
```

Instantiate parent json member as JSONArray.

3.2.3 Member Function Documentation**3.2.3.1 create()**

```
void Json_array_container::create ( ) [virtual]
```

Deletes whatever currently stored in the JsonBuffer, and instantiates parent json member as a new JSONArray.

Implements **Json_container< JSONArray >** (p. 7).

3.2.3.2 parse()

```
void Json_array_container::parse (
    const String & json ) [virtual]
```

Deletes whatever currently stored in the JsonBuffer, and instantiates parent json member as a new JSONArray parsed from the string passed as argument.

Implements **Json_container**< **JSONArray** > (p. 7).

The documentation for this class was generated from the following files:

- containers/Json_array_container.h
- containers/Json_array_container.cpp

3.3 Json_container< T > Class Template Reference

```
#include <Json_container.h>
```

Public Member Functions

- **Json_container** ()
- **~Json_container** ()
- virtual void **create** ()=0
- virtual void **parse** (const String & **json**)=0
- T * **get** ()

Protected Attributes

- DynamicJsonBuffer * **buffer**
- T * **json**

3.3.1 Detailed Description

```
template<class T>
class Json_container< T >
```

Parent class of **Json_object_container** (p. 10) and **Json_array_container** (p. 5). This class is abstract, and forces every child class to override the create and parse methods. The ArduinoJson library requires JsonObject and JsonArrays to be created within a JsonBuffer. When a JsonObject or JsonArray is retrieved from a buffer, it's returned as a reference. This means that neither of those can live outside of the buffer, so this class makes sure that a JsonBuffer only contains one JsonObject or JsonArray for it to be easy to keep track of which buffer is related to what object or array when passed around. This class is generic for it to be specified upon instantiation if it's to hold a JsonObject or JsonArray.

3.3.2 Constructor & Destructor Documentation

3.3 Json_container< T > Class Template Reference**7****3.3.2.1 Json_container()**

```
template<class T >
Json_container< T >:: Json_container ( ) [inline]
```

Instantiating the DynamicJsonBuffer.

3.3.2.2 ~Json_container()

```
template<class T >
Json_container< T >::~~Json_container ( ) [inline]
```

Deletes the DynamicJsonBuffer.

3.3.3 Member Function Documentation**3.3.3.1 create()**

```
template<class T>
virtual void Json_container< T >::create ( ) [pure virtual]
```

Implemented in **Json_array_container** (p.5), and **Json_object_container** (p.10).

3.3.3.2 parse()

```
template<class T>
virtual void Json_container< T >::parse (
    const String & json ) [pure virtual]
```

Implemented in **Json_array_container** (p.5), and **Json_object_container** (p.10).

3.3.3.3 get()

```
template<class T >
T * Json_container< T >::get ( ) [inline]
```

Returns the generic json data, either a JsonObject or JSONArray depending what the object was instantiated as.

3.3.4 Member Data Documentation

Generated by Doxygen

3.3.4.1 buffer

```
template<class T>
DynamicJsonBuffer* Json_container< T >::buffer [protected]
```

3.3.4.2 json

```
template<class T>
T* Json_container< T >::json [protected]
```

The documentation for this class was generated from the following file:

- containers/Json_container.h

3.4 Json_handler Class Reference

```
#include <Json_handler.h>
```

Public Member Functions

- **Json_container**< JsonObject > * **create_object** ()
- **Json_container**< JsonArray > * **create_array** ()
- template<class T >
Json_container< JsonObject > * **create_object** (const String &key, const T &value)
- template<class T >
Json_container< JsonArray > * **create_array** (const T *value, const int &data_count)
- template<class T >
void **append_to** (**Json_container**< JsonObject > *obj, const String &key, const T &value)
- template<class T >
void **append_to** (**Json_container**< JsonArray > *arr, const String &key, const T &value)

3.4.1 Detailed Description

The **Json_handler** (p. 8) class provides functionality for creating and appending to JsonObjects and/or JsonArrays. This class have no member variables, as it operates upon objects stored in **Json_container** (p. 6) objects.

3.4.2 Member Function Documentation

3.4.2.1 create_object() [1/2]

```
Json_container< JsonObject > * Json_handler::create_object ( )
```

Creates and returns a pointer to a **Json_object_container** (p. 10) without initial key-value pair.

3.4 Json_handler Class Reference**9****3.4.2.2 create_array()** [1/2]

```
Json_container< JSONArray > * Json_handler::create_array ( )
```

Creates and returns a pointer to a **Json_array_container** (p. 5) without initial key-value pair.

3.4.2.3 create_object() [2/2]

```
template<class T >
Json_container< JsonObject > * Json_handler::create_object (
    const String & key,
    const T & value ) [inline]
```

Creates a pointer to a **Json_object_container** (p. 10) with predefined key and value.

3.4.2.4 create_array() [2/2]

```
template<class T >
Json_container< JSONArray > * Json_handler::create_array (
    const T * value,
    const int & data_count ) [inline]
```

Creates a pointer to a **Json_array_container** (p. 5) with predefined key and value.

3.4.2.5 append_to() [1/2]

```
template<class T >
void Json_handler::append_to (
    Json_container< JsonObject > * obj,
    const String & key,
    const T & value ) [inline]
```

Appends a key-value pair to an existing object.

3.4.2.6 append_to() [2/2]

```
template<class T >
void Json_handler::append_to (
    Json_container< JSONArray > * arr,
    const String & key,
    const T & value ) [inline]
```

Appends a key-value pair to an existing array.

The documentation for this class was generated from the following files:

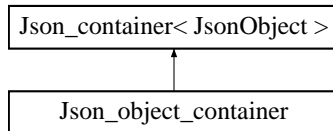
- handlers/Json_handler.h
- handlers/Json_handler.cpp

Generated by Doxygen

3.5 Json_object_container Class Reference

```
#include <Json_object_container.h>
```

Inheritance diagram for Json_object_container:



Public Member Functions

- **Json_object_container** ()
- void **create** ()
- void **parse** (const String & **json**)

Additional Inherited Members

3.5.1 Detailed Description

Child class of **Json_container** (p.6) instantiated as **Json_container<JsonObject>** (p.6). Overrides the create and parse methods, and has a constructor for instantiating the json member variable in the parent class as a JsonObject type.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 Json_object_container()

```
Json_object_container::Json_object_container ( )
```

Instantiate parent json member as JsonObject.

3.5.3 Member Function Documentation

3.5.3.1 create()

```
void Json_object_container::create ( ) [virtual]
```

Deletes whatever currently stored in the JsonBuffer, and instantiates parent json member as a new JsonObject.

Implements **Json_container<JsonObject >** (p.7).

3.6 Result Struct Reference

11

3.5.3.2 parse()

```
void Json_object_container::parse (
    const String & json ) [virtual]
```

Deletes whatever currently stored in the JsonBuffer, and instantiates parent json member as a new JsonObject parsed from the string passed as argument.

Implements **Json_container**< **JsonObject** > (p. 7).

The documentation for this class was generated from the following files:

- containers/Json_object_container.h
- containers/Json_object_container.cpp

3.6 Result Struct Reference

```
#include <Result.h>
```

Public Attributes

- String **name**
- int **data**

3.6.1 Detailed Description

Result (p. 11) is a struct containing the name of a specific kind of data a sensor can read, as well as the read data. The reason for this struct is that one sensor might be able to measure more than one thing, so by adding multiple **Result** (p. 11) objects to that sensor would make it possible to name the different measurements, and contain the read values in a sensible way.

3.6.2 Member Data Documentation

3.6.2.1 name

```
String Result::name
```

3.6.2.2 data

```
int Result::data
```

The documentation for this struct was generated from the following file:

- sensors/Result.h

Generated by Doxygen

3.7 SADM_functions Class Reference

```
#include <SADM_functions.h>
```

Public Member Functions

- virtual `~SADM_functions()`=0

Static Public Member Functions

- static void `init_stepper()`
- static void `set_auto_rotate (Json_container<JsonObject> *instruction)`
- static void `auto_rotate()`
- static void `rotate (int steps)`
- static void `rotate (float degrees)`
- static void `rotate (Json_container<JsonObject> *instruction)`
- static bool `get_auto_rotate_en()`

Static Private Attributes

- static int `steps = 0`
- static bool `dir = false`
- static bool `auto_rotate_en = false`
- static const int `stepsPerRev = 32`
- static const float `factor = 3.25`
- static const int `step_limit = (int)(1024 * factor)`
- static Stepper * `stepper = new Stepper(stepsPerRev, 8, 10, 9, 11)`

3.7.1 Detailed Description

Abstract class with all static members for the functions to be compatible with insertion into the `instruction_interpreter` list in the `Instruction_handler` (p. 2). The `instruction_interpreter` list takes a general function pointer as value for it to be able to contain functions for every device to be controlled. For a class' member functions to be treated as functions rather than methods of a class, they have to be static, hence the structure of this class. This class is responsible for controlling the the SADM, and has member functions for rotating in different ways by providing different arguments.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 ~SADM_functions()

```
virtual SADM_functions::~SADM_functions () [pure virtual]
```

Pure virtual destructor to make the class abstract.

3.7 SADM_functions Class Reference**13****3.7.3 Member Function Documentation****3.7.3.1 init_stepper()**

```
void SADM_functions::init_stepper ( ) [static]
```

Sets the speed of the stepper motor to a fixed speed (700 in this case). The number used to set the speed is usually defined as RPM, but in this case the motor is set up in way so that it's no longer RPM.

3.7.3.2 set_auto_rotate()

```
void SADM_functions::set_auto_rotate (
    Json_container< JsonObject > * instruction ) [static]
```

Set's the auto_rotate_en member either true or false depending on the instruction parameter.

3.7.3.3 auto_rotate()

```
void SADM_functions::auto_rotate ( ) [static]
```

Rotates the SADM one step each time it's called. Automatically rotates the SADM when continuously called.

3.7.3.4 rotate() [1/3]

```
void SADM_functions::rotate (
    int steps ) [static]
```

Rotates the SADM the passed number of steps.

3.7.3.5 rotate() [2/3]

```
void SADM_functions::rotate (
    float degrees ) [static]
```

Converts from degrees to steps, and rotates the SADM.

3.7.3.6 rotate() [3/3]

```
void SADM_functions::rotate (
    Json_container< JsonObject > * instruction ) [static]
```

Calls the rotate function matching the instruction parameter type.

3.7.3.7 get_auto_rotate_en()

```
bool SADM_functions::get_auto_rotate_en ( ) [static]
```

Returns true if auto rotate is enabled, false if not.

Generated by Doxygen

3.7.4 Member Data Documentation

3.7.4.1 steps

```
int SADM_functions::steps = 0 [static], [private]
```

3.7.4.2 dir

```
bool SADM_functions::dir = false [static], [private]
```

3.7.4.3 auto_rotate_en

```
bool SADM_functions::auto_rotate_en = false [static], [private]
```

3.7.4.4 stepsPerRev

```
const int SADM_functions::stepsPerRev = 32 [static], [private]
```

3.7.4.5 factor

```
const float SADM_functions::factor = 3.25 [static], [private]
```

3.7.4.6 step_limit

```
const int SADM_functions::step_limit = (int)(1024 * factor) [static], [private]
```

3.7.4.7 stepper

```
Stepper * SADM_functions::stepper = new Stepper( stepsPerRev, 8, 10, 9, 11) [static], [private]
```

The documentation for this class was generated from the following files:

- containers/SADM_functions.h
- containers/SADM_functions.cpp

3.8 SatStat_HWLayer Class Reference

```
#include <SatStat_HWLayer.h>
```

Public Member Functions

- **SatStat_HWLayer** ()
- void **setup** ()
- void **loop** ()

Private Member Functions

- void **handshake** ()
- bool **connection** ()
- bool **connection_init** ()
- bool **provide_sensor_data** ()

Private Attributes

- **Serial_handler serial_handler**
- **Instruction_handler instruction_handler**
- **Sensor_container sensor_container**
- unsigned long **sensor_interval_start_time**
- const unsigned long **sensor_interval_duration** = 1000
- unsigned long **outer_timeout_start_time**
- const unsigned long **outer_timeout_duration** = 10000

3.8.1 Detailed Description

This class acts as the "main" class. Doxygen, the documentation generation tool, doesn't support .ino files, therefore this class has been created to be able to generate documentation for the "main" class. The project still has a .ino file for it to be considered an arduino project, but the setup and loop functions in that file simply calls the corresponding methods in this one. The setup method of this class handles the handshake protocol, where as the loop method handles the receiving of instructions from serial, transmission of sensor data through serial, and execution of instructions.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 SatStat_HWLayer()

```
SatStat_HWLayer::SatStat_HWLayer ( )
```

Sets up Vcc and GND pins for DHT11 temperature and humidity sensor, and initializes the serial port.

3.8.3 Member Function Documentation

3.8.3.1 setup()

```
void SatStat_HWLayer::setup ( ) [inline]
```

Handles the handshake protocol as defined in SatStat communication protocol.

3.8.3.2 loop()

```
void SatStat_HWLayer::loop ( ) [inline]
```

Continuously listens for serial inputs, prints read sensor data to serial at a given interval and execute received instructions.

3.8.3.3 handshake()

```
void SatStat_HWLayer::handshake ( ) [private]
```

Loops until serial handshake is received. No timeout for this method as this is the root of the handshake protocol. If an error occur at any other phase in the procol, it will reset to this point and wait for client to try again.

3.8.3.4 connection()

```
bool SatStat_HWLayer::connection ( ) [private]
```

Loops until connection request is received, or a timeout occur.

3.8.3.5 connection_init()

```
bool SatStat_HWLayer::connection_init ( ) [private]
```

Loops until connection acknowledgement is received, or a timeout occur.

3.8.3.6 provide_sensor_data()

```
bool SatStat_HWLayer::provide_sensor_data ( ) [private]
```

Loops until available sensor data request is received, or a timeout occur.

3.8.4 Member Data Documentation

3.8.4.1 serial_handler

```
Serial_handler SatStat_HWLayer::serial_handler [private]
```

3.8.4.2 instruction_handler

```
Instruction_handler SatStat_HWLayer::instruction_handler [private]
```

3.9 Sensor Class Reference

17

3.8.4.3 sensor_container

```
Sensor_container SatStat_HWLayer::sensor_container [private]
```

3.8.4.4 sensor_interval_start_time

```
unsigned long SatStat_HWLayer::sensor_interval_start_time [private]
```

3.8.4.5 sensor_interval_duration

```
const unsigned long SatStat_HWLayer::sensor_interval_duration = 1000 [private]
```

3.8.4.6 outer_timeout_start_time

```
unsigned long SatStat_HWLayer::outer_timeout_start_time [private]
```

3.8.4.7 outer_timeout_duration

```
const unsigned long SatStat_HWLayer::outer_timeout_duration = 10000 [private]
```

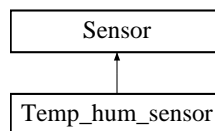
The documentation for this class was generated from the following files:

- SatStat_HWLayer.h
- SatStat_HWLayer.cpp

3.9 Sensor Class Reference

```
#include <Sensor.h>
```

Inheritance diagram for Sensor:

**Public Member Functions**

- **Sensor** (const String & **name**, const int & **pin**, const int & **data_count**=1)
- **~Sensor** ()
- virtual const **Result** * **read_sensor** ()=0
- const String & **get_name** () const
- const int & **get_data_count** () const

Generated by Doxygen

Protected Attributes

- String **name**
- int **pin**
- int **data_count**
- **Result** * **result**

3.9.1 Detailed Description

The **Sensor** (p. 17) class is the parent class of every specific sensor added to the system. The purpose of this class is to ensure that every sensor are of the same type, to be able to store all of them in a collection. This is an abstract class, and it foreces subclasses to override the `read_sensor` function as this is different for every sensor, but is yet required. `get_name` and `get_data_count` are inherited for every child as they do exactly the same for every type of sensor.

3.9.2 Constructor & Destructor Documentation**3.9.2.1 Sensor()**

```
Sensor::Sensor (
    const String & name,
    const int & pin,
    const int & data_count = 1 )
```

Constructor setting name, pin and data_count as well as initializing the **Result** (p. 11) pointer as an array of size provided by the data_count parameter.

3.9.2.2 ~Sensor()

```
Sensor::~Sensor ( )
```

Destructor deleting the **Result** (p. 11) pointer.

3.9.3 Member Function Documentation**3.9.3.1 read_sensor()**

```
virtual const Result* Sensor::read_sensor ( ) [pure virtual]
```

Implemented in **Temp_hum_sensor** (p. 26).

3.10 Sensor_container Class Reference

19

3.9.3.2 get_name()

```
const String & Sensor::get_name ( ) const
```

Returns a constant string reference to the name member.

3.9.3.3 get_data_count()

```
const int & Sensor::get_data_count ( ) const
```

Returns a constant int reference to the data_count member. data_count represents the number of readings the sensor provides.

3.9.4 Member Data Documentation

3.9.4.1 name

```
String Sensor::name [protected]
```

3.9.4.2 pin

```
int Sensor::pin [protected]
```

3.9.4.3 data_count

```
int Sensor::data_count [protected]
```

3.9.4.4 result

```
Result* Sensor::result [protected]
```

The documentation for this class was generated from the following files:

- sensors/Sensor.h
- sensors/Sensor.cpp

3.10 Sensor_container Class Reference

```
#include <Sensor_container.h>
```

Generated by Doxygen

Public Member Functions

- **Sensor_container** ()
- **~Sensor_container** ()
- **Json_container**< JsonObject > * **read_sensor** (const String &name)
- **Json_container**< JsonObject > * **read_sensors** ()
- LinkedList< String, **Sensor** * > & **get_available_sensors** ()

Private Attributes

- LinkedList< String, **Sensor** * > **sensor_collection**
- **Json_handler** **json_handler**

3.10.1 Detailed Description

The **Sensor_container** (p. 19) class holds every available sensor in a linked list, and has methods for retrieving a list of available sensors, as well as reading the different sensors.

3.10.2 Constructor & Destructor Documentation**3.10.2.1 Sensor_container()**

```
Sensor_container::Sensor_container ( )
```

Instantiate sensors with name and the pin it's connected to.

3.10.2.2 ~Sensor_container()

```
Sensor_container::~~Sensor_container ( )
```

Delete every sensor in the sensor collection.

3.10.3 Member Function Documentation**3.10.3.1 read_sensor()**

```
Json_container< JsonObject > * Sensor_container::read_sensor (
    const String & name )
```

Read sensor corresponding to the name sent as argument, and return result as **Json_container**<JsonObject> (p. 6) pointer.

3.11 Serial_handler Class Reference

21

3.10.3.2 read_sensors()

```
Json_container< JsonObject > * Sensor_container::read_sensors ( )
```

Reads all sensors in the sensor collection, and return result as a **Json_container<JsonObject>** (p. 6) pointer.

3.10.3.3 get_available_sensors()

```
LinkedList< String, Sensor * > & Sensor_container::get_available_sensors ( )
```

Returns a reference to the linked list containing every available sensor.

3.10.4 Member Data Documentation**3.10.4.1 sensor_collection**

```
LinkedList<String, Sensor*> Sensor_container::sensor_collection [private]
```

3.10.4.2 json_handler

```
Json_handler Sensor_container::json_handler [private]
```

The documentation for this class was generated from the following files:

- containers/Sensor_container.h
- containers/Sensor_container.cpp

3.11 Serial_handler Class Reference

```
#include <Serial_handler.h>
```

Public Member Functions

- **Serial_handler** ()
- void **serial_init** ()
- void **serial_listener** ()
- void **send_nack** ()
- bool **handshake_approved** ()
- bool **connection_request_approved** ()
- bool **connection_init_approved** ()
- bool **available_data_request_approved** ()
- void **print_to_serial** (**Json_container**< **JsonObject** > *json)

Generated by Doxygen

Private Member Functions

- bool **config_approved** (const unsigned long & **baud_rate**, const String & **config**)
- void **send_handshake_response** ()
- void **send_sensor_collection** ()
- void **send_ack** ()

Private Attributes

- unsigned long **baud_rate**
- String **config**
- String **newline_format**
- **Sensor_container sensor_container**
- **Json_handler json_handler**
- **Instruction_handler instruction_handler**

3.11.1 Detailed Description

The **Serial_handler** (p.21) class is responsible for establishing connection with a client through the serial port, as well as listening for input and printing output through the serial port when a connection has been established. It has methods complementing the handshake protocol defined in the SatStat communication protocol document, and for listening and printing to serial.

3.11.2 Constructor & Destructor Documentation**3.11.2.1 Serial_handler()**

```
Serial_handler::Serial_handler ( )
```

Sets default baud rate, configuration and newline format.

3.11.3 Member Function Documentation**3.11.3.1 serial_init()**

```
void Serial_handler::serial_init ( )
```

Initializes serial with configuration stored in the member variables **baud_rate** and **config**.

3.11.3.2 serial_listener()

```
void Serial_handler::serial_listener ( )
```

Listens for serial input. Has to be continuously called to store the input as soon as it's available.

3.11 Serial_handler Class Reference

23

3.11.3.3 send_nack()

```
void Serial_handler::send_nack ( )
```

Prints JSON formatted negative acknowledgement to serial. Output looks as follows: {"serial_handshake":"failed"}.

3.11.3.4 handshake_approved()

```
bool Serial_handler::handshake_approved ( )
```

Reads serial until handshake received, and sends handshake response according to SatStat communication protocol when it is. Sends NACK and returns false if what's received is not a proper handshake. If no input is received before timeout occurs, it returns false without sending NACK.

3.11.3.5 connection_request_approved()

```
bool Serial_handler::connection_request_approved ( )
```

Reads serial until connection request received, and initializes serial with the received configuration when it is. Sends NACK and returns false if what's received is not a proper request. If no input is received before timeout occurs, it returns false without sending NACK.

3.11.3.6 connection_init_approved()

```
bool Serial_handler::connection_init_approved ( )
```

Reads serial until connection acknowledgement received. Sends NACK and returns false if what's received is not a proper acknowledgement. If no input is received before timeout occurs, it returns false without sending NACK.

3.11.3.7 available_data_request_approved()

```
bool Serial_handler::available_data_request_approved ( )
```

Reads serial until request for available data is received, and responds by sending a list of available sensors when it is. Sends NACK and returns false if what's received is not a proper request. If no input is received before timeout occurs, it returns false without sending NACK.

3.11.3.8 print_to_serial()

```
void Serial_handler::print_to_serial (
    Json_container< JsonObject > * json )
```

Prints the **Json_container**<JsonObject> (p.6) pointer passed as argument to the serial.

3.11.3.9 config_approved()

```
bool Serial_handler::config_approved (
    const unsigned long & baud_rate,
    const String & config ) [private]
```

Checks if the configuration passed as arguments are valid. Returns true if they are, false if one or more of them are not.

Generated by Doxygen

3.11.3.10 send_handshake_response()

```
void Serial_handler::send_handshake_response ( ) [private]
```

Creates a **Json_container**<**JsonObject**> (p.6) pointer, appends the available baud rates, configurations and newline formats and print it to the serial.

3.11.3.11 send_sensor_collection()

```
void Serial_handler::send_sensor_collection ( ) [private]
```

Creates a **Json_container**<**JsonObject**> (p.6) pointer, appends the name and data type of the sensors in the sensor collection and prints it to the serial.

3.11.3.12 send_ack()

```
void Serial_handler::send_ack ( ) [private]
```

Creates a **Json_container**<**JsonObject**> (p.6) pointer, appends the acknowledgement message and prints it to the serial.

3.11.4 Member Data Documentation**3.11.4.1 baud_rate**

```
unsigned long Serial_handler::baud_rate [private]
```

3.11.4.2 config

```
String Serial_handler::config [private]
```

3.11.4.3 newline_format

```
String Serial_handler::newline_format [private]
```

3.11.4.4 sensor_container

```
Sensor_container Serial_handler::sensor_container [private]
```

3.12 Temp_hum_sensor Class Reference

25

3.11.4.5 json_handler

```
Json_handler Serial_handler::json_handler [private]
```

3.11.4.6 instruction_handler

```
Instruction_handler Serial_handler::instruction_handler [private]
```

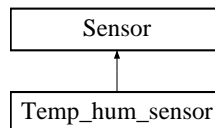
The documentation for this class was generated from the following files:

- handlers/Serial_handler.h
- handlers/Serial_handler.cpp

3.12 Temp_hum_sensor Class Reference

```
#include <Temp_hum_sensor.h>
```

Inheritance diagram for Temp_hum_sensor:



Public Member Functions

- **Temp_hum_sensor** (const String & **name**, const int & **pin**)
- const **Result** * **read_sensor** ()

Private Attributes

- dht **DHT**

Additional Inherited Members

3.12.1 Detailed Description

The **Temp_hum_sensor** (p. 25) is an example of a class for a specific sensor, in this case the DHT11 temperature and humidity sensor. This class inherits the members of the sensor class, and includes the DHT library. It has its own constructor, and overrides the `read_sensor` method as required. This is to support the specific sensor, as well as keeping the same structure as every other sensor to make sure creation and reading of different kinds of sensors work the exact same way.

3.12.2 Constructor & Destructor Documentation

Generated by Doxygen

3.12.2.1 Temp_hum_sensor()

```
Temp_hum_sensor::Temp_hum_sensor (
    const String & name,
    const int & pin )
```

Pass parameter inputs to parent constructor arguments.

3.12.3 Member Function Documentation

3.12.3.1 read_sensor()

```
const Result * Temp_hum_sensor::read_sensor ( ) [virtual]
```

Read all data the sensor can provide, and return as a **Result** (p. 11) pointer.

Implements **Sensor** (p. 18).

3.12.4 Member Data Documentation

3.12.4.1 DHT

```
dht Temp_hum_sensor::DHT [private]
```

The documentation for this class was generated from the following files:

- sensors/Temp_hum_sensor.h
- sensors/Temp_hum_sensor.cpp

Index

- ~Instruction_handler
 - Instruction_handler, 3
- ~Json_container
 - Json_container< T >, 7
- ~SADM_functions
 - SADM_functions, 12
- ~Sensor
 - Sensor, 18
- ~Sensor_container
 - Sensor_container, 20
- append_to
 - Json_handler, 9
- auto_rotate
 - SADM_functions, 13
- auto_rotate_en
 - SADM_functions, 14
- available_data_request_approved
 - Serial_handler, 23
- baud_rate
 - Serial_handler, 24
- buffer
 - Json_container< T >, 7
- config
 - Serial_handler, 24
- config_approved
 - Serial_handler, 23
- connection
 - SatStat_HWLayer, 16
- connection_init
 - SatStat_HWLayer, 16
- connection_init_approved
 - Serial_handler, 23
- connection_request_approved
 - Serial_handler, 23
- create
 - Json_array_container, 5
 - Json_container< T >, 7
 - Json_object_container, 10
- create_array
 - Json_handler, 8, 9
- create_object
 - Json_handler, 8, 9
- data
 - Result, 11
- data_count
 - Sensor, 19
- DHT
 - Temp_hum_sensor, 26
- dir
 - SADM_functions, 14
- factor
 - SADM_functions, 14
- fetch_instruction
 - Instruction_handler, 3
- get
 - Json_container< T >, 7
- get_auto_rotate_en
 - SADM_functions, 13
- get_available_sensors
 - Sensor_container, 21
- get_data_count
 - Sensor, 19
- get_name
 - Sensor, 18
- handshake
 - SatStat_HWLayer, 16
- handshake_approved
 - Serial_handler, 23
- init_stepper
 - SADM_functions, 13
- insert_instruction
 - Instruction_handler, 3
- Instruction_handler, 2
 - ~Instruction_handler, 3
 - fetch_instruction, 3
 - insert_instruction, 3
 - Instruction_handler, 3
 - instruction_interpreter, 4
 - instruction_queue, 4
 - interpret_instruction, 4
 - queue_is_empty, 3
 - sadm_auto_rotate, 4
 - sadm_auto_rotate_en, 4
- instruction_handler
 - SatStat_HWLayer, 16
 - Serial_handler, 25
- instruction_interpreter
 - Instruction_handler, 4
- instruction_queue
 - Instruction_handler, 4
- interpret_instruction
 - Instruction_handler, 4
- json
 - Json_container< T >, 8
- Json_array_container, 5
 - create, 5
 - Json_array_container, 5
 - parse, 5
- Json_container
 - Json_container< T >, 6
- Json_container< T >, 6
 - ~Json_container, 7
- buffer, 7

- create, 7
- get, 7
- json, 8
- Json_container, 6
- parse, 7
- Json_handler, 8
 - append_to, 9
 - create_array, 8, 9
 - create_object, 8, 9
- json_handler
 - Sensor_container, 21
 - Serial_handler, 24
- Json_object_container, 10
 - create, 10
 - Json_object_container, 10
 - parse, 10
- loop
 - SatStat_HWLayer, 16
- name
 - Result, 11
 - Sensor, 19
- newline_format
 - Serial_handler, 24
- outer_timeout_duration
 - SatStat_HWLayer, 17
- outer_timeout_start_time
 - SatStat_HWLayer, 17
- parse
 - Json_array_container, 5
 - Json_container< T >, 7
 - Json_object_container, 10
- pin
 - Sensor, 19
- print_to_serial
 - Serial_handler, 23
- provide_sensor_data
 - SatStat_HWLayer, 16
- queue_is_empty
 - Instruction_handler, 3
- read_sensor
 - Sensor, 18
 - Sensor_container, 20
 - Temp_hum_sensor, 26
- read_sensors
 - Sensor_container, 20
- Result, 11
 - data, 11
 - name, 11
- result
 - Sensor, 19
- rotate
 - SADM_functions, 13
- sadm_auto_rotate
 - Instruction_handler, 4
 - sadm_auto_rotate_en
 - Instruction_handler, 4
 - SADM_functions, 12
 - ~SADM_functions, 12
 - auto_rotate, 13
 - auto_rotate_en, 14
 - dir, 14
 - factor, 14
 - get_auto_rotate_en, 13
 - init_stepper, 13
 - rotate, 13
 - set_auto_rotate, 13
 - step_limit, 14
 - stepper, 14
 - steps, 14
 - stepsPerRev, 14
- SatStat_HWLayer, 15
 - connection, 16
 - connection_init, 16
 - handshake, 16
 - instruction_handler, 16
 - loop, 16
 - outer_timeout_duration, 17
 - outer_timeout_start_time, 17
 - provide_sensor_data, 16
 - SatStat_HWLayer, 15
 - sensor_container, 16
 - sensor_interval_duration, 17
 - sensor_interval_start_time, 17
 - serial_handler, 16
 - setup, 15
- send_ack
 - Serial_handler, 24
- send_handshake_response
 - Serial_handler, 23
- send_nack
 - Serial_handler, 22
- send_sensor_collection
 - Serial_handler, 24
- Sensor, 17
 - ~Sensor, 18
 - data_count, 19
 - get_data_count, 19
 - get_name, 18
 - name, 19
 - pin, 19
 - read_sensor, 18
 - result, 19
 - Sensor, 18
- sensor_collection
 - Sensor_container, 21
- Sensor_container, 19
 - ~Sensor_container, 20
 - get_available_sensors, 21
 - json_handler, 21
 - read_sensor, 20
 - read_sensors, 20

- sensor_collection, 21
- Sensor_container, 20
- sensor_container
 - SatStat_HWLayer, 16
 - Serial_handler, 24
- sensor_interval_duration
 - SatStat_HWLayer, 17
- sensor_interval_start_time
 - SatStat_HWLayer, 17
- Serial_handler, 21
 - available_data_request_approved, 23
 - baud_rate, 24
 - config, 24
 - config_approved, 23
 - connection_init_approved, 23
 - connection_request_approved, 23
 - handshake_approved, 23
 - instruction_handler, 25
 - json_handler, 24
 - newline_format, 24
 - print_to_serial, 23
 - send_ack, 24
 - send_handshake_response, 23
 - send_nack, 22
 - send_sensor_collection, 24
 - sensor_container, 24
 - Serial_handler, 22
 - serial_init, 22
 - serial_listener, 22
- serial_handler
 - SatStat_HWLayer, 16
- serial_init
 - Serial_handler, 22
- serial_listener
 - Serial_handler, 22
- set_auto_rotate
 - SADM_functions, 13
- setup
 - SatStat_HWLayer, 15
- step_limit
 - SADM_functions, 14
- stepper
 - SADM_functions, 14
- steps
 - SADM_functions, 14
- stepsPerRev
 - SADM_functions, 14
- Temp_hum_sensor, 25
 - DHT, 26
 - read_sensor, 26
 - Temp_hum_sensor, 25

Bachelor of Engineering**Project appendix****Winter semester 2019****SatStat Hardware Layer Beta 1 - API documentation**

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document describes the code for version Beta 1 in great detail. It's recommended that it's used as support material for the SatStat Hardware Layer Beta 1 - Class Diagram, as it's large and complex on it's own.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

SatStat Hardware Layer
Beta 1

Generated by Doxygen 1.8.15

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	1
2.1 Class List	1
3 Class Documentation	2
3.1 HWLayer Class Reference	2
3.1.1 Detailed Description	3
3.1.2 Member Function Documentation	3
3.2 Instruction_handler Class Reference	4
3.2.1 Detailed Description	4
3.2.2 Constructor & Destructor Documentation	4
3.2.3 Member Function Documentation	5
3.3 Json_array_container Class Reference	6
3.3.1 Detailed Description	6
3.3.2 Constructor & Destructor Documentation	6
3.3.3 Member Function Documentation	6
3.4 Json_container< T > Class Template Reference	7
3.4.1 Detailed Description	7
3.4.2 Constructor & Destructor Documentation	7
3.4.3 Member Function Documentation	8
3.5 Json_handler Class Reference	8
3.5.1 Detailed Description	8
3.5.2 Member Function Documentation	9
3.6 Json_object_container Class Reference	10
3.6.1 Detailed Description	10
3.6.2 Constructor & Destructor Documentation	10
3.6.3 Member Function Documentation	10
3.7 SADM_functions Class Reference	11
3.7.1 Detailed Description	11
3.7.2 Constructor & Destructor Documentation	12
3.7.3 Member Function Documentation	12
3.8 Sensor Class Reference	13
3.8.1 Detailed Description	13
3.8.2 Constructor & Destructor Documentation	14
3.8.3 Member Function Documentation	14
3.9 Sensor_container Class Reference	14
3.9.1 Detailed Description	15
3.9.2 Constructor & Destructor Documentation	15
3.9.3 Member Function Documentation	15
3.10 Serial_handler Class Reference	16
3.10.1 Detailed Description	17

1 Hierarchical Index	1
3.10.2 Constructor & Destructor Documentation	17
3.10.3 Member Function Documentation	17
3.11 Subscriber_functions Class Reference	19
3.11.1 Detailed Description	19
3.11.2 Member Function Documentation	19
3.12 Temp_hum_sensor Class Reference	20
3.12.1 Detailed Description	20
3.12.2 Constructor & Destructor Documentation	21
3.12.3 Member Function Documentation	21
Index	23

1 Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

HWLayer	2
Instruction_handler	4
Json_container< T >	7
Json_container< JSONArray >	7
Json_array_container	6
Json_container< JsonObject >	7
Json_object_container	10
Json_handler	8
SADM_functions	11
Sensor	13
Temp_hum_sensor	20
Sensor_container	14
Serial_handler	16
Subscriber_functions	19

2 Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Generated by Doxygen

HWLayer	2
Instruction_handler	4
Json_array_container	6
Json_container< T >	7
Json_handler	8
Json_object_container	10
SADM_functions	11
Sensor	13
Sensor_container	14
Serial_handler	16
Subscriber_functions	19
Temp_hum_sensor	20

3 Class Documentation

3.1 HWLayer Class Reference

```
#include <HWLayer.h>
```

Public Member Functions

- void [setup](#) ()
- void [loop](#) ()

Private Member Functions

- void [handshake](#) ()
- bool [connection](#) ()
- bool [connection_init](#) ()
- bool [provide_sensor_data](#) ()

Private Attributes

- [Serial_handler](#) **serial_handler**
- [Instruction_handler](#) **instruction_handler**
- [Sensor_container](#) **sensor_container**
- unsigned long **sensor_interval_start_time**
- const unsigned long **sensor_interval_duration** = 1000
- unsigned long **outer_timeout_start_time**
- const unsigned long **outer_timeout_duration** = 10000

Generated by Doxygen

3.1 HWLayer Class Reference

3

3.1.1 Detailed Description

This class acts as the "main" class. Doxygen, the documentation generation tool, doesn't support .ino files, therefore this class has been created to be able to generate documentation for the "main" class. The project still has a .ino file for it to be considered an arduino project, but the setup and loop functions in that file simply calls the corresponding methods in this one. The setup method of this class handles the handshake protocol, where as the loop method handles the receiving of instructions from serial, transmission of sensor data through serial, and execution of instructions.

3.1.2 Member Function Documentation

3.1.2.1 setup()

```
void HWLayer::setup ( )
```

Sets up Vcc and GND pins for DHT11 temperature and humidity sensor, and initializes the serial port. Also handles the handshake protocol as defined in SatStat communication protocol.

3.1.2.2 loop()

```
void HWLayer::loop ( )
```

Continuously listens for serial inputs, prints read sensor data to serial at a given interval and execute received instructions.

3.1.2.3 handshake()

```
void HWLayer::handshake ( ) [private]
```

Loops until serial handshake is received. No timeout for this method as this is the root of the handshake protocol. If an error occur at any other phase in the procol, it will reset to this point and wait for client to try again.

3.1.2.4 connection()

```
bool HWLayer::connection ( ) [private]
```

Loops until connection request is received, or a timeout occur.

3.1.2.5 connection_init()

```
bool HWLayer::connection_init ( ) [private]
```

Loops until connection acknowledgement is received, or a timeout occur.

3.1.2.6 provide_sensor_data()

```
bool HWLayer::provide_sensor_data ( ) [private]
```

Loops until available sensor data request is received, or a timeout occur.

The documentation for this class was generated from the following files:

- HWLayer.h
- HWLayer.cpp

Generated by Doxygen

3.2 Instruction_handler Class Reference

```
#include <Instruction_handler.h>
```

Public Member Functions

- [Instruction_handler](#) ()
- [~Instruction_handler](#) ()
- bool [insert_instruction](#) (const String &input_data)
- [Json_container](#)< JsonObject > * [fetch_instruction](#) ()
- bool [queue_is_empty](#) () const
- bool [sadm_auto_rotate_en](#) ()
- void [sadm_auto_rotate](#) ()
- void [interpret_instruction](#) ()

Private Attributes

- [LinkedList](#)< String, void(*)([Json_container](#)< JsonObject > *)> **instruction_interpreter**

Static Private Attributes

- static [QueueArray](#)< [Json_container](#)< JsonObject > * > **instruction_queue**

3.2.1 Detailed Description

The [Instruction_handler](#) class holds the instructions received through the serial port in the `instruction_queue` member. It also has a linked list with supported instruction and a pointer to the function corresponding to the specific instruction. This list works as a lookup table where you look for a specific instruction (key), and receive a pointer to a function (value) responsible for handling the execution of that specific instruction. The main responsibility of this class is to insert, fetch and execute instructions.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 [Instruction_handler](#)()

```
Instruction_handler::Instruction_handler ( )
```

Constructor initializing the stepper, as well as appending instructions and the corresponding functions to the `instruction_interpreter`.

3.2.2.2 [~Instruction_handler](#)()

```
Instruction_handler::~~Instruction_handler ( )
```

Delete every `JsonObject` in the `instruction_queue`.

3.2 Instruction_handler Class Reference

5

3.2.3 Member Function Documentation

3.2.3.1 insert_instruction()

```
bool Instruction_handler::insert_instruction (
    const String & input_data )
```

Inserts passed instruction into the instruction_queue. Returns true if success, false if not.

3.2.3.2 fetch_instruction()

```
Json_container< JsonObject > * Instruction_handler::fetch_instruction ( )
```

Fetches an instruction from the instruction queue. Returned as `Json_container<JsonObject>` pointer.

3.2.3.3 queue_is_empty()

```
bool Instruction_handler::queue_is_empty ( ) const
```

Returns true if instruction queue is empty, false if not.

3.2.3.4 sadm_auto_rotate_en()

```
bool Instruction_handler::sadm_auto_rotate_en ( )
```

Returns true if auto rotate is enabled, false if not.

3.2.3.5 sadm_auto_rotate()

```
void Instruction_handler::sadm_auto_rotate ( )
```

Calls the static auto_rotate function from the `SADM_functions` container.

3.2.3.6 interpret_instruction()

```
void Instruction_handler::interpret_instruction ( )
```

Fetches the first instruction in the instruction_queue, getting the corresponding function from the instruction_↔ interpreter and calling the retrieved function passing the instruction as argument. Returns true if success, false if not.

The documentation for this class was generated from the following files:

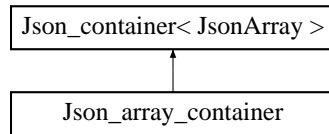
- handlers/Instruction_handler.h
- handlers/Instruction_handler.cpp

Generated by Doxygen

3.3 Json_array_container Class Reference

```
#include <Json_array_container.h>
```

Inheritance diagram for `Json_array_container`:



Public Member Functions

- [Json_array_container](#) ()
- void [create](#) ()
- bool [parse](#) (const String &json)

Additional Inherited Members

3.3.1 Detailed Description

Child class of [Json_container](#) instantiated as `Json_container<JSONArray>`. Overrides the create and parse methods, and has a constructor for instantiating the json member variable in the parent class as a JSONArray type.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 `Json_array_container()`

```
Json_array_container::Json_array_container ( )
```

Instantiate parent json member as JSONArray.

3.3.3 Member Function Documentation

3.3.3.1 `create()`

```
void Json_array_container::create ( ) [virtual]
```

Deletes whatever currently stored in the JsonBuffer, and instantiates parent json member as a new JSONArray.

Implements [Json_container< JSONArray >](#).

3.4 Json_container< T > Class Template Reference**7****3.3.3.2 parse()**

```
bool Json_array_container::parse (
    const String & json ) [virtual]
```

Deletes whatever currently stored in the JsonBuffer, and instantiates parent json member as a new JsonArray parsed from the string passed as argument. Returns true if success, false if not.

Implements [Json_container< JsonArray >](#).

The documentation for this class was generated from the following files:

- containers/Json_array_container.h
- containers/Json_array_container.cpp

3.4 Json_container< T > Class Template Reference

```
#include <Json_container.h>
```

Public Member Functions

- [Json_container](#) ()
- virtual [~Json_container](#) ()
- virtual void **create** ()=0
- virtual bool **parse** (const String &json)=0
- T & [get](#) () const

Protected Attributes

- DynamicJsonBuffer * **buffer**
- T * **json**

3.4.1 Detailed Description

```
template<class T>
class Json_container< T >
```

Parent class of [Json_object_container](#) and [Json_array_container](#). This class is abstract, and forces every child class to override the create and parse methods. The ArduinoJson library requires JsonObject and JsonArrays to be created within a JsonBuffer. When a JsonObject or JsonArray is retrieved from a buffer, it's returned as a reference. This means that neither of those can live outside of the buffer, so this class makes sure that a JsonBuffer only contains one JsonObject or JsonArray for it to be easy to keep track of which buffer is related to what object or array when passed around. This class is generic for it to be specified upon instantiation if it's to hold a JsonObject or JsonArray.

3.4.2 Constructor & Destructor Documentation

Generated by Doxygen

3.4.2.1 `Json_container()`

```
template<class T >
Json_container< T >::Json_container ( ) [inline]
```

Instantiating the DynamicJsonBuffer.

3.4.2.2 `~Json_container()`

```
template<class T >
Json_container< T >::~~Json_container ( ) [inline], [virtual]
```

Deletes the DynamicJsonBuffer.

3.4.3 Member Function Documentation

3.4.3.1 `get()`

```
template<class T >
T & Json_container< T >::get ( ) const [inline]
```

Returns the generic json data, either a JsonObject or JsonArray depending what the object was instantiated as.

The documentation for this class was generated from the following file:

- containers/Json_container.h

3.5 `Json_handler` Class Reference

```
#include <Json_handler.h>
```

Public Member Functions

- `Json_container< JsonObject > * create_object ()`
- `Json_container< JsonArray > * create_array ()`
- `template<class T >`
`Json_container< JsonObject > * create_object (const String &key, const T &value)`
- `template<class T >`
`Json_container< JsonArray > * create_array (const T *value, const int &data_count)`
- `template<class T >`
`void append_to (Json_container< JsonObject > *obj, const String &key, const T &value)`
- `template<class T >`
`void append_to (Json_container< JsonArray > *arr, const String &key, const T &value)`

3.5.1 Detailed Description

The `Json_handler` class provides functionality for creating and appending to JsonObjects and/or JsonArrays. This class have no member variables, as it operates upon objects stored in `Json_container` objects.

3.5 Json_handler Class Reference**9****3.5.2 Member Function Documentation****3.5.2.1 create_object()** [1/2]

```
Json_container< JsonObject > * Json_handler::create_object ( )
```

Creates and returns a pointer to a [Json_object_container](#) without initial key-value pair.

3.5.2.2 create_array() [1/2]

```
Json_container< JsonArray > * Json_handler::create_array ( )
```

Creates and returns a pointer to a [Json_array_container](#) without initial key-value pair.

3.5.2.3 create_object() [2/2]

```
template<class T >
Json_container< JsonObject > * Json_handler::create_object (
    const String & key,
    const T & value ) [inline]
```

Creates a pointer to a [Json_object_container](#) with predefined key and value.

3.5.2.4 create_array() [2/2]

```
template<class T >
Json_container< JsonArray > * Json_handler::create_array (
    const T * value,
    const int & data_count ) [inline]
```

Creates a pointer to a [Json_array_container](#) with predefined key and value.

3.5.2.5 append_to() [1/2]

```
template<class T >
void Json_handler::append_to (
    Json_container< JsonObject > * obj,
    const String & key,
    const T & value ) [inline]
```

Appends a key-value pair to an existing object.

3.5.2.6 append_to() [2/2]

```
template<class T >
void Json_handler::append_to (
    Json_container< JsonArray > * arr,
    const String & key,
    const T & value ) [inline]
```

Appends a key-value pair to an existing array.

The documentation for this class was generated from the following files:

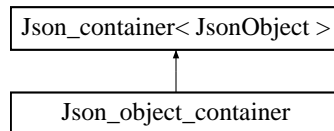
- handlers/Json_handler.h
- handlers/Json_handler.cpp

Generated by Doxygen

3.6 Json_object_container Class Reference

```
#include <Json_object_container.h>
```

Inheritance diagram for `Json_object_container`:



Public Member Functions

- [Json_object_container](#) ()
- void [create](#) ()
- bool [parse](#) (const String &json)

Additional Inherited Members

3.6.1 Detailed Description

Child class of [Json_container](#) instantiated as [Json_container<JsonObject>](#). Overrides the create and parse methods, and has a constructor for instantiating the json member variable in the parent class as a JsonObject type.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 Json_object_container()

```
Json_object_container::Json_object_container ( )
```

Instantiate parent json member as JsonObject.

3.6.3 Member Function Documentation

3.6.3.1 create()

```
void Json_object_container::create ( ) [virtual]
```

Deletes whatever currently stored in the JsonBuffer, and instantiates parent json member as a new JsonObject.

Implements [Json_container<JsonObject>](#).

3.7 SADM_functions Class Reference

11

3.6.3.2 parse()

```
bool Json_object_container::parse (
    const String & json ) [virtual]
```

Deletes whatever currently stored in the JsonBuffer, and instantiates parent json member as a new JsonObject parsed from the string passed as argument. Returns true if success, false if not.

Implements [Json_container< JsonObject >](#).

The documentation for this class was generated from the following files:

- containers/Json_object_container.h
- containers/Json_object_container.cpp

3.7 SADM_functions Class Reference

```
#include <SADM_functions.h>
```

Public Member Functions

- virtual [~SADM_functions](#) ()=0

Static Public Member Functions

- static void [init_stepper](#) ()
- static void [set_auto_rotate](#) ([Json_container](#)< JsonObject > *instruction)
- static void [auto_rotate](#) ()
- static void [rotate](#) (int steps)
- static void [rotate](#) (float degrees)
- static void [rotate](#) ([Json_container](#)< JsonObject > *instruction)
- static bool [get_auto_rotate_en](#) ()

Static Private Attributes

- static int **steps** = 0
- static bool **dir** = false
- static bool **auto_rotate_en** = false
- static const int **stepsPerRev** = 32
- static const float **factor** = 3.25
- static const int **step_limit** = (int)(1024 * factor)
- static Stepper * **stepper** = new Stepper(stepsPerRev, 8, 10, 9, 11)

3.7.1 Detailed Description

Abstract class with all static members for the functions to be compatible with insertion into the instruction_interpreter list in the [Instruction_handler](#). The instruction_interpreter list takes a general function pointer as value for it to be able to contain functions for every device to be controlled. For a class' member functions to be treated as functions rather than methods of a class, they have to be static, hence the structure of this class. This class is responsible for controlling the the SADM, and has member functions for rotating in different ways by providing different arguments.

Generated by Doxygen

3.7.2 Constructor & Destructor Documentation

3.7.2.1 ~SADM_functions()

```
virtual SADM_functions::~~SADM_functions ( ) [pure virtual]
```

Pure virtual destructor to make the class abstract.

3.7.3 Member Function Documentation

3.7.3.1 init_stepper()

```
void SADM_functions::init_stepper ( ) [static]
```

Sets the speed of the stepper motor to a fixed speed (700 in this case). The number used to set the speed is usually defined as RPM, but in this case the motor is set up in way so that it's no longer RPM.

3.7.3.2 set_auto_rotate()

```
void SADM_functions::set_auto_rotate (
    Json_container< JsonObject > * instruction ) [static]
```

Set's the auto_rotate_en member either true or false depending on the instruction parameter.

3.7.3.3 auto_rotate()

```
void SADM_functions::auto_rotate ( ) [static]
```

Rotates the SADM one step each time it's called. Automatically rotates the SADM when continuously called.

3.7.3.4 rotate() [1/3]

```
void SADM_functions::rotate (
    int steps ) [static]
```

Rotates the SADM the passed number of steps.

3.7.3.5 rotate() [2/3]

```
void SADM_functions::rotate (
    float degrees ) [static]
```

Converts from degrees to steps, and rotates the SADM.

3.8 Sensor Class Reference

13

3.7.3.6 rotate() [3/3]

```
void SADM_functions::rotate (
    Json_container< JsonObject > * instruction ) [static]
```

Calls the rotate function matching the instruction parameter type.

3.7.3.7 get_auto_rotate_en()

```
bool SADM_functions::get_auto_rotate_en ( ) [static]
```

Returns true if auto rotate is enabled, false if not.

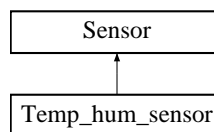
The documentation for this class was generated from the following files:

- containers/SADM_functions.h
- containers/SADM_functions.cpp

3.8 Sensor Class Reference

```
#include <Sensor.h>
```

Inheritance diagram for Sensor:

**Public Member Functions**

- [Sensor](#) (const String &name, const int &pin)
- virtual void **read_sensor** ()=0
- const String & [get_name](#) () const

Protected Attributes

- String **name**
- int **pin**

3.8.1 Detailed Description

The [Sensor](#) class is the parent class of every specific sensor added to the system. The purpose of this class is to ensure that every sensor are of the same type, to be able to store all of them in a collection. This is an abstract class, and it foreces subclasses to override the `read_sensor` function as this is different for every sensor, but is yet required. `get_name` and `get_data_count` are inherited for every child as they do exactly the same for every type of sensor.

Generated by Doxygen

3.8.2 Constructor & Destructor Documentation

3.8.2.1 Sensor()

```
Sensor::Sensor (
    const String & name,
    const int & pin ) [inline]
```

Constructor setting name, pin and data_count as well as initializing the Result pointer as an array of size provided by the data_count parameter.

3.8.3 Member Function Documentation

3.8.3.1 get_name()

```
const String & Sensor::get_name ( ) const [inline]
```

Returns a constant string reference to the name member.

The documentation for this class was generated from the following file:

- sensors/Sensor.h

3.9 Sensor_container Class Reference

```
#include <Sensor_container.h>
```

Public Member Functions

- [Sensor_container \(\)](#)
- [~Sensor_container \(\)](#)
- void [read_sensor](#) (const String &name)
- void [read_all_sensors](#) ()
- void [append_data](#) (Json_container< JsonObject > *dest, sstl::Subscribable *src)
- [Json_container< JsonObject > * get_data](#) (const String &name)
- [Json_container< JsonObject > * get_all_data](#) ()
- [Json_container< JsonObject > * get_sub_data](#) ()
- void [append_available_data](#) (Json_container< JsonObject > *dest)
- [LinkedList< String, Sensor * > & get_available_sensors](#) ()

Private Attributes

- [LinkedList< String, Sensor * > sensor_collection](#)
- [Json_handler json_handler](#)

3.9 Sensor_container Class Reference

15

3.9.1 Detailed Description

The [Sensor_container](#) class holds every available sensor in a linked list, and has methods for retrieving a list of available sensors, as well as reading the different sensors.

3.9.2 Constructor & Destructor Documentation

3.9.2.1 Sensor_container()

```
Sensor_container::Sensor_container ( )
```

Instantiate sensors with name and the pin it's connected to.

3.9.2.2 ~Sensor_container()

```
Sensor_container::~~Sensor_container ( )
```

Delete every sensor in the sensor collection.

3.9.3 Member Function Documentation

3.9.3.1 read_sensor()

```
void Sensor_container::read_sensor (
    const String & name )
```

Read sensor corresponding to the name sent as argument, and return result as [Json_container<JsonObject>](#) pointer.

3.9.3.2 read_all_sensors()

```
void Sensor_container::read_all_sensors ( )
```

Reads all sensors in the sensor collection, and return result as a [Json_container<JsonObject>](#) pointer.

3.9.3.3 append_data()

```
void Sensor_container::append_data (
    Json_container< JsonObject > * dest,
    sstl::Subscribable * src )
```

Appends the name and value of the given Subscribable object to the destination [Json_container<JsonObject>](#).

Generated by Doxygen

3.9.3.4 get_data()

```
Json_container< JsonObject > * Sensor_container::get_data (
    const String & name )
```

Returns a [Json_container<JsonObject>](#) pointer to a specific data object.

3.9.3.5 get_all_data()

```
Json_container< JsonObject > * Sensor_container::get_all_data ( )
```

Returns a [Json_container<JsonObject>](#) pointer containing all available data.

3.9.3.6 get_sub_data()

```
Json_container< JsonObject > * Sensor_container::get_sub_data ( )
```

Returns a [Json_container<JsonObject>](#) pointer containing the subscribed data.

3.9.3.7 append_available_data()

```
void Sensor_container::append_available_data (
    Json_container< JsonObject > * dest )
```

Appends the identifiers of all available data objects to the given [Json_container<JsonObject>](#).

3.9.3.8 get_available_sensors()

```
LinkedList< String, Sensor * > & Sensor_container::get_available_sensors ( )
```

Returns a reference to the linked list containing every available sensor.

The documentation for this class was generated from the following files:

- containers/Sensor_container.h
- containers/Sensor_container.cpp

3.10 Serial_handler Class Reference

```
#include <Serial_handler.h>
```

Public Member Functions

- [Serial_handler](#) ()
- void [serial_init](#) ()
- void [serial_listener](#) ()
- void [send_nack](#) ()
- bool [handshake_approved](#) ()
- bool [connection_request_approved](#) ()
- bool [connection_init_approved](#) ()
- bool [available_data_request_approved](#) ()
- void [send_available_data](#) ([Sensor_container](#) &sc)
- void [print_to_serial](#) ([Json_container<JsonObject>](#) *json)

3.10 Serial_handler Class Reference

17

Private Member Functions

- bool `config_approved` (const unsigned long &baud_rate, const String &config)
- void `send_handshake_response` ()
- void `send_ack` ()

Private Attributes

- unsigned long `baud_rate`
- String `config`
- String `newline_format`
- `Json_handler` `json_handler`
- `Instruction_handler` `instruction_handler`

3.10.1 Detailed Description

The `Serial_handler` class is responsible for establishing connection with a client through the serial port, as well as listening for input and printing output through the serial port when a connection has been established. It has methods complementing the handshake protocol defined in the SatStat communication protocol document, and for listening and printing to serial.

3.10.2 Constructor & Destructor Documentation

3.10.2.1 Serial_handler()

```
Serial_handler::Serial_handler ( )
```

Sets default baud rate, configuration and newline format.

3.10.3 Member Function Documentation

3.10.3.1 serial_init()

```
void Serial_handler::serial_init ( )
```

Initializes serial with configuration stored in the member variables `baud_rate` and `config`.

3.10.3.2 serial_listener()

```
void Serial_handler::serial_listener ( )
```

Listens for serial input. Has to be continuously called to store the input as soon as it's available.

Generated by Doxygen

3.10.3.3 send_nack()

```
void Serial_handler::send_nack ( )
```

Prints JSON formatted negative acknowledgement to serial. Output looks as follows: {"serial_handshake":"failed"}.

3.10.3.4 handshake_approved()

```
bool Serial_handler::handshake_approved ( )
```

Reads serial until handshake received, and sends handshake response according to SatStat communication protocol when it is. Sends NACK and returns false if what's received is not a proper handshake. If no input is received before timeout occurs, it returns false without sending NACK.

3.10.3.5 connection_request_approved()

```
bool Serial_handler::connection_request_approved ( )
```

Reads serial until connection request received, and initializes serial with the received configuration when it is. Sends NACK and returns false if what's received is not a proper request. If no input is received before timeout occurs, it returns false without sending NACK.

3.10.3.6 connection_init_approved()

```
bool Serial_handler::connection_init_approved ( )
```

Reads serial until connection acknowledgement received. Sends NACK and returns false if what's received is not a proper acknowledgement. If no input is received before timeout occurs, it returns false without sending NACK.

3.10.3.7 available_data_request_approved()

```
bool Serial_handler::available_data_request_approved ( )
```

Reads serial until request for available data is received, and responds by sending a list of available sensors when it is. Sends NACK and returns false if what's received is not a proper request. If no input is received before timeout occurs, it returns false without sending NACK.

3.10.3.8 send_available_data()

```
void Serial_handler::send_available_data (
    Sensor_container & sc )
```

Creates a `Json_container<JsonObject>` pointer, appends the name and data type of the sensors in the sensor collection and prints it to the serial.

3.10.3.9 print_to_serial()

```
void Serial_handler::print_to_serial (
    Json_container< JsonObject > * json )
```

Prints the `Json_container<JsonObject>` pointer passed as argument to the serial.

3.11 Subscriber_functions Class Reference**19****3.10.3.10 config_approved()**

```
bool Serial_handler::config_approved (
    const unsigned long & baud_rate,
    const String & config ) [private]
```

Checks if the configuration passed as arguments are valid. Returns true if they are, false if one or more of them are not.

3.10.3.11 send_handshake_response()

```
void Serial_handler::send_handshake_response ( ) [private]
```

Creates a [Json_container<JsonObject>](#) pointer, appends the available baud rates, configurations and newline formats and print it to the serial.

3.10.3.12 send_ack()

```
void Serial_handler::send_ack ( ) [private]
```

Creates a [Json_container<JsonObject>](#) pointer, appends the acknowledgement message and prints it to the serial.

The documentation for this class was generated from the following files:

- handlers/Serial_handler.h
- handlers/Serial_handler.cpp

3.11 Subscriber_functions Class Reference

```
#include <Subscriber_functions.h>
```

Static Public Member Functions

- static void [subscribe](#) ([Json_container](#)< [JsonObject](#) > *request)
- static void [unsubscribe](#) ([Json_container](#)< [JsonObject](#) > *request)

3.11.1 Detailed Description

Abstract class with all static members for the methods to be compatible with list insertion. This class is responsible for handling subscriptions listed in the SatStat communication protocol.

3.11.2 Member Function Documentation

Generated by Doxygen

3.11.2.1 subscribe()

```
void Subscriber_functions::subscribe (
    Json_container< JsonObject > * request ) [static]
```

Copies an entry from the data list to the subscription list.

3.11.2.2 unsubscribe()

```
void Subscriber_functions::unsubscribe (
    Json_container< JsonObject > * request ) [static]
```

Deletes an entry from the subscription list. The entry remains in the data list.

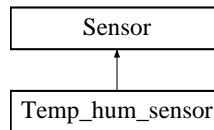
The documentation for this class was generated from the following files:

- containers/Subscriber_functions.h
- containers/Subscriber_functions.cpp

3.12 Temp_hum_sensor Class Reference

```
#include <Temp_hum_sensor.h>
```

Inheritance diagram for Temp_hum_sensor:



Public Member Functions

- [Temp_hum_sensor](#) (const String &name, const int &pin)
- void [read_sensor](#) () override

Private Attributes

- dht **DHT**

Additional Inherited Members

3.12.1 Detailed Description

The [Temp_hum_sensor](#) is an example of a class for a specific sensor, in this case the DHT11 temperature and humidity sensor. This class inherits the members of the sensor class, and includes the DHT library. It has it's own constructor, and overrides the `read_sensor` method as required. This is to support the specific sensor, as well as keeping the same structure as every other sensor to make sure creation and reading of different kinds of sensors work the exact same way.

3.12 Temp_hum_sensor Class Reference

21

3.12.2 Constructor & Destructor Documentation

3.12.2.1 Temp_hum_sensor()

```
Temp_hum_sensor::Temp_hum_sensor (
    const String & name,
    const int & pin )
```

Pass parameter inputs to parent constructor arguments.

3.12.3 Member Function Documentation

3.12.3.1 read_sensor()

```
void Temp_hum_sensor::read_sensor ( ) [override], [virtual]
```

Read all data the sensor can provide, and return as a Result pointer.

Implements [Sensor](#).

The documentation for this class was generated from the following files:

- sensors/Temp_hum_sensor.h
- sensors/Temp_hum_sensor.cpp

Index

- ~Instruction_handler
 - Instruction_handler, 4
- ~Json_container
 - Json_container< T >, 8
- ~SADM_functions
 - SADM_functions, 12
- ~Sensor_container
 - Sensor_container, 15
- append_available_data
 - Sensor_container, 16
- append_data
 - Sensor_container, 15
- append_to
 - Json_handler, 9
- auto_rotate
 - SADM_functions, 12
- available_data_request_approved
 - Serial_handler, 18
- config_approved
 - Serial_handler, 18
- connection
 - HWLayer, 3
- connection_init
 - HWLayer, 3
- connection_init_approved
 - Serial_handler, 18
- connection_request_approved
 - Serial_handler, 18
- create
 - Json_array_container, 6
 - Json_object_container, 10
- create_array
 - Json_handler, 9
- create_object
 - Json_handler, 9
- fetch_instruction
 - Instruction_handler, 5
- get
 - Json_container< T >, 8
- get_all_data
 - Sensor_container, 16
- get_auto_rotate_en
 - SADM_functions, 13
- get_available_sensors
 - Sensor_container, 16
- get_data
 - Sensor_container, 15
- get_name
 - Sensor, 14
- get_sub_data
 - Sensor_container, 16
- handshake
 - HWLayer, 3
- handshake_approved
 - Serial_handler, 18
- HWLayer, 2
 - connection, 3
 - connection_init, 3
 - handshake, 3
 - loop, 3
 - provide_sensor_data, 3
 - setup, 3
- init_stepper
 - SADM_functions, 12
- insert_instruction
 - Instruction_handler, 5
- Instruction_handler, 4
 - ~Instruction_handler, 4
 - fetch_instruction, 5
 - insert_instruction, 5
 - Instruction_handler, 4
 - interpret_instruction, 5
 - queue_is_empty, 5
 - sadm_auto_rotate, 5
 - sadm_auto_rotate_en, 5
- interpret_instruction
 - Instruction_handler, 5
- Json_array_container, 6
 - create, 6
 - Json_array_container, 6
 - parse, 6
- Json_container
 - Json_container< T >, 7
- Json_container< T >, 7
 - ~Json_container, 8
 - get, 8
 - Json_container, 7
- Json_handler, 8
 - append_to, 9
 - create_array, 9
 - create_object, 9
- Json_object_container, 10
 - create, 10
 - Json_object_container, 10
 - parse, 10
- loop
 - HWLayer, 3
- parse
 - Json_array_container, 6
 - Json_object_container, 10
- print_to_serial
 - Serial_handler, 18
- provide_sensor_data
 - HWLayer, 3

queue_is_empty
 Instruction_handler, 5

read_all_sensors
 Sensor_container, 15

read_sensor
 Sensor_container, 15
 Temp_hum_sensor, 21

rotate
 SADM_functions, 12

sadm_auto_rotate
 Instruction_handler, 5

sadm_auto_rotate_en
 Instruction_handler, 5

SADM_functions, 11
 ~SADM_functions, 12
 auto_rotate, 12
 get_auto_rotate_en, 13
 init_stepper, 12
 rotate, 12
 set_auto_rotate, 12

send_ack
 Serial_handler, 19

send_available_data
 Serial_handler, 18

send_handshake_response
 Serial_handler, 19

send_nack
 Serial_handler, 17

Sensor, 13
 get_name, 14
 Sensor, 14

Sensor_container, 14
 ~Sensor_container, 15
 append_available_data, 16
 append_data, 15
 get_all_data, 16
 get_available_sensors, 16
 get_data, 15
 get_sub_data, 16
 read_all_sensors, 15
 read_sensor, 15
 Sensor_container, 15

Serial_handler, 16
 available_data_request_approved, 18
 config_approved, 18
 connection_init_approved, 18
 connection_request_approved, 18
 handshake_approved, 18
 print_to_serial, 18
 send_ack, 19
 send_available_data, 18
 send_handshake_response, 19
 send_nack, 17
 Serial_handler, 17
 serial_init, 17
 serial_listener, 17

serial_init
 Serial_handler, 17
 serial_listener, 17

Temp_hum_sensor, 20
 read_sensor, 21
 Temp_hum_sensor, 21

unsubscribe
 Subscriber_functions, 20



KONGSBERG

**Bachelor of Engineering****Project appendix****Winter semester 2019**

SatStat Hardware Layer Beta 2 - API documentation

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document describes the code for version Beta 2 in great detail. It's recommended that it's used as support material for the SatStat Hardware Layer Beta 2 - Class Diagram, as it's large and complex on it's own.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------



KONGSBERG

SatStat Hardware Layer
Beta 2

Generated by Doxygen 1.8.15

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	2
2.1 Class List	2
3 Class Documentation	2
3.1 Func_ptr< TRet, TParams > Struct Template Reference	2
3.1.1 Detailed Description	3
3.1.2 Constructor & Destructor Documentation	3
3.1.3 Member Function Documentation	4
3.2 Function_control Class Reference	4
3.2.1 Detailed Description	5
3.2.2 Constructor & Destructor Documentation	5
3.2.3 Member Function Documentation	5
3.3 HWLayer Class Reference	6
3.3.1 Detailed Description	6
3.3.2 Constructor & Destructor Documentation	7
3.3.3 Member Function Documentation	7
3.4 Instruction Struct Reference	7
3.4.1 Detailed Description	8
3.4.2 Constructor & Destructor Documentation	8
3.4.3 Member Function Documentation	8
3.5 Instruction_container Class Reference	9
3.5.1 Detailed Description	9
3.5.2 Constructor & Destructor Documentation	9
3.5.3 Member Function Documentation	9
3.6 Json_container< T > Class Template Reference	10
3.6.1 Detailed Description	11
3.6.2 Constructor & Destructor Documentation	11
3.6.3 Member Function Documentation	12
3.7 Message_handler Class Reference	12
3.7.1 Detailed Description	13
3.7.2 Member Function Documentation	13
3.8 Request_container Class Reference	14
3.8.1 Detailed Description	15
3.8.2 Constructor & Destructor Documentation	15
3.8.3 Member Function Documentation	15
3.9 Request_functions Class Reference	16
3.9.1 Detailed Description	16
3.9.2 Constructor & Destructor Documentation	16
3.9.3 Member Function Documentation	17
3.10 SADM_functions Class Reference	17

1 Hierarchical Index	1
3.10.1 Detailed Description	18
3.10.2 Constructor & Destructor Documentation	18
3.10.3 Member Function Documentation	18
3.11 Sensor Class Reference	20
3.11.1 Detailed Description	20
3.11.2 Constructor & Destructor Documentation	21
3.11.3 Member Function Documentation	21
3.12 Sensor_container Class Reference	21
3.12.1 Detailed Description	22
3.12.2 Constructor & Destructor Documentation	22
3.12.3 Member Function Documentation	22
3.13 Serial_handler Class Reference	24
3.13.1 Detailed Description	24
3.13.2 Constructor & Destructor Documentation	24
3.13.3 Member Function Documentation	25
3.14 Temp_hum_sensor Class Reference	27
3.14.1 Detailed Description	27
3.14.2 Constructor & Destructor Documentation	27
3.14.3 Member Function Documentation	28
3.15 Uf_param Struct Reference	28
3.15.1 Constructor & Destructor Documentation	28
Index	29

1 Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Func_ptr< TRet, TParams >	2
Func_ptr< void >	2
Func_ptr< void, Json_container< JsonObject > & >	2
Function_control	4
SADM_functions	17
HWLayer	6
Instruction	7
Instruction_container	9
Json_container< T >	10

Generated by Doxygen

Json_container < JsonObject >	10
Message_handler	12
Request_container	14
Request_functions	16
Sensor	20
Temp_hum_sensor	27
Sensor_container	21
Serial_handler	24
Uf_param	28

2 Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Func_ptr < TRet , TParams >	2
Function_control	4
HWLayer	6
Instruction	7
Instruction_container	9
Json_container < T >	10
Message_handler	12
Request_container	14
Request_functions	16
SADM_functions	17
Sensor	20
Sensor_container	21
Serial_handler	24
Temp_hum_sensor	27
Uf_param	28

3 Class Documentation

3.1 **Func_ptr**< **TRet**, **TParams** > Struct Template Reference

```
#include <Func_ptr.h>
```

3.1 Func_ptr< TRet, TParams > Struct Template Reference**3****Public Types**

- typedef TRet(* **t_func**) (TParams...)

Public Member Functions

- **Func_ptr** ()
- **Func_ptr** (t_func func)
- **Func_ptr** (const **Func_ptr** &other)
- **Func_ptr** & **operator=** (t_func func)
- **Func_ptr** & **operator=** (const **Func_ptr** &other)
- TRet **operator()** (TParams... args)
- bool **is_set** ()

Private Attributes

- t_func **m_func**
- bool **m_set**

3.1.1 Detailed Description

```
template<typename TRet, typename... TParams>
struct Func_ptr< TRet, TParams >
```

Generic function pointer implementation.

3.1.2 Constructor & Destructor Documentation**3.1.2.1 Func_ptr()** [1/3]

```
template<typename TRet, typename... TParams>
Func_ptr< TRet, TParams >::Func_ptr ( ) [inline]
```

Default constructor. Setting m_func to nullptr and m_set to false.

3.1.2.2 Func_ptr() [2/3]

```
template<typename TRet, typename... TParams>
Func_ptr< TRet, TParams >::Func_ptr (
    t_func func ) [inline]
```

Constructor. Takes a raw function pointer, sets m_function to point to that function and sets m_set to true.

3.1.2.3 Func_ptr() [3/3]

```
template<typename TRet, typename... TParams>
Func_ptr< TRet, TParams >::Func_ptr (
    const Func_ptr< TRet, TParams > & other ) [inline]
```

Copy constructor. Utilizing the assignment operator to perform a deep copy from other to this.

Generated by Doxygen

3.1.3 Member Function Documentation

3.1.3.1 operator=() [1/2]

```
template<typename TRet, typename... TParams>
Func_ptr& Func_ptr< TRet, TParams >::operator= (
    t_func func ) [inline]
```

Assignment operator overload. Takes a raw function pointer, sets `m_function` to point to that function and sets `m_set` to true. Also returns a reference to this to allow chaining.

3.1.3.2 operator=() [2/2]

```
template<typename TRet, typename... TParams>
Func_ptr& Func_ptr< TRet, TParams >::operator= (
    const Func_ptr< TRet, TParams > & other ) [inline]
```

Assignment operator overload. Takes a `Func_ptr` and performs a deep copy from that object to this. Also returns a reference to this to allow chaining.

3.1.3.3 operator>()

```
template<typename TRet, typename... TParams>
TRet Func_ptr< TRet, TParams >::operator() (
    TParams... args ) [inline]
```

Function call operator overload. Allow direct call to the function.

3.1.3.4 is_set()

```
template<typename TRet, typename... TParams>
bool Func_ptr< TRet, TParams >::is_set ( ) [inline]
```

Checks if the function pointer actually points to a function or not.

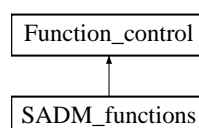
The documentation for this struct was generated from the following file:

- containers/Func_ptr.h

3.2 Function_control Class Reference

```
#include <Function_control.h>
```

Inheritance diagram for `Function_control`:



3.2 Function_control Class Reference**5****Public Member Functions**

- virtual `~Function_control ()=0`

Static Public Member Functions

- static bool `is_available ()`
- static void `run ()`

Static Protected Member Functions

- static void `reserve (const String &name, const Func_ptr< void > &func)`
- static void `release ()`

Static Private Attributes

- static String `m_name`
- static bool `m_available = true`
- static `Func_ptr< void > m_func`

3.2.1 Detailed Description

`Function_control` is a wrapper around a function. A function can reserve `Function_control`, by passing itself in. When reserved, no other function can enter until it's released. This class is created to make prototyping of functions easier.

3.2.2 Constructor & Destructor Documentation**3.2.2.1 `~Function_control()`**

```
virtual Function_control::~~Function_control ( ) [pure virtual]
```

Pure virtual destructor to make the class abstract.

3.2.3 Member Function Documentation**3.2.3.1 `is_available()`**

```
bool Function_control::is_available ( ) [static]
```

Checks if `Function_control` is currently available.

Generated by Doxygen

3.2.3.2 run()

```
void Function_control::run ( ) [static]
```

Executes the function currently reserving [Function_control](#).

3.2.3.3 reserve()

```
void Function_control::reserve (
    const String & name,
    const Func_ptr< void > & func ) [static], [protected]
```

Reserve [Function_control](#) with the given function. [Function_control](#) will become unavailable.

3.2.3.4 release()

```
void Function_control::release ( ) [static], [protected]
```

Releases the function currently reserving [Function_control](#). [Function_control](#) will become available.

The documentation for this class was generated from the following files:

- handlers/Function_control.h
- handlers/Function_control.cpp

3.3 HWLayer Class Reference

```
#include <HWLayer.h>
```

Public Member Functions

- [HWLayer](#) ()
- [~HWLayer](#) ()
- void [setup](#) ()
- void [loop](#) ()

Private Attributes

- [Sensor_container](#) **sensor_container**
- [Message_handler](#) **message_handler**
- [Serial_handler](#) * **serial_handler**
- unsigned long **sensor_interval_start_time**
- const unsigned long **sensor_interval_duration** = 2000

3.3.1 Detailed Description

This class acts as the "main" class. Doxygen, the documentation generation tool, doesn't support .ino files, therefore this class has been created to be able to generate documentation for the "main" class. The project still has a .ino file for it to be considered an Arduino project, but the setup and loop functions in that file simply calls the corresponding methods in this one. The setup method of this class handles the handshake protocol, where as the loop method handles the receiving of messages from serial, transmission of sensor data through serial, and execution of messages.

3.4 Instruction Struct Reference**7****3.3.2 Constructor & Destructor Documentation****3.3.2.1 HWLayer()**

```
HWLayer::HWLayer ( )
```

Constructor instantiating the serial_handler member.

3.3.2.2 ~HWLayer()

```
HWLayer::~~HWLayer ( )
```

Destructor deleting the serial_handler member.

3.3.3 Member Function Documentation**3.3.3.1 setup()**

```
void HWLayer::setup ( )
```

Sets up Vcc and GND pins for DHT22 temperature and humidity sensor, and initializes the serial port. Also handles the handshake protocol as defined in SatStat communication protocol.

3.3.3.2 loop()

```
void HWLayer::loop ( )
```

Continuously listens for serial inputs, prints read sensor data to serial at a given interval and execute received messages.

The documentation for this class was generated from the following files:

- HWLayer.h
- HWLayer.cpp

3.4 Instruction Struct Reference

```
#include <Instruction_container.h>
```

Public Member Functions

- [Instruction](#) (const String &identifier, const [Json_container](#)< JsonObject > &signature, const [Func_ptr](#)< void, [Json_container](#)< JsonObject > & > &func)
- void [run](#) ([Json_container](#)< JsonObject > &doc)
- String & [get_identifier](#) ()
- [Json_container](#)< JsonObject > & [get_signature](#) ()

Generated by Doxygen

Private Attributes

- String **m_identifier**
- [Json_container](#)< JsonObject > **m_signature**
- [Func_ptr](#)< void, [Json_container](#)< JsonObject > & > **m_func**

3.4.1 Detailed Description

Struct holding a function, it's name and it's signature.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 Instruction()

```
Instruction::Instruction (
    const String & identifier,
    const Json\_container< JsonObject > & signature,
    const Func\_ptr< void, Json\_container< JsonObject > & > & func ) [inline]
```

Constructor setting the identifier, signature and function pointer.

3.4.3 Member Function Documentation

3.4.3.1 run()

```
void Instruction::run (
    Json\_container< JsonObject > & doc ) [inline]
```

Calls the the function the function pointer holds.

3.4.3.2 get_identifier()

```
String& Instruction::get_identifier ( ) [inline]
```

Returns the identifier of the instruction as String.

3.4.3.3 get_signature()

```
Json\_container<JsonObject>& Instruction::get_signature ( ) [inline]
```

Returns the signature of the instruction as [Json_container](#)<JsonObject>.

The documentation for this struct was generated from the following file:

- containers/Instruction_container.h

3.5 Instruction_container Class Reference**9****3.5 Instruction_container Class Reference**

```
#include <Instruction_container.h>
```

Public Member Functions

- [Instruction_container](#) ()
- [Instruction](#) * [get](#) (const String &ins)
- [LinkedList](#)< String, [Instruction](#) * > & [get_available_instructions](#) ()
- [template](#)<typename TFirst, typename ... TParams>
[Json_container](#)< [JsonObject](#) > **[parse_params](#)** (TFirst t, TParams ...args) const

Private Member Functions

- [template](#)<typename TFirst, typename... TParams>
[Json_container](#)< [JsonObject](#) > [parse_params](#) (TFirst t, TParams... args) const
- [template](#)<typename TFirst, typename... TParams>
void [format_params](#) ([Json_container](#)< [JsonObject](#) > &dest, TFirst t, TParams... args) const
- [template](#)<typename TFirst >
void [format_params](#) ([Json_container](#)< [JsonObject](#) > &dest, TFirst t) const

Private Attributes

- [LinkedList](#)< String, [Instruction](#) * > **[m_available_instructions](#)**

3.5.1 Detailed Description

Class containing a list of available instructions.

3.5.2 Constructor & Destructor Documentation**3.5.2.1 Instruction_container()**

```
Instruction_container::Instruction_container ( )
```

Constructor appending every available instruction to the list.

3.5.3 Member Function Documentation**3.5.3.1 get()**

```
Instruction * Instruction\_container::get (
    const String & ins )
```

Returns a specific instruction in the list.

Generated by Doxygen

3.5.3.2 get_available_instructions()

```
LinkedList< String, Instruction * > & Instruction_container::get_available_instructions ( )
```

Returns the entire list of available instructions.

3.5.3.3 parse_params()

```
template<typename TFirst , typename... TParams>
Json_container<JsonObject> Instruction_container::parse_params (
    TFirst t,
    TParams... args ) const [private]
```

Returns JSON formatted parameters of a function. Called in the constructor when creating the `Instruction` object which will be added to the list. Arguments has to be of type `Uf_param`.

3.5.3.4 format_params() [1/2]

```
template<typename TFirst , typename... TParams>
void Instruction_container::format_params (
    Json_container< JsonObject > & dest,
    TFirst t,
    TParams... args ) const [inline], [private]
```

Called by `parse_params` or itself when the parameter pack is not empty.

3.5.3.5 format_params() [2/2]

```
template<typename TFirst >
void Instruction_container::format_params (
    Json_container< JsonObject > & dest,
    TFirst t ) const [inline], [private]
```

Base format method. Called by `parse_params` or the other `format_params` method when the parameter pack is empty.

The documentation for this class was generated from the following files:

- containers/Instruction_container.h
- containers/Instruction_container.cpp

3.6 Json_container< T > Class Template Reference

```
#include <Json_container.h>
```

Public Member Functions

- `Json_container` (const `Json_container` &src)
- `Json_container` & `operator=` (const `Json_container` &src)
- `~Json_container` ()
- `T * operator->` ()
- bool `parse` (const String &str)
- `T & get` ()
- `template<>`
`Json_container` ()
- `template<>`
bool `parse` (const String &str)

3.6 Json_container< T > Class Template Reference

11

Private Member Functions

- void **copy** (const [Json_container](#) &src)
- `template<>`
void **copy** (const [Json_container](#) &src)

Private Attributes

- `DynamicJsonBuffer * m_buffer`
- `T * m_json`

3.6.1 Detailed Description

```
template<typename T>
class Json_container< T >
```

The ArduinoJson library requires JsonObject and JsonArrays to be created within a JsonBuffer. When a JsonObject or JsonArray is retrieved from a buffer, it's returned as a reference. This means that neither of those can live outside of the buffer, so this class encapsulates the JsonBuffer together with a JsonObject or JsonArray to simplify object creation and passing. This class is generic for it to be specified upon instantiation if it's to hold a JsonObject or JsonArray.

3.6.2 Constructor & Destructor Documentation**3.6.2.1 Json_container()** [1/2]

```
template<typename T >
Json_container< T >::Json_container (
    const Json\_container< T > & src ) [inline]
```

Copy constructor. Calls the copy method to perform a deep copy.

3.6.2.2 ~Json_container()

```
template<typename T >
Json_container< T >::~~Json_container ( ) [inline]
```

Destructor. Clears and deletes the DynamicJsonBuffer.

3.6.2.3 Json_container() [2/2]

```
template<>
Json_container< JsonArray >::Json_container ( ) [inline]
```

Default constructor. Instantiates the DynamicJsonBuffer and creates an empty array.

Generated by Doxygen

3.6.3 Member Function Documentation

3.6.3.1 operator=()

```
template<typename T >
Json_container< T > & Json_container< T >::operator= (
    const Json_container< T > & src ) [inline]
```

Assignment operator overload. Utilizes the copy method to perform a deep copy.

3.6.3.2 operator->()

```
template<typename T >
T * Json_container< T >::operator-> ( ) [inline]
```

Returns a pointer to the JSON data. Allow the user to access the public members of m_json directly without a getter.

3.6.3.3 get()

```
template<typename T >
T & Json_container< T >::get ( ) [inline]
```

Returns a reference to the JSON data.

3.6.3.4 copy()

```
template<>
void Json_container< JsonArray >::copy (
    const Json_container< T > & src ) [inline], [private]
```

Performs a deep copy from the source object to this object.

3.6.3.5 parse()

```
template<>
bool Json_container< JsonArray >::parse (
    const String & str ) [inline]
```

Parses a string to the JsonArray.

The documentation for this class was generated from the following file:

- containers/Json_container.h

3.7 Message_handler Class Reference

```
#include <Message_handler.h>
```

3.7 Message_handler Class Reference

13

Public Member Functions

- bool [insert_message](#) (const String &input_data)
- [Json_container](#)< [JsonObject](#) > [fetch_message](#) ()
- bool [has_requests](#) () const
- bool [has_instructions](#) () const
- void [interpret_request](#) ()
- void [interpret_instruction](#) ()
- void [append_available_instructions](#) ([Json_container](#)< [JsonObject](#) > &dest)

Private Attributes

- [Request_container](#) **request_container**
- [Instruction_container](#) **instruction_container**
- [QueueArray](#)< [Json_container](#)< [JsonObject](#) > > **message_queue**
- [QueueArray](#)< [Json_container](#)< [JsonObject](#) > > **instruction_queue**

3.7.1 Detailed Description

The [Message_handler](#) class holds the messages received through the serial port. The main responsibility of this class is to insert, fetch and execute instructions and requests.

3.7.2 Member Function Documentation**3.7.2.1 insert_message()**

```
bool Message_handler::insert_message (
    const String & input_data )
```

Inserts passed message into the message or instruction queue. Returns true if success, false if not.

3.7.2.2 fetch_message()

```
Json\_container< JsonObject > Message_handler::fetch_message ( )
```

Fetches a message from the message queue. Returned as [Json_container](#)<[JsonObject](#)>.

3.7.2.3 has_requests()

```
bool Message_handler::has_requests ( ) const
```

Returns true if message queue is empty, false if not.

3.7.2.4 has_instructions()

```
bool Message_handler::has_instructions ( ) const
```

Returns true if instruction queue is empty, false if not.

Generated by Doxygen

3.7.2.5 interpret_request()

```
void Message_handler::interpret_request ( )
```

Fetches the first request in the `message_queue`, gets the corresponding function from the `request_container` and calls the retrieved function.

3.7.2.6 interpret_instruction()

```
void Message_handler::interpret_instruction ( )
```

Fetches the first instruction in the `instruction_queue`, gets the corresponding function from the `instruction_container` and calls the retrieved function.

3.7.2.7 append_available_instructions()

```
void Message_handler::append_available_instructions (
    Json_container< JsonObject > & dest )
```

Appends all available instructions to the destination `Json_container<JsonObject>`.

The documentation for this class was generated from the following files:

- `handlers/Message_handler.h`
- `handlers/Message_handler.cpp`

3.8 Request_container Class Reference

```
#include <Request_container.h>
```

Public Member Functions

- `Request_container` ()
- `bool exists` (const String &key)
- `bool has_params` (const String &key)
- `Func_ptr< void > & get_no_param` (const String &key)
- `Func_ptr< void, Json_container< JsonObject > & > & get_with_param` (const String &key)

Private Member Functions

- `void append` (const String &key, const `Func_ptr< void > &func`)
- `void append` (const String &key, const `Func_ptr< void, Json_container< JsonObject > & > &func`)

Private Attributes

- `LinkedList< String, Func_ptr< void > >` **no_param**
- `LinkedList< String, Func_ptr< void, Json_container< JsonObject > & >` **with_param**

3.8 Request_container Class Reference

15

3.8.1 Detailed Description

Class containing two lists of available requests. The reason for having two lists is that some requests takes parameters, and some don't. By having two lists, it's possible to append both kinds of requests without knowing if it takes parameters or not. It also makes it possible to retrieve them from the correct list by checking if it exists at all, and whether or not it is in the list with requests that has parameters.

3.8.2 Constructor & Destructor Documentation**3.8.2.1 Request_container()**

```
Request_container::Request_container ( )
```

Appends available requests to the correct list.

3.8.3 Member Function Documentation**3.8.3.1 exists()**

```
bool Request_container::exists (
    const String & key )
```

Checks if the given key is present in either of the lists.

3.8.3.2 has_params()

```
bool Request_container::has_params (
    const String & key )
```

Checks if the given key is present in the with_params list. In other words, if it has parameters.

3.8.3.3 get_no_param()

```
Func_ptr< void > & Request_container::get_no_param (
    const String & key )
```

Returns the given object from the no_param list.

3.8.3.4 get_with_param()

```
Func_ptr< void, Json_container< JsonObject > & > & Request_container::get_with_param (
    const String & key )
```

Returns the given object from the with_param list.

Generated by Doxygen

3.8.3.5 `append()` [1/2]

```
void Request_container::append (
    const String & key,
    const Func_ptr< void > & func ) [private]
```

Appends the given `Func_ptr` to the `no_param` list.

3.8.3.6 `append()` [2/2]

```
void Request_container::append (
    const String & key,
    const Func_ptr< void, Json_container< JsonObject > & > & func ) [private]
```

Appends the given `Func_ptr` to the `with_param` list.

The documentation for this class was generated from the following files:

- `containers/Request_container.h`
- `containers/Request_container.cpp`

3.9 Request_functions Class Reference

```
#include <Request_functions.h>
```

Public Member Functions

- virtual `~Request_functions()`=0

Static Public Member Functions

- static void `subscribe` (`Json_container< JsonObject > &request`)
- static void `unsubscribe` (`Json_container< JsonObject > &request`)
- static void `reset` ()

3.9.1 Detailed Description

Abstract class with all static members for the methods to be compatible with list insertion. This class is responsible for handling all the requests listed in the SatStat communication protocol.

3.9.2 Constructor & Destructor Documentation**3.9.2.1** `~Request_functions()`

```
virtual Request_functions::~Request_functions ( ) [pure virtual]
```

Pure virtual destructor to make class abstract.

3.10 SADM_functions Class Reference

17

3.9.3 Member Function Documentation**3.9.3.1 subscribe()**

```
void Request_functions::subscribe (
    Json_container< JsonObject > & request ) [static]
```

Handles subscription to various data.

3.9.3.2 unsubscribe()

```
void Request_functions::unsubscribe (
    Json_container< JsonObject > & request ) [static]
```

Handles unsubscription of data.

3.9.3.3 reset()

```
void Request_functions::reset ( ) [static]
```

Resets the Arduino.

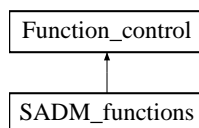
The documentation for this class was generated from the following files:

- containers/Request_functions.h
- containers/Request_functions.cpp

3.10 SADM_functions Class Reference

```
#include <SADM_functions.h>
```

Inheritance diagram for SADM_functions:

**Public Member Functions**

- virtual `~SADM_functions ()=0`

Generated by Doxygen

Static Public Member Functions

- static void `set_step_size` (Json_container< JsonObject > &ins)
- static void `set_stepping_mode` (Json_container< JsonObject > &ins)
- static void `set_ratio` (Json_container< JsonObject > &ins)
- static void `set_speed` (Json_container< JsonObject > &ins)
- static void `set_dir` (Json_container< JsonObject > &ins)
- static void `rotate` (Json_container< JsonObject > &ins)
- static void `rotate` ()
- static void `instant_release` ()

Static Private Member Functions

- static int `deg_to_steps` (const float °)

Static Private Attributes

- static float `m_step_size` = 1.8
- static int `m_stepping_mode` = 8
- static float `m_ratio` = 1
- static unsigned long `m_period`
- static int `m_steps` = 0
- static bool `m_rising` = true
- static unsigned long `m_last_pulse` = micros()
- static const int `step_pin` = 5
- static const int `dir_pin` = 6

Additional Inherited Members**3.10.1 Detailed Description**

Abstract class with all static members for the methods to be compatible with list insertion. This class is responsible for controlling the the SADM, and has member functions for rotating in different ways by providing different arguments.

3.10.2 Constructor & Destructor Documentation**3.10.2.1 ~SADM_functions()**

```
virtual SADM_functions::~SADM_functions ( ) [pure virtual]
```

Pure virtual destructor to make the class abstract.

3.10.3 Member Function Documentation

3.10 SADM_functions Class Reference**19****3.10.3.1 set_step_size()**

```
void SADM_functions::set_step_size (
    Json_container< JsonObject > & ins ) [static]
```

Interprets the `set_step_size` instruction received from SWL. Directly sets `m_step_size`.

3.10.3.2 set_stepping_mode()

```
void SADM_functions::set_stepping_mode (
    Json_container< JsonObject > & ins ) [static]
```

Interprets the `set_stepping_mode` instruction received from SWL. Directly sets `m_stepping_mode`.

3.10.3.3 set_ratio()

```
void SADM_functions::set_ratio (
    Json_container< JsonObject > & ins ) [static]
```

Interprets the `set_ratio` instruction received from SWL. Directly sets `m_ratio`.

3.10.3.4 set_speed()

```
void SADM_functions::set_speed (
    Json_container< JsonObject > & ins ) [static]
```

Interprets the `set_speed` instruction received from SWL. Translates from RPM to period and updates `m_period`.

3.10.3.5 set_dir()

```
void SADM_functions::set_dir (
    Json_container< JsonObject > & ins ) [static]
```

Interprets the `set_direction` instruction received from SWL. Directly sets the voltage level on the direction pin to either high or low depending on the direction parameter.

3.10.3.6 rotate() [1/2]

```
void SADM_functions::rotate (
    Json_container< JsonObject > & ins ) [static]
```

Interprets the `rotate` instruction received from SWL. Sets the number of steps to rotate, and reserves `Function_control` with the other rotate function.

3.10.3.7 rotate() [2/2]

```
void SADM_functions::rotate ( ) [static]
```

Square wave generator responsible for rotating the SADM the given number of steps.

Generated by Doxygen

3.10.3.8 instant_release()

```
void SADM_functions::instant_release ( ) [static]
```

Releases itself from [Function_control](#) the first time it's called. When functions only needs to be called once, this function reserves [Function_control](#).

3.10.3.9 deg_to_steps()

```
int SADM_functions::deg_to_steps (
    const float & deg ) [static], [private]
```

Converts from degrees to number of steps.

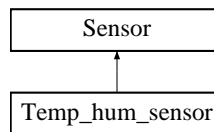
The documentation for this class was generated from the following files:

- containers/SADM_functions.h
- containers/SADM_functions.cpp

3.11 Sensor Class Reference

```
#include <Sensor.h>
```

Inheritance diagram for Sensor:



Public Member Functions

- [Sensor](#) (const String &name, const int &pin)
- virtual [~Sensor](#) ()
- virtual void [read_sensor](#) ()=0
- const String & [get_name](#) () const

Protected Attributes

- String **name**
- int **pin**

3.11.1 Detailed Description

The [Sensor](#) class is the parent class of every specific sensor added to the system. The purpose of this class is to ensure that every sensor are of the same type, to be able to store all of them in a collection. This is an abstract class, and it forces subclasses to override the [read_sensor](#) function as this is different for every sensor, but is yet required. [get_name](#) and [get_data_count](#) are inherited for every child as they do exactly the same for every type of sensor.

3.12 Sensor_container Class Reference

21

3.11.2 Constructor & Destructor Documentation**3.11.2.1 Sensor()**

```
Sensor::Sensor (
    const String & name,
    const int & pin ) [inline]
```

Constructor setting name and pin.

3.11.2.2 ~Sensor()

```
virtual Sensor::~~Sensor ( ) [inline], [virtual]
```

As other classes will inherit this one, we need a virtual destructor to let the compiler know that a polymorphic object instantiated through a [Sensor](#) type might have it's own destructor that has to be called upon deletion.

3.11.3 Member Function Documentation**3.11.3.1 read_sensor()**

```
virtual void Sensor::read_sensor ( ) [pure virtual]
```

Pure virtual read_sensor method to enforce inheriting classes to override this method.

Implemented in [Temp_hum_sensor](#).

3.11.3.2 get_name()

```
const String & Sensor::get_name ( ) const [inline]
```

Returns a constant string reference to the name member.

The documentation for this class was generated from the following file:

- sensors/Sensor.h

3.12 Sensor_container Class Reference

```
#include <Sensor_container.h>
```

Generated by Doxygen

Public Member Functions

- [Sensor_container](#) ()
- [~Sensor_container](#) ()
- void [read_sensor](#) (const String &name)
- void [read_all_sensors](#) ()
- void [append_data](#) ([Json_container](#)< JsonObject > &dest, sstl::Subscribable *src)
- [Json_container](#)< JsonObject > [get_data](#) (const String &name)
- [Json_container](#)< JsonObject > [get_all_data](#) ()
- [Json_container](#)< JsonObject > [get_sub_data](#) ()
- void [append_available_data](#) ([Json_container](#)< JsonObject > &dest)
- [LinkedList](#)< String, [Sensor](#) * > & [get_available_sensors](#) ()

Private Attributes

- [LinkedList](#)< String, [Sensor](#) * > **sensor_collection**

3.12.1 Detailed Description

The [Sensor_container](#) class holds every available sensor in a list, and has methods for retrieving a list of available sensors, as well as reading the different sensors.

3.12.2 Constructor & Destructor Documentation**3.12.2.1 Sensor_container()**

```
Sensor_container::Sensor_container ( )
```

Instantiate sensors with name and the pin it's connected to.

3.12.2.2 ~Sensor_container()

```
Sensor_container::~~Sensor_container ( )
```

Delete every sensor in the sensor collection.

3.12.3 Member Function Documentation**3.12.3.1 read_sensor()**

```
void Sensor_container::read_sensor (
    const String & name )
```

Read sensor corresponding to the name sent as argument.

3.12 Sensor_container Class Reference**23****3.12.3.2 read_all_sensors()**

```
void Sensor_container::read_all_sensors ( )
```

Reads all sensors in the sensor collection.

3.12.3.3 append_data()

```
void Sensor_container::append_data (
    Json_container< JsonObject > & dest,
    sstl::Subscribable * src )
```

Appends data from a Subscribable object to the destination [Json_container<JsonObject>](#).

3.12.3.4 get_data()

```
Json_container< JsonObject > Sensor_container::get_data (
    const String & name )
```

Returns the given sensor data as [Json_container<JsonObject>](#).

3.12.3.5 get_all_data()

```
Json_container< JsonObject > Sensor_container::get_all_data ( )
```

Returns a [Json_container<JsonObject>](#) containing all sensor data.

3.12.3.6 get_sub_data()

```
Json_container< JsonObject > Sensor_container::get_sub_data ( )
```

Returns a [Json_container<JsonObject>](#) containing sensor data that has been subscribed to.

3.12.3.7 append_available_data()

```
void Sensor_container::append_available_data (
    Json_container< JsonObject > & dest )
```

Appends the available data to the destination [Json_container<JsonObject>](#).

3.12.3.8 get_available_sensors()

```
LinkedList< String, Sensor * > & Sensor_container::get_available_sensors ( )
```

Returns a reference to the linked list containing every available sensor.

The documentation for this class was generated from the following files:

- containers/Sensor_container.h
- containers/Sensor_container.cpp

Generated by Doxygen

3.13 Serial_handler Class Reference

```
#include <Serial_handler.h>
```

Public Member Functions

- [Serial_handler](#) ([Sensor_container](#) &sc, [Message_handler](#) &ih)
- void [handshake](#) ()
- void [send_available_data](#) ()
- void [send_available_instructions](#) ()
- void [serial_listener](#) ()
- void [send_ping](#) ()
- void [print_to_serial](#) ([Json_container](#)< [JsonObject](#) > &json)

Private Member Functions

- bool [handshake_approved](#) ()
- bool [connection_request_approved](#) ()
- bool [connection_init_approved](#) ()
- bool [config_approved](#) (const unsigned long &baud_rate, const String &config)
- void [serial_init](#) ()
- void [send_handshake_response](#) ()
- void [send_ack](#) ()
- void [send_error_message](#) (const String &msg)

Private Attributes

- [Sensor_container](#) * **sensor_container**
- [Message_handler](#) * **message_handler**
- unsigned long **m_start_time**
- const unsigned long **m_timeout_duration** = 1000
- int **m_timeout_counter**
- unsigned long **baud_rate**
- String **config**
- String **newline_format**

3.13.1 Detailed Description

The [Serial_handler](#) class is responsible for establishing connection with a client through the serial port, as well as listening for input and printing output through the serial port when a connection has been established. It has methods complementing the handshake protocol defined in the SatStat communication protocol document.

3.13.2 Constructor & Destructor Documentation

3.13 Serial_handler Class Reference

25

3.13.2.1 Serial_handler()

```
Serial_handler::Serial_handler (
    Sensor_container & sc,
    Message_handler & ih )
```

Constructor taking a reference to a [Sensor_container](#) and a [Message_handler](#). Sets all the members.

3.13.3 Member Function Documentation

3.13.3.1 handshake()

```
void Serial_handler::handshake ( )
```

Executes handshake protocol according to SatStat communication protocol.

3.13.3.2 send_available_data()

```
void Serial_handler::send_available_data ( )
```

Creates a [Json_container<JsonObject>](#), appends the available data and prints it to the serial.

3.13.3.3 send_available_instructions()

```
void Serial_handler::send_available_instructions ( )
```

Creates a [Json_container<JsonObject>](#), appends the available instructions and prints it to the serial.

3.13.3.4 serial_listener()

```
void Serial_handler::serial_listener ( )
```

Listens for serial input. Has to be continuously called to store the input as soon as it's available.

3.13.3.5 send_ping()

```
void Serial_handler::send_ping ( )
```

Sends a ping to the client.

3.13.3.6 print_to_serial()

```
void Serial_handler::print_to_serial (
    Json_container< JsonObject > & json )
```

Prints the given [Json_container<JsonObject>](#) to the serial.

Generated by Doxygen

3.13.3.7 handshake_approved()

```
bool Serial_handler::handshake_approved ( ) [private]
```

Reads serial until handshake received. Sends handshake response according to SatStat communication protocol when it is. Sends error message if what's received is not a proper handshake. If no input is received before timeout occurs, a ping is sent to notify the client we're still here.

3.13.3.8 connection_request_approved()

```
bool Serial_handler::connection_request_approved ( ) [private]
```

Reads serial until connection request received, and initializes serial with the received configuration. Sends error message and returns false if what's received is not a proper request. If no input is received before timeout occurs, it returns false without sending error message.

3.13.3.9 connection_init_approved()

```
bool Serial_handler::connection_init_approved ( ) [private]
```

Reads serial until connection acknowledgement received. Sends error message and returns false if what's received is not a proper acknowledgement. If no input is received before timeout occurs, it returns false without sending error message.

3.13.3.10 config_approved()

```
bool Serial_handler::config_approved (
    const unsigned long & baud_rate,
    const String & config ) [private]
```

Checks if the given configuration is valid.

3.13.3.11 serial_init()

```
void Serial_handler::serial_init ( ) [private]
```

Initializes serial with configuration stored in the member variables baud_rate and config.

3.13.3.12 send_handshake_response()

```
void Serial_handler::send_handshake_response ( ) [private]
```

Creates a [Json_container<JsonObject>](#), appends the available baud rates, configurations and newline formats and print it to the serial.

3.13.3.13 send_ack()

```
void Serial_handler::send_ack ( ) [private]
```

Creates a [Json_container<JsonObject>](#), appends the acknowledgement message and prints it to the serial.

3.14 Temp_hum_sensor Class Reference

27

3.13.3.14 send_error_message()

```
void Serial_handler::send_error_message (
    const String & msg ) [private]
```

Converts the given error message into a JSON formatted error message and prints to serial.

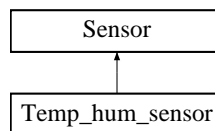
The documentation for this class was generated from the following files:

- handlers/Serial_handler.h
- handlers/Serial_handler.cpp

3.14 Temp_hum_sensor Class Reference

```
#include <Temp_hum_sensor.h>
```

Inheritance diagram for Temp_hum_sensor:



Public Member Functions

- [Temp_hum_sensor](#) (const String &name, const int &pin)
- [~Temp_hum_sensor](#) ()
- void [read_sensor](#) () override

Private Attributes

- DHT_Unified * **dht**

Additional Inherited Members

3.14.1 Detailed Description

The [Temp_hum_sensor](#) is an example of a class for a specific sensor, in this case the DHT22 temperature and humidity sensor. This class inherits from the [Sensor](#) class, and includes the DHT library. It has its own constructor, and overrides the `read_sensor` method as required. This is to support the specific sensor, as well as keeping the same structure as every other sensor to make sure creation and reading of different kinds of sensors work the exact same way.

3.14.2 Constructor & Destructor Documentation

Generated by Doxygen

3.14.2.1 Temp_hum_sensor()

```
Temp_hum_sensor::Temp_hum_sensor (
    const String & name,
    const int & pin )
```

Constructor. Passes the parameter inputs to parent constructor arguments, initializes dht and adds temperature and humidity entries to the SSTL data list.

3.14.2.2 ~Temp_hum_sensor()

```
Temp_hum_sensor::~Temp_hum_sensor ( )
```

Destructor. Deletes dht.

3.14.3 Member Function Documentation

3.14.3.1 read_sensor()

```
void Temp_hum_sensor::read_sensor ( ) [override], [virtual]
```

Read all data the sensor can provide and update the SSTL list with the new readings.

Implements [Sensor](#).

The documentation for this class was generated from the following files:

- sensors/Temp_hum_sensor.h
- sensors/Temp_hum_sensor.cpp

3.15 Uf_param Struct Reference

Public Member Functions

- [Uf_param](#) (const String &identifier, const String &type)

Public Attributes

- String **identifier**
- String **type**

3.15.1 Constructor & Destructor Documentation

3.15.1.1 Uf_param()

```
Uf_param::Uf_param (
    const String & identifier,
    const String & type ) [inline]
```

Constructor setting the member variables.

The documentation for this struct was generated from the following file:

- containers/Instruction_container.h

Index

- ~Function_control
 - Function_control, 5
- ~HWLayer
 - HWLayer, 7
- ~Json_container
 - Json_container< T >, 11
- ~Request_functions
 - Request_functions, 16
- ~SADM_functions
 - SADM_functions, 18
- ~Sensor
 - Sensor, 21
- ~Sensor_container
 - Sensor_container, 22
- ~Temp_hum_sensor
 - Temp_hum_sensor, 28
- append
 - Request_container, 15, 16
- append_available_data
 - Sensor_container, 23
- append_available_instructions
 - Message_handler, 14
- append_data
 - Sensor_container, 23
- config_approved
 - Serial_handler, 26
- connection_init_approved
 - Serial_handler, 26
- connection_request_approved
 - Serial_handler, 26
- copy
 - Json_container< T >, 12
- deg_to_steps
 - SADM_functions, 20
- exists
 - Request_container, 15
- fetch_message
 - Message_handler, 13
- format_params
 - Instruction_container, 10
- Func_ptr
 - Func_ptr< TRet, TParams >, 3
- Func_ptr< TRet, TParams >, 2
 - Func_ptr, 3
 - is_set, 4
 - operator(), 4
 - operator=, 4
- Function_control, 4
 - ~Function_control, 5
 - is_available, 5
 - release, 6
 - reserve, 6
 - run, 5
- get
 - Instruction_container, 9
 - Json_container< T >, 12
- get_all_data
 - Sensor_container, 23
- get_available_instructions
 - Instruction_container, 9
- get_available_sensors
 - Sensor_container, 23
- get_data
 - Sensor_container, 23
- get_identifier
 - Instruction, 8
- get_name
 - Sensor, 21
- get_no_param
 - Request_container, 15
- get_signature
 - Instruction, 8
- get_sub_data
 - Sensor_container, 23
- get_with_param
 - Request_container, 15
- handshake
 - Serial_handler, 25
- handshake_approved
 - Serial_handler, 25
- has_instructions
 - Message_handler, 13
- has_params
 - Request_container, 15
- has_requests
 - Message_handler, 13
- HWLayer, 6
 - ~HWLayer, 7
 - HWLayer, 7
 - loop, 7
 - setup, 7
- insert_message
 - Message_handler, 13
- instant_release
 - SADM_functions, 19
- Instruction, 7
 - get_identifier, 8
 - get_signature, 8
 - Instruction, 8
 - run, 8
- Instruction_container, 9
 - format_params, 10
 - get, 9
 - get_available_instructions, 9

- Instruction_container, 9
- parse_params, 10
- interpret_instruction
 - Message_handler, 14
- interpret_request
 - Message_handler, 13
- is_available
 - Function_control, 5
- is_set
 - Func_ptr< TRet, TParams >, 4
- Json_container
 - Json_container< T >, 11
- Json_container< T >, 10
 - ~Json_container, 11
 - copy, 12
 - get, 12
 - Json_container, 11
 - operator->, 12
 - operator=, 12
 - parse, 12
- loop
 - HWLayer, 7
- Message_handler, 12
 - append_available_instructions, 14
 - fetch_message, 13
 - has_instructions, 13
 - has_requests, 13
 - insert_message, 13
 - interpret_instruction, 14
 - interpret_request, 13
- operator()
 - Func_ptr< TRet, TParams >, 4
- operator->
 - Json_container< T >, 12
- operator=
 - Func_ptr< TRet, TParams >, 4
 - Json_container< T >, 12
- parse
 - Json_container< T >, 12
- parse_params
 - Instruction_container, 10
- print_to_serial
 - Serial_handler, 25
- read_all_sensors
 - Sensor_container, 22
- read_sensor
 - Sensor, 21
 - Sensor_container, 22
 - Temp_hum_sensor, 28
- release
 - Function_control, 6
- Request_container, 14
 - append, 15, 16
 - exists, 15
 - get_no_param, 15
 - get_with_param, 15
 - has_params, 15
 - Request_container, 15
- Request_functions, 16
 - ~Request_functions, 16
 - reset, 17
 - subscribe, 17
 - unsubscribe, 17
- reserve
 - Function_control, 6
- reset
 - Request_functions, 17
- rotate
 - SADM_functions, 19
- run
 - Function_control, 5
 - Instruction, 8
- SADM_functions, 17
 - ~SADM_functions, 18
 - deg_to_steps, 20
 - instant_release, 19
 - rotate, 19
 - set_dir, 19
 - set_ratio, 19
 - set_speed, 19
 - set_step_size, 18
 - set_stepping_mode, 19
- send_ack
 - Serial_handler, 26
- send_available_data
 - Serial_handler, 25
- send_available_instructions
 - Serial_handler, 25
- send_error_message
 - Serial_handler, 26
- send_handshake_response
 - Serial_handler, 26
- send_ping
 - Serial_handler, 25
- Sensor, 20
 - ~Sensor, 21
 - get_name, 21
 - read_sensor, 21
 - Sensor, 21
- Sensor_container, 21
 - ~Sensor_container, 22
 - append_available_data, 23
 - append_data, 23
 - get_all_data, 23
 - get_available_sensors, 23
 - get_data, 23
 - get_sub_data, 23
 - read_all_sensors, 22
 - read_sensor, 22
 - Sensor_container, 22
- Serial_handler, 24
 - config_approved, 26

- connection_init_approved, [26](#)
- connection_request_approved, [26](#)
- handshake, [25](#)
- handshake_approved, [25](#)
- print_to_serial, [25](#)
- send_ack, [26](#)
- send_available_data, [25](#)
- send_available_instructions, [25](#)
- send_error_message, [26](#)
- send_handshake_response, [26](#)
- send_ping, [25](#)
- Serial_handler, [24](#)
- serial_init, [26](#)
- serial_listener, [25](#)
- serial_init
 - Serial_handler, [26](#)
- serial_listener
 - Serial_handler, [25](#)
- set_dir
 - SADM_functions, [19](#)
- set_ratio
 - SADM_functions, [19](#)
- set_speed
 - SADM_functions, [19](#)
- set_step_size
 - SADM_functions, [18](#)
- set_stepping_mode
 - SADM_functions, [19](#)
- setup
 - HWLayer, [7](#)
- subscribe
 - Request_functions, [17](#)
- Temp_hum_sensor, [27](#)
 - ~Temp_hum_sensor, [28](#)
 - read_sensor, [28](#)
 - Temp_hum_sensor, [27](#)
- Uf_param, [28](#)
 - Uf_param, [28](#)
- unsubscribe
 - Request_functions, [17](#)

Bachelor of Engineering

Project appendix

Winter semester 2019

Control Cabinet - Hardware components

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Erica Fegri
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains the list of components included in the Diagnostics system control cabinet.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

Control Cabinet: List of Hardware bought from suppliers

Phoenix Contact:				
Artikkelnummer	Type	Beskrivelse	Bestillingsmengde	Enhet
2900443	PLC-BPT- 5DC/21	Relesokkel		1 STK
2961367	REL-MR 4,5DC/21	Enkeltrelé		1 STK
2904602	QUINT4-PS/1AC/24DC/20	Strømforsyning		1 STK
2908262	PTCB E1 24DC/1-8A NO	Elektronisk automatsikring		5 STK
3211757	PT 4	Gjennomgangsklemme		1 STK
3211760	PT 4 BU	Gjennomgangsklemme		2 STK
3211766	PT 4-PE	Jordingsrekkeklemme		2 STK
1077082	PT 4-FE	Gjennomgangsklemme		5 STK
3030420	D-ST 4	Endedeksel		1 STK
3209510	PT 2,5	Gjennomgangsklemme		40 STK
3030417	D-ST 2,5	Endedeksel		10 STK
3209536	PT 2,5-PE	Jordingsrekkeklemme		10 STK
3030336	FBS 2-6	Stikkbro		1 STK
3036932	FBS 2-6 BU	Stikkbro		1 STK
3022276	CLIPFIX 35-5	Endeholder		20 STK
0811969	KLM 3	Holder for klemmelistermerke		10 STK
1050017:0001	ZB 5,LGS:FORTL.ZAHLEN 1-10	Merkeremse		10 STK
1050017:0011	ZB 5,LGS:FORTL.ZAHLEN 11-20	Merkeremse		10 STK
1050017:0021	ZB 5,LGS:FORTL.ZAHLEN 21-30	Merkeremse		10 STK
1050017:0031	ZB 5,LGS:FORTL.ZAHLEN 31-40	Merkeremse		10 STK
1050017:0041	ZB 5,LGS:FORTL.ZAHLEN 41-50	Merkeremse		10 STK
1051016:0001	ZB 6,LGS:FORTL.ZAHLEN 1-10	Merkeremse		10 STK
0899361	E PE A 405X400X201	Koblingsboks		1 STK
0899444	AE MP SH EP 405X400	Monteringsplate		1 STK
3002638	NS 35/ 7,5 UNP AE EP 250X400	Monteringsskinne uperforet		3 STK
1424475	G-INS-M16-T68N-PNES-GY	Kabelnippel		10 STK
1417660	A-INL-M16-P-LG	Kontramutter		10 STK
2320018	DC/DC-omformer - MINI-PS- 12- 24DC/ 5-15DC/2 - 2320018			1 sk
OEM Motor				
7404201060	Motor	1 stk		508 SEK
EM-KP72-87.5	DIN-skinnefeste	1 stk		112 SEK
EM-314-12-24V	Driver	1 stk		1379 SEK
EM-328	Parametersetter	1 stk		460 SEK
Elfa				
YW1S-33E20	Vribryter med fjærretur		Artikkelnummer: 301-02-804	104kr
Contelec VERT-X 2831-736-221-102		Posisjonssensor	164-90-125	641kr
Private				
Allen Bradley	Solid state relays	2 stk		
Fuse	20 A	1 stk		

Bachelor of Engineering

Project appendix

Winter semester 2019

SADM Drivetrain requirements

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Erica Fegri, Rita Hogstad, Ming Kit Wong, Håvar Østrem
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains requirements connected to SADM Drivetrain with dated comments and discussion topics connected to each requirements.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

SADM - Drivetrain

Target release	
Epic	SADM - Drivetrain USN5-111 TO DO
Document status	DRAFT
Document owner	Erica Fegri
Designer	Ming Kit Wong Rita Hogstad
Tech lead	Ming Kit Wong Rita Hogstad
Technical writers	Ming Kit Wong Rita Hogstad
QA	Ming Kit Wong Rita Hogstad

Requirements

ID	Description	Importance
R2-01	The SADM shall link the solar array wing mechanically to the spacecraft structure during launch and all operational phases	DISCARDED

User Story

USN5-119 - As a mechanical engineer I have to design the shaft so that solar arrays can be attached to the spacecraft.

DISCARDED

Date	Discussion
RH 16.03.19	Not met - outside our scope.
Verification ID	Verification method
AV-01	Analysis

ID	Description	Importance
----	-------------	------------

R2-09	<p>The SADM shall be compliant with the solar array wing inertia as follows:</p> <ul style="list-style-type: none"> - I_{xx} = up to 0,013 kgm² - I_{yy} = up to 15 kgm² - I_{zz} = up to 15 kgm² <p>Note: The values for the solar array Moment of Inertia are provided at the SADM Interface</p>	SHOULD
-------	---	---------------

User Story

USN5-124 - The design of interface between solar array and SADM and motor must be adapted to the weight and size of solar arrays so that it functions as intended. **DISCARDED**

Date	Discussion
MKW 20.03.19	The values are updated to fit the solar array measurements and weight.
Verification ID	Verification method
AV-02	Analysis and calculations

ID	Description	Importance
R2-18	<p>The SADM shall keep its position under the following spacecraft rotational accelerations:</p> <ul style="list-style-type: none"> - Powered: +- 0,02 rad/s² (TBC) - Unpowered: +- 0,01 rad/s² (TBC) <p>An amplification factor of 2 shall be considered</p>	DISCARDED
Date	Discussion	

EF 07.12.19	Review with SH: <ul style="list-style-type: none"> ▪ Shall the solar arrays keep its position against the sun? (Position locked relative to the sun) ▪ Or shall the shaft be locked in position during rotational acceleration? (Position locked relative to the spacecraft)
EF 07.03.19	Review with SH: <ul style="list-style-type: none"> • Shall the solar arrays keep its position against the sun? (Position locked relative to the sun) • Or shall the shaft be locked in position during rotational acceleration? (Position locked relative to the spacecraft) <p>Review with external supervisor 06.03.19:</p> <ul style="list-style-type: none"> • The shaft shall be locked in position during rotational acceleration - position locked relative to the spacecraft. <p>This requirement is discarded.</p>

User Story

USN5-134 - The drivetrain shall keep its position under spacecraft rotational accelerations so that the solar arrays holds it's positions.

DISCARDED

Verification ID	Verification method
AV-04	Discarded

ID	Description	Importance
R2-19	The SADM shall be able to operate in normal angular speed mode between 0,0 X degrees/s to X degrees/s. The maximum output angular velocity capability shall be at least 3 degrees/s	MUST
Date	Discussion	
HØ 07.02.19	<ul style="list-style-type: none"> • Choose values for X (Depends on motor parameters?) • What is the difference between "normal angular speed mode" and "nominal speed"? • Is there a conflict between this requirement and R2-17? • How much real-time variation in speed and acceleration is required? 	



EF 12.05.19	The motor and mechanical interfaces tolerates much higher speed. The speed is controlled by DS.
-------------	---

User Story

USN5-151 - The power output, the angular velocity and velocity resolution from the motor, as well as the gearing ratio, gear stiffness, and shaft stiffness should all be of adequate capacity for the SADM to rotate the shaft connected to the solar arrays with adjust...

DISCARDED

Verification ID	Verification method
TV-14	Test and inspection

ID	Description	Importance
R2-21	The SADM shall provide a full step resolution at output shaft <1 degrees	SHOULD

User Story

USN5-119 - As a mechanical engineer I have to design the shaft so that solar arrays can be attached to the spacecraft.

DISCARDED

Date	Discussion
EF 19.05.19	The motor's full step size is 1.8 degrees. Gear ratio is 2/1 (driven/driving). Resolution on solar array wings: $1.8 \text{ deg} / 2 = 0.9 \text{ deg}$ output resolution. Step size can be adjusted to higher resolutions as well.

Verification ID	Verification method
AV-05	Analysis.

ID	Description	Importance
R3-01	The mass of the SADM shall not exceed 2kg (Harness not included)	MUST

Date	Discussion
HØ 08.02.19	<ul style="list-style-type: none"> Is this requirement final?

User Story

USN5-155 - The total mass of the SADM, excluding wiring, must be two kilograms or less. This means the drivetrain can only be a certain percentage of this mass and other massive components must be taken into consideration when designing the drivetrain. **DONE**

Verification ID	Verification method
-----------------	---------------------

AV-06 TV-21	Test with weight, analyse using SolidWorks.
----------------	---

ID	Description	Importance
R3-02	The SADM shall not exceed an envelope of 100 X 100 X 200 (millimeters)	MUST

Date	Discussion
HØ 08.02.19 EF	<ul style="list-style-type: none"> 100 mm x 100 mm x 200 mm or 95 mm x 95 mm x 200 mm? 100 mm x 100 mm x 200 mm or 95 mm x 95 mm x 200 mm.

User Story

USN5-156 - The outer spatial dimensions of The Housing of the SADM must be exactly one hundred by one hundred by two hundred millimeters. The inner spatial dimensions depends on the thickness of the housing walls. The drivetrain and other components inside the SADM... **DONE**

Verification ID	Verification method
RV-03 TV-22	Review and test.

ID	Description	Importance
R7-05	The SADM driveline shall be single point failure free except for accepted mechanical parts	DISCARDED

Date	Discussion
EF 15.02.19 RH EF 19.05.19	<ul style="list-style-type: none"> What elements will this requirement include? <p>Not applicable - discarded.</p>

User Story

USN5-140 - The driveline should be single point failure free so that it will be more independent and the whole SADM won't stop working if one component stops working **DISCARDED**

Verification ID	Verification method
AV-32	Discarded

ID	Description	Importance
----	-------------	------------

R8-01	<p>The SADM shall have a minimum operational life time capability of</p> <p>- 5 years in space for 600km SSO application</p> <p>13 000 cycles (life time factors according to ECSS-E-ST-33-01 [SD1] can be applied for qualification</p>	DISCARDED
Date	Discussion	
RH 07.02.19	<ul style="list-style-type: none"> How do we test or analyse and verify this requirement? Mechanical guys; how confident are in your ability to do just that? 	
EF 02.05.19	Discarded. Not able to verify, nor selected equipment.	
User Story		
<p>USN5-135 - The drive train should have a life time of 5 years in an orbit of 600km so that it will satisfy the customers</p> <p>DISCARDED</p>		
Verification ID	Verification method	
-	Discarded	



Bachelor of Engineering

Project appendix

Winter semester 2019

SADM Housing requirements

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Erica Fegri, Rita Hogstad, Ming Kit Wong, Håvar Østrem
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains requirements connected to SADM Housing with dated comments and discussion topics connected to each requirements.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

SADM - Housing

Target release	
Epic	SADM - Housing USN5-113 TO DO
Document status	DRAFT
Document owner	Erica Fegri
Designer	Rita Hogstad Ming Kit Wong
Tech lead	Rita Hogstad Ming Kit Wong
Technical writers	Rita Hogstad Ming Kit Wong
QA	Rita Hogstad Ming Kit Wong

Requirements

ID	Description	Importance
R2-08	The SADM physical reference frame shall be as defined below and depicted in following figure: <ul style="list-style-type: none"> - The origin shall be at the intersection of the SADM mounting plane with the S/C interface - The X-axis shall be in point - The Y-axis shall be coincident - The Z-axis shall be in the mounting plane and point through a mounting point 	DISCARDED
Jira Issue	User Story	
USN5-123 DONE	The SADM physical reference should be illustrated in a sketch so that it's orientation is easy to understand.	
Verification ID	Verification method	
RV-02	Discarded	

ID	Description	Importance
R2-11	The SADM shall be able to oscillate or rotate around the X- axis in both directions (CW or CCW), with a total angular range of minimum +- 180 degrees	HIGH
Jira Issue	User Story	
<div style="border: 1px solid black; padding: 2px;"> USN5-130 DISCARDED </div>	The SADM shall be able to oscillate or rotate 180 degrees around the X-axis in both directions so that it always can point the solar arrays against the sun	
Date	Discussion	
EF 19.05.19	Mechanically verified, must be verified with flex implemented. Waiting for flex to arrive.	
Verification ID	Verification method	
TV-08 RV-04	Test and analysis.	

ID	Description	Importance
R8-01	<p>The SADM shall have a minimum operational life time capability of</p> <ul style="list-style-type: none"> - 5 years in space for 600km SSO application <p>13 000 cycles (life time factors according to ECSS-E-ST-33-01 [SD1] can be applied for qualification</p>	DISCARDED
Jira Issue	User Story	
<div style="border: 1px solid black; padding: 2px;"> USN5-138 DISCARDED </div>	The housing should have a life time of 5 years in an orbit of 660km so that it will satisfy the customers	
Verification ID	Verification method	
-	Discarded	

ID	Description	Importance
R2-22	The SADM shall provide an axial translation stiffness of $>- 1.0 \times 10^8$ N/m	DISCARDED
Jira Issue	User Story	
USN5-142 DISCARDED	The SADM shall provide an axial translation stiffness of $>- 1.0 \times 10^8$ N/m	
Date	Discussion	
RH MKW 10.05.19	Not prioritized	
Verification ID	Verification method	
TV-16 AV-13	Discarded.	

ID	Description	Importance
R2-23	The SADM shall provide a radial translation stiffness of $>- 1.0 \times 10^8$ N/m with the solar array interface constrained in cross axis bending (interface plane remains parallel under loading)	DISCARDED
Jira Issue	User Story	
USN5-143 DISCARDED	The SADM shall provide a radial translation stiffness of $>- 1.0 \times 10^8$ N/m with the solar array interface constrained in cross axis bending (interface plane remains parallel under loading)	
Verification ID	Verification method	
TV-17 Av-14	Discarded.	
Date	Note	
RH MKW 10.05.19	Not prioritized.	

ID	Description	Importance
R2-24	The SADM shall provide a cross axis bending stiffness of $>- 1.0 \times 10^5$ Nm/rad	DISCARDED

Jira Issue	User Story
<div style="border: 1px solid gray; padding: 2px;"> USN5-144 DISCARDED </div>	The SADM shall provide a cross axis bending stiffness of $>- 1.0 \times 10^5$ Nm/rad
Verification ID	Verification method
TV-18 AV-15	Discarded.
Note	
MKW RH 10.05.19	Not prioritized.

ID	Description	Importance
R2-25	<p>The SADM shall provide a torsional stiffness of $>_ 10^4$ Nm/rad (assuming a blocked motor shaft, at 5 to 15 Nm external load on the output shaft</p> <p>In the SADM backlash angular range (SADM-REQ-00170) with low loading (only overcoming frictional torque) an area with "zero stiffness" is expected and therefore not considered as a deviation to the requirement</p>	DISCARDED
Jira Issue	User Story	
<div style="border: 1px solid gray; padding: 2px;"> USN5-145 DISCARDED </div>	The SADM shall provide a torsional stiffness of $>_ 10^4$ Nm/rad (assuming a blocked motor shaft, at 5 to 15 Nm external load on the output shaft	
Verification ID	Verification method	
TV-19 Av-16	Discarded.	
Note		
MKW RH 10.05.19	Not prioritized.	

ID	Description	Importance
----	-------------	------------



R2-26	The SADM shall have a first Eigen-frequency > 250 Hz when its stator part is mounted on rigid interfaces and when its rotor part is connected to a dummy mass of minimum 2,5 kg with CoG placed at minimum 20 mm from the SADM I/F	DISCARDED
Jira Issue	User Story	
<div style="border: 1px solid black; padding: 2px;"> USN5-147 DISCARDED </div>	Our product is to be mounted on a 8u-dummy, with the center of gravity placed at minimum 20 mm from the SADM interface, during vibration testing. The SADM shall have a first Eigen-frequency greater than 250 Hz to fulfill this requirement.	
Verification ID	Verification method	
TV-20 AV-17	Discarded	
	Note	
MKW RH 10.05.19	Does this one needs review? It seems clear to me (MKW) Not prioritized.	

ID	Description	Importance
R6-03	The SADM needs to include preferred venting path to allow for the typical depressurization profiles.	DISCARDED
Jira Issue	User Story	
<div style="border: 1px solid black; padding: 2px;"> USN5-149 DISCARDED </div>	The housing needs to include venting paths so that there will be low pressure in the SADM	
Verification ID	Verification method	
RV-25	Discarded	

ID	Description	Importance
R5-21	Assuming the SADM being an internal equipment, external materials shall withstand a TID level of 10 Mrad.	DISCARDED
Jira Issue	User Story	
<div style="border: 1px solid black; padding: 2px;"> USN5-150 DISCARDED </div>	The materials of the housing should meet the requirement of Total Ionising Dose (TID) level of 10 Mrad (Milliradians)	

Verification ID	Verification method
RV-22	Discarded



Bachelor of Engineering

Project appendix

Winter semester 2019

SADM Motor requirements

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Erica Fegri, Rita Hogstad, Ming Kit Wong, Håvar Østrem
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains requirements connected to SADM Motor with dated comments and discussion topics connected to each requirements.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

SADM - Motor

Target release	
Epic	SADM - Motor USN5-112 TO DO
Document status	DRAFT
Document owner	Erica Fegri
Designer	Erica Fegri Rita Hogstad Håvar Østrem Ming Kit Wong
Tech lead	Erica Fegri Rita Hogstad Håvar Østrem Ming Kit Wong
Technical writers	Erica Fegri Rita Hogstad Håvar Østrem Ming Kit Wong
QA	

Objective

This group of requirements consists of requirements that directly concern the motor.

Requirements

ID	Description	Importance
R2-03	The SADM shall hold the solar array wings in position when not powered	SHOULD
User Story		
USN5-120 - Our product needs to hold the solar array wings in position when not powered DISCARDED		
Date	Discussion	
EF 19.05.19	Hybrid stepper motor - permanent magnets. The motor has unpowered holding torque, but this is unknown. The motor is considered not to fulfill this requirement on ground due to gravity. In orbit this will presumably be fulfilled.	
Verification ID	Verification method	
TV-02	Take measurement of the position with sensors or Diagnostic System.	

ID	Description	Importance
R2-10	<p>The SADM functional performance and motorization margin shall be determined considering the following solar array characteristics when attached to the SADM:</p> <ul style="list-style-type: none"> - 1st torsional mode of the SA: 0,2 Hz with inertia of 31 kgm² - 2nd torsional mode of the SA: 0,7 Hz with inertia of 30 kgm² - 3rd torsional mode of the SA: 3,7 Hz with inertia of 33 kgm² - Considered SA modal dampning 0,3% or Q = 166 - SA residual inertia: 16 kgm² Mass 110 kg +/- 5% - Mass: 0.40kg +/- 5% 	SHOULD
Date	Discussion	
RH 12.05.19	Do not have software for verification of this requirements.	
User Story		
<p>USN5-125 - Our product needs to perform and function with the given values for a solar array.</p> <p>DISCARDED</p>		
Verification ID	Verification method	
TV-07 AV-07	Tests failed in Solidworks.	

ID	Description	Importance
R2-12	The SADM shall satisfy motorization margin requirement as specified in §4.7.5.3 per ECSS-E-ST-33-01, [SD1]	CAN
Date	Discussion	
EF 19.05.19	The motor is NA for this requirement, due to the standard require vacuum compability and more. For the prototype a space graded motor is not considered as necessary or affordable.	
User Story		
<p>USN5-126 - The European standard for the space industry must be followed when selecting and designing the placement of the motor. Other requirements from the specified standard must be followed as well. DISCARDED</p>		
Verification ID	Verification method	

AV-03	Not tested. Discarded.
-------	------------------------

ID	Description	Importance
R2-13	The unpowered holding torque of the SADM shall be greater than or equal to 2 (TBC) Nm	CAN
Date	Discussion	
EF 13.02.19	TBC - To be considered.	
EF 02.05.19	<ul style="list-style-type: none"> ▪ This requirement is NA for the components implemented in the SADM. ▪ Hybrid stepper motor - permanent magnets. The motor has unpowered holding torque, but this is unknown. ▪ Test for this was not achievable at the time for documentation hand-in, and was not prioritized. 	

User Story

USN5-127 - The motor must be able to hold the solar array in the same position with a load of 2Nm or greater, when it is not unpowered. **DISCARDED**

Verification ID	Verification method
TV-09 AV-08	Discarded

ID	Description	Importance
R2-14	<p>The SADM shall be able to deliver a minimum holding torque of 8 Nm with a motor running at a driving current of 150 mA</p> <p>The SADM shall be able to deliver a minimum holding torque of 0.04 Nm with a motor running at a driving current of 1A.</p>	SHOULD
Date	Discussion	

EF 06.03.19	Review with external supervisor 06.03.19:
EF 26.04.19	<ul style="list-style-type: none"> ▪ The exact values for holding torque are to be changed. 8 Nm is not a valid value. ▪ The holding torque for the motor we have can be tested. <p>$T = Kt \cdot I$ $Kt = 0.083$ (from datasheet.) Holding torque at motor shaft (gear ratio not considered) when 150 mA:</p> <p>$T = 0.083 \cdot 0.150 \text{ A} = 0.0125 \text{ Nm}$ holding torque.</p>
EF 14.05.19	<ul style="list-style-type: none"> ▪ Gear ratio = 2 (driven/drive) ▪ Holding torque at solar arrays: 6.23 mNm. Gear ratio chosen to be as high as possible in order to make torque on solar arrays when rotating high. ▪ Leads to less holding torque. <p>Running on 1A, this requirement is verified with analysis, calculations.</p>

User Story

USN5-128 - The motor must be able to hold the solar array in the same position with a load of 8 Nm, when the motor is running with a current of 150 mA. **DISCARDED**

Verification ID	Verification method
TV-10	Calculations
AV-09	

ID	Description	Importance
R2-15	The SADM irreversibility quasi-static torque shall be less than 70 (TBC) Nm	SHOULD
Date	Discussion	

EF 07.03.19	Review with external supervisor 06.03.19: <ul style="list-style-type: none"> ▪ It means that if the quasi-static torque exceeds this maximum value, the SADM must rotate in order to avoid any degradation. ▪ The background for this requirement is on-ground handling. ▪ This is related to the holding torque for motor, gear and shaft without power. ▪ The gear should withstand back winding. ▪ We should consider implementing a physical stop mechanism i order to protect the flexprint. 	
EF 16.05.19		The force on motor with a gear ratio 2 will be $2 \cdot 70 \text{ Nm} = 140 \text{ Nm}$, which is beyond motor powered and unpowered holding torque with good margin.

User Story

USN5-129 - The mechanism must allow movement when load exceeds 70 Nm
DISCARDED

Verification ID	Verification method
TV-11 AV-10	Test and analysis.

ID	Description	Importance
R2-17	<p>NA:</p> <p>The SADM shall have enough torque margins to withstand a short circuit of motor winding without loss of functionality at 0,1 degrees/s (nominal speed)</p> <p>Note: using the redundant interface in case of failure in nominal winding and vice versa</p>	<div style="background-color: #ffc107; padding: 5px; display: inline-block;">NA</div>
Date	Discussion	

<p>EF 14.02.19</p> <p>EF 07.03.19</p>	<p>Questions for the external supervisor:</p> <ul style="list-style-type: none"> • Short circuit of motor winding due to start? • Is the user story correctly formulated? • How is this adressed in the datasheet? • What is SADM in this context; is it the shaft, gear ++? • What is the consequence of not meeting this requirement? <p>Review with external supervisor 06.03.19:</p> <ul style="list-style-type: none"> ▪ This requirement is NA. ▪ This requirement is relevant when having redudant power supply/ redundant motor control. Due to silplicity and that nanosats should be as simple, light and cheap as possible, we consider solutions like this excess.
---------------------------------------	--

User Story

USN5-141 - The SADM shall have enough torque margins to withstand a short circuit of motor winding without loss of functionality so that operation continues uninterrupted. DISCARDED

Verification ID	Verification method
TV-13	Test. Not applicable.

ID	Description	Importance
R2-18	<p>The SADM shall keep its position under the following spacecraft rotational accelerations:</p> <ul style="list-style-type: none"> - Powered: +- 0,02 rad/s² (TBC) - Unpowered: +- 0,01 rad/s² (TBC) <p>An amplification factor of 2 shall be considered</p>	DISCARDED
Date	Discussion	

EF 07.12.19	Review with SH:
EF 07.03.19	<ul style="list-style-type: none"> • Shall the solar arrays keep its position against the sun? (Position locked relative to the sun) • Or shall the shaft be locked in position during rotational acceleration? (Position locked relative to the spacecraft) <p>Review with external supervisor 06.03.19:</p> <ul style="list-style-type: none"> ▪ The shaft shall be locked in position during rotational acceleration - position locked relative to the spacecraft.

User Story

USN5-136 - The motor shall keep its position under spacecraft rotational accelerations so that the solar arrays holds it's positions.

DISCARDED

Verification ID	Verification method
AV-04	Discarded.

ID	Description	Importance
R2-20	<p>The SADM angular speed at the output shaft shall have a stability performance better than:</p> <ul style="list-style-type: none"> - 30 arcsec/s for frequency lower than 0,5 Hz - 0,5 arcsec/s in the frequency rnge from 0,5 Hz to 3 Hz - 1000 arcsec/s for frquency higher than 3 Hz 	CAN
Date	Discussion	
EF 17.02.19	<p>Review with external supervisor 06.03.19:</p> <ul style="list-style-type: none"> ▪ We can assess this requirement our selves. ▪ We should consider eigenfrequency <p>This is not applicable.</p>	
EF 07.03.19		
RH 25.04.19		

User Story

USN5-133 - The shaft is rotating with a speed that must have a stability greater than the values given in the requirements.

DISCARDED

Verification ID	Verification method
TV-15	Not tested, not applicable.

ID	Description	Importance
R8-01	The SADM shall have a minimum operational life time capability of - 5 years in space for 600km SSO application 13 000 cycles (life time factors according to ECSS-E-ST-33-01 [SD1] can be applied for qualification	SHOULD

Date	Discussion

User Story

USN5-137 - The motor should have a life time of 5 years in an orbit of 660km so that it will satisfy the customers

DISCARDED

Verification ID	Verification method

ID	Description	Importance
R4-02	The zero reference position shall be clearly marked on the SADM rotor and stator and included on the Interface Control Drawing	DISCARDED

Date	Discussion
RH 19.05.19	Interface control drawing is not included. Zero reference position is clearly marked on the parts.

User Story

USN5-158 - The motor of the SADM should be marked with a zero reference point so that it will be easier to mate the assembly

DONE

Verification ID	Verification method
RV-06	Discarded.

ID	Description	Importance
----	-------------	------------

R4-08	<p>The stepper motor electrical shall have the following nominal electrical interface:</p> <ul style="list-style-type: none"> - 2 phase, bipolar communication with 1.8 degrees full step angle - Phase resistance: 18.5 Ohm ± 10% - Phase inductance: 37mH ± 20% - Power: < 5W <p>The stepper motor electrical shall have the following nominal electrical interface:</p> <ul style="list-style-type: none"> - 2 phase, bipolar communication with 1.8 degrees full step angle - Phase resistance: 3.5 ohm/phase - Phase inductance: 1.2 mH/phase 	SHOULD
-------	---	---------------

Date	Discussion
EF 06.03.19	<p>Review with external supervisor 06.03.19:</p> <ul style="list-style-type: none"> ▪ 2 phase, bipolar communication with 1.8 degrees full stepangle OK. ▪ Values for phase resistance and phase inductanse will be revisioned, adapted to a stepper motor thar fulfills the other requirements. ▪ Power < 5W; Verification will tell if this value is valid.

User Story

USN5-161 - The motor should have specific nominal electrical interfaces so that it will satisfy the SADM power needs

DISCARDED

Verification ID	Verification method
RV-11	Review.

ID	Description	Importance
R4-09	The motor interface shall be according to standard NEMA dimensions.	CAN

User Story

USN5-162 - The motors interface should be designed/chosen by standard sizes(NEMA) so that it will be easier to connect to other standard products **DISCARDED**

Date	Discussion
EF	<p>NEMA 17 - motor shall have flange 1.7 x 1.7 inches. = 43.18 x 43.18 mm.</p> <p>SADM-motor has 42.0 x 42.0 mm flange.</p>

Verification ID	Verification method
RV-12	Review. Almost passed.



Bachelor of Engineering

Project appendix

Winter semester 2019

SADM Power & signal transfer

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Erica Fegri, Rita Hogstad, Ming Kit Wong, Håvar Østrem
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains requirements connected to SADM Power and signal transfer with dated comments and discussion topics connected to each requirements.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

SADM - Power and signal transfer

Target release	
Epic	SADM - Power and signal transfer USN5-114 TO DO
Document status	DRAFT
Document owner	Erica Fegri
Designer	Erica Fegri Håvar Østrem
Tech lead	Erica Fegri Håvar Østrem
Technical writers	Erica Fegri Håvar Østrem
QA	Erica Fegri Håvar Østrem

Objective

This subsystem concerns power and signal to the motor that rotates the shaft to the solar arrays, as well as transferring the power from the solar arrays through the SADM and to the interface for power transferring between SADM and spacecraft. We need to consider factors like the amount of power, how to transfer the power due to rotational motion, interfaces and more.

Requirements

ID	Description	Importance
R2-04	The SADM shall provide angular position information. The DS shall provide angular information.	MUST
Date	Discussion	
EF 15.02.19	<ul style="list-style-type: none"> Angular from motor drive? Sensor? 	
EF 08.04.19	The sensor is considered to not be a part of the SADM, it will be a part of DS.	

User Story

USN5-121 - Information about the solar array position shall be available so that the rest of the system (DS) can use this information.

DONE

Verification ID	Verification method
-----------------	---------------------

TV-03	Test
-------	------

ID	Description	Importance
----	-------------	------------

R2-05	The SADM shall transfer solar array grounding lines	MUST
-------	---	-------------

Date	Discussion
------	------------

EF 15.02.19	<ul style="list-style-type: none"> ▪ Grounding lines dimensioned with same cross-section as power transfers. ▪ Solar array grounding lines are seperated from SADM grounding.
-------------	---

User Story

USN5-122 - The SADM shall transfer solar array grounding lines to ensure grounding

DISCARDED

Verification ID	Verification method
-----------------	---------------------

TV-04	Inspection and review.
-------	------------------------

ID	Description	Importance
----	-------------	------------

R4-03	The length of the rotor harness shall be 500 mm flying leads	MUST
-------	--	-------------

User Story

USN5-159 - As an electrical engineer, I shall make sure the rotor harness have 500mm flying leads so that it will be easy to connect the leads to future connectors. **DISCARDED**

Verification ID	Verification method
-----------------	---------------------

RV-07	Review.
-------	---------

ID	Description	Importance
----	-------------	------------

R4-05	The maximum total current transfer in the power lines shall be 20A forward and 20A return; - 10A each direction each solar array	MUST
-------	--	-------------

Date	Discussion
HØ 08.02.19	<ul style="list-style-type: none"> Does all 4 electrical paths go through the entire electrical SADM subsystem (meaning 4 x 10A-flexes) or are they connected in the shaft (meaning 2 x 20A flexes)? <p>Erica: Up to us?</p> <ul style="list-style-type: none"> There are two Solar array wings, one on each end of the SADM. Both solar arrays are electrically connected to the satellite body via the SADM subsystem through two electrical paths. Four paths in total. These paths consist of: Wires through the hollow shaft, a connection between wires on the inside of the shaft and a flex connected to the outside of the shaft, the flex itself, a connection between the flex and a housing wall separating the SADM from the rest of the satellite body and an interface through the housing wall and the rest of the satellite body. Every path and its components and connections must be able to withstand an electric current of ten ampere without overheating, breaking or inducing high levels of induction or other electromagnetic effects that could disturb the functionality of other electrical components.

User Story

USN5-210 - As an electrical engineer, I need to design the cables, flex and equipment to have adequate conductivity.

DISCARDED

Verification ID	Verification method
TV-23	FYI - requirement.

ID	Description	Importance
R4-06	Electrical grounding between SADM shaft and housing shall be redundant.	SHOULD
Date	Discussion and decisions	



EF 08.02.19	<p>Requirement group Electrical interface</p> <p>Redundant:</p> <ul style="list-style-type: none"> ▪ Electrical grounding trough cable (find standard for determine dimension of cable)? ▪ Grounding through mechanical contact? - Material conductivity ▪ Test: Kontinuitetstest? (Find standard for verification method?) ▪ Grounding and EMC - research
EF 19.05.19	<p>This requirement is not met because of mechanically challenges. The flex has two traces reserved for this purpose. Possible solutions described in report.</p>

Jira Issue	User Story
------------	------------

USN5-164 - As an electrical engineer, I have to ensure sufficient grounding to maintain optimal operation of the SADM/spacecraft
DISCARDED

Verification ID	Verification method
-----------------	---------------------

RV-09	Review. Not met when documentation hand-in.
-------	---

ID	Description	Importance
R4-07	Solar Array grounding lines shall be insulated from SADM grounding	MUST

Date	Discussion and decisions
------	--------------------------

EF 08.02.19	<p>The grounding lines from the solar arrays must be insulated from SADM grounding, and need to be considered when determine the electrical interface for power between SADM and spacecraft/DS.</p> <p>QA:</p> <ul style="list-style-type: none"> • Commom earth and minus? No.
EF 12.05.19	<ul style="list-style-type: none"> ▪ The power tranfers supprts own leads for SA grounding.

User Story

USN5-160 - As an electrical engineer, I need to make sure the grounding lines from the solar arrays are seperated from the SADM grounding. DISCARDED

Verification ID	Verification method
-----------------	---------------------

--	--

RV-10	Review.
-------	---------

ID	Description	Importance
R4-11	The SADM power consumption shall not exceed 5 Watts. (TBC) The SADM power consumption shall not exceed 24 Watts. (TBC)	SHOULD

Date	Discussion and decisions
EF 09.02.2019	<ul style="list-style-type: none"> How many Watts? - Find this value. Example; 5 Watt.
EF 02.05.19	<p>Current can be controlled in motor driver. Depending on current level.</p> <ul style="list-style-type: none"> Current level should be as low as possible in order to minimize heat. Since the pull-out torque depends on the applied current level from motor driver, the current level has to be regulated when testing SADM complete assembled with dummy-loads to imitate solar arrays.

Jira Issue	User Story
------------	------------

USN5-165 - The SADM power consumption shall not exceed a certain level, so that the spacecraft's total power consumption does not exceed the total capacity. **DISCARDED**

Verification ID	Verification method
AV-19	Test when motor is running with given load at given angular velocity.

ID	Description	Importance
R7-03	The SADM power transfers shall meet with the derating performance in ECSS-Q-ST-30-11C [SD8] for nominal operation	SHOULD

User Story

USN5-148 - As an electrical engineer, I need to make sure the required derating performances of the power transfers will be adequate
DISCARDED

Verification ID	Verification method
AV-30	Analysis - See design description Electrical Harness

ID	Description	Importance
R4-03	The length of the rotor harness shall be 500 mm flying leads	HIGH



User Story

USN5-159 - As an electrical engineer, I shall make sure the rotor harness have 500mm flying leads so that it will be easy to connect the leads to future connectors. **DISCARDED**

Verification ID	Verification method
-----------------	---------------------

RV-07	Review.
-------	---------

ID	Description	Importance
R7-04	<p>The SADM power transfers shall meet the derating performance in ECSS-Q-ST-30-11C [SD8] in failure case (loss of one single wire, see definition below)</p> <p>Definition of failure case: If only one wire is lost, the increase in current applies only for one wire in the bundle and the margin is considered acceptable as long as the current is below single wired de-rated capacity.</p>	SHOULD

Date	Discussion and decisions
------	--------------------------

EF 11.02.19	
----------------	--

User Story

USN5-196 - As an electrical engineer, I should make sure the power transfers meets acceptable derating performance, so that the power transfer can still be operative in failure cause. **DISCARDED**

Verification ID	Verification method
-----------------	---------------------

AV-31	Analysis. See Design Description.
-------	-----------------------------------

ID	Description	Importance
R8-01	<p>The SADM shall have a minimum operational life time capability of</p> <p>- 5 years in space for 600km SSO application</p> <p>13 000 cycles (life time factors according to ECSS-E-ST-33-01 [SD1] can be applied for qualification</p>	DISCARDED

User Story

USN5-139 - The power transfer system should have a life time of 5 years in an orbit of 660km so that it will satisfy the customers

DISCARDED

Verification ID	Verification method
-	Discarded.



Bachelor of Engineering

Project appendix

Winter semester 2019

Project Planning

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Rita Hogstad
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

As a big part of any project there will be need for project planning. In this appendix the project planning will be explained.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

Contents

- 1 Gantt Diagram** **2**
 - 1.1 Our Gantt chart 2

- 2 Unified Chart** **4**
 - 2.1 The four phases 4
 - 2.1.1 Inception 4
 - 2.1.2 Elaboration 5
 - 2.1.3 Construction 5
 - 2.2 Workflows 5
 - 2.2.1 Project administration 5
 - 2.2.2 Research 5
 - 2.2.3 Requirements 5
 - 2.2.4 Verification 6
 - 2.2.5 Design 6
 - 2.2.6 Production 6
 - 2.3 Simplified Gantt 6

- 3 References** **8**

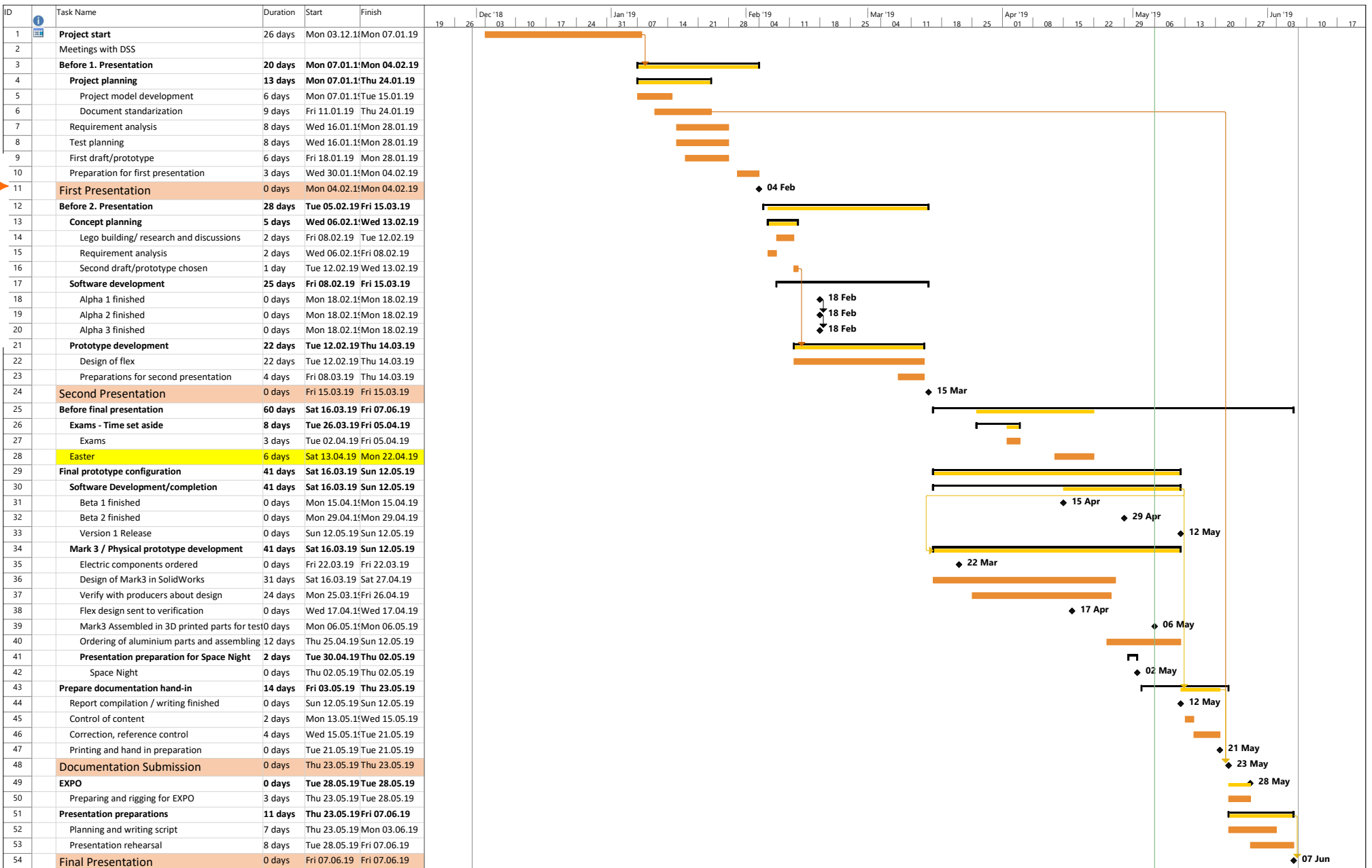
1 Gantt Diagram

The Gantt chart is a well established way of planning a project. The first Gantt charts were even used in the early 1900's[1]. We chose to use Microsoft Project to make our Gantt chart, the reason for this was that it is free via our school, it looks nice and it is easy to use. It is important to mention that this Gantt chart was initially an estimation made 30th of January 2019, so the chart was revised later in the course of the project to update it so it would be representing what was done when.

So why did we chose to make a Gantt diagram? It is a very useful tool for organizing a project. It allows the group members to know whether the project is on, ahead of, or behind schedule, and to let us know the amount of time a task is supposed to take. It is a nice way to see if some tasks are related to others, and what needs to be finished before you can start on the next task. It is a good way of informing the client of where you are in the process and what your next task will be, which makes it easier to keep the client on your team. The group becomes more productive, and it enhances communication between members[2].

1.1 Our Gantt chart

To make a Gantt chart we first decided what our milestones are, which we put as the three presentations and the document hand-in. Then we created tasks we were going to do between the milestones and estimated how long they would take. After we put in the tasks, we had to figure out which of the task were independent of each other and which tasks could be done in parallel. If two tasks are dependent of each other, we cannot start with the second one before the first one is finished. It is important to mention that our project is built on an agile process. This allow us to incorporate parts of the different process phases into multiple sprints, as well as utilizing an evolutionary development process. For us, utilizing an evolutionary development process means that we are developing several prototypes along the way, where improvements are applied to the previous one for every new one.



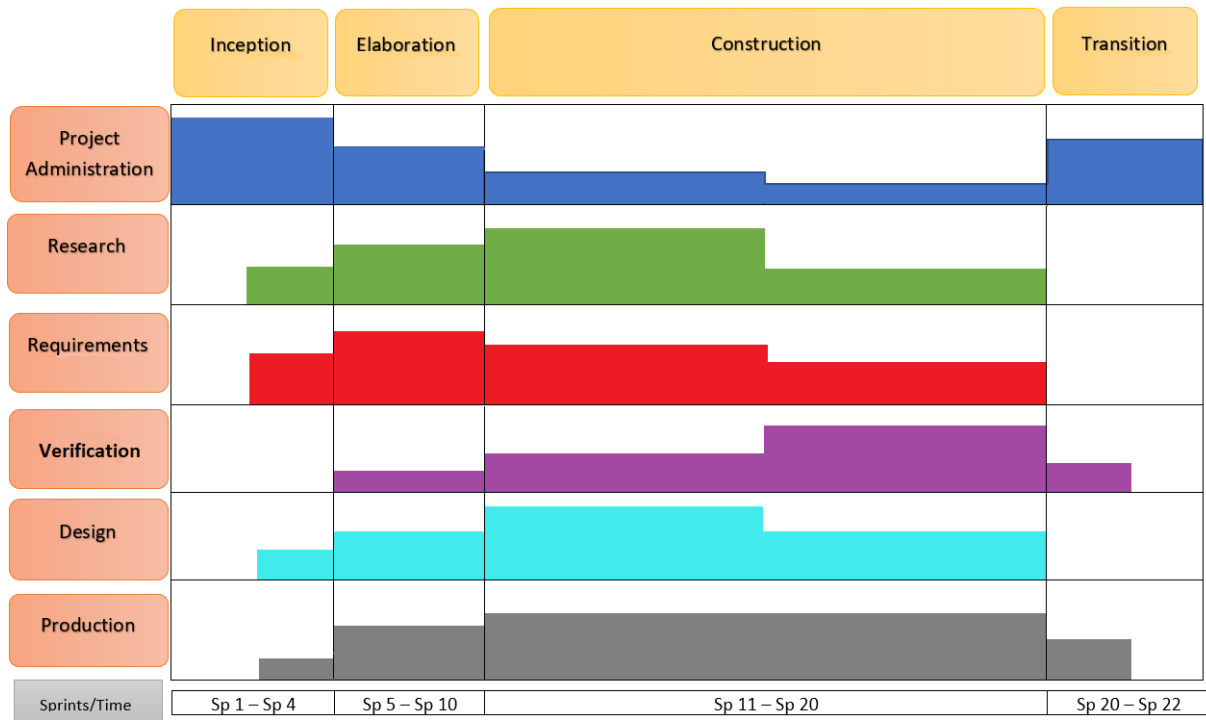
Project: Gantt diagram Bachelo
Date: Mon 06.05.19

Task	Summary	Inactive Milestone	Duration-only	Start-only	External Milestone	Manual Progress
Split	Project Summary	Inactive Summary	Manual Summary Rollup	Finish-only	Deadline	Slippage
Milestone	Inactive Task	Manual Task	Manual Summary	External Tasks	Progress	



2 Unified Chart

The diagram is read from the top, and then from left to right. In each workflow category there is an area/box with a color that corresponds to this workflow. We can see that at the Inception phase, the dark blue area of the Project administration workflow is quite large compared to the other workflows. Which means that in this inception phase, we use more time and resources on this workflow, and less time and resources on research and design. In the elaboration phase we can see that there are a big box at the requirements workflow, which means that we will use more time and resources at analysing and work on requirements in this phase.



2.1 The four phases

The Unified Process diagram is divided into four phases that focuses on different aspects of the project process.

2.1.1 Inception

This is the phase we were in before the first presentation. In the inception phase the group discussed and compared different types of process models. The V-Model, Scrum and unified process was considered, and the group gathered aspects from these among others, and combined them into our own process model. Document templates and process tools were chosen, requirements, and test templates were made and filled with requirements from DSS. All this is under the project administration category, and as shown in the diagram the group used a good amount of resources on this. The first version of the prototype was also made in this phase.

2.1.2 Elaboration

“The process of developing or presenting a theory, policy, or system in further detail”[3] This phase happens between the first and the second presentation. In this phase the group will dive deeper into the details of the project and product. More planning and requirement analysis are going to take place to better understand the stakeholders needs. The second version of the prototype is made here. There will be much more technical work in this phase than in the inception phase.

2.1.3 Construction

This phase is between the second and the final presentation. At the end of this phase our product will be finished, both the software and the prototype itself. Therefore, it will be a lot of designing, production and testing. The process model should be well practiced at this point so that the process will go smooth. Transition In the transition, or the wrap-up phase, the product is finished, and it is time to do the final “user acceptance testing” and get ready for the final presentation. There will be a lot of project administration in this phase in the form of finish writing the report and make the presentation.

2.2 Workflows

The different colored boxes/rectangles in the various workflows represent how much time and resources the group has used on the different workflows. In a typical Unified process model chart the workflows are Business modelling, requirements, analysis and design, implementation, test and deployment. Our group has chosen to rename and split up some of these workflows listed below.

2.2.1 Project administration

- Process model
- Project planning
- Documentation
- Meetings

2.2.2 Research

The group has basic knowledge in their respective disciplines so there will be need for more knowledge along the way. Especially as none of the group members has produced space compatible products before. Excercises in research can include reading, meetings and testing.

2.2.3 Requirements

Based on stakeholders wishes and needs, analyzing and processing requirements are the primary activities and the framework of the product we are going to make.

2.2.4 Verification

To ensure that the requirements are covered and done right we need to test them in different ways, and this happens in the verification workflow. We have made a test document that is traceable to the requirement document. Methods for verification can be (but are not limited to):

- **Test** - Performance test, functional test and other physical measurements
- **Analysis** - FEM, simulations, theoretical calculations and empirical evaluations
- **Review** - Double checking of documents and drawings
- **Inspection** visually inspect parts or assembly are according to drawings, non-destructive testing of materials.
- **Unit test** - Testing of the smallest parts of programming code that needs testing

2.2.5 Design

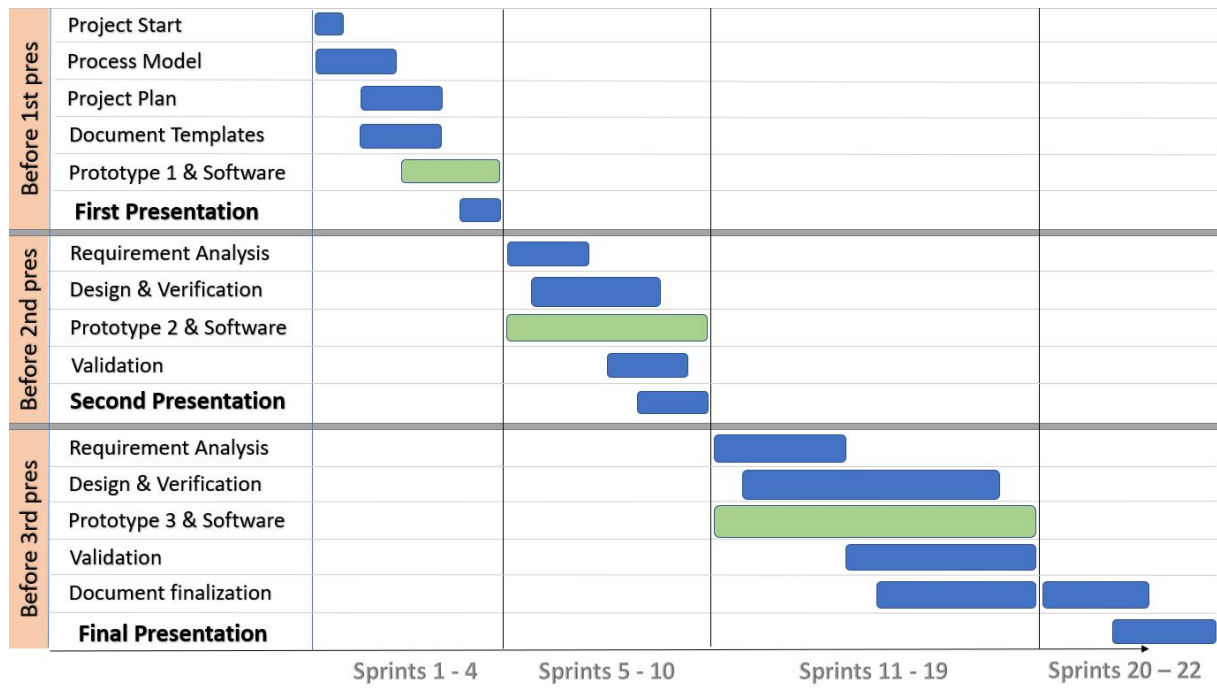
In the design workflow we draw the products' parts in SolidWorks, make circuit drawings or program the diagnostics system. Software development and Prototype development.

2.2.6 Production

When the product or program is designed we need to produce the parts or the program, assemble the prototype and make the Software.

2.3 Simplified Gantt

For the first presentation there was also made a simplified Gantt Chart based on the Unified Chart. We did this because it was easier to present, simpler and more appealing to the audience. It has the same milestones as the original Gantt, but it is less detailed because several tasks have been combined and generalized in the simplified Gantt.



3 References

- [1] T. Gantt, “What is a gantt chart?” <https://www.teamgantt.com/what-is-a-gantt-chart>.
- [2] Nishadha, “5 reasons to use gantt charts (uses of gantt charts),” <https://creately.com/blog/diagrams/5-reasons-to-use-gantt-charts/>.
- [3] O. Dictionaries, “elaboration,” <https://en.oxforddictionaries.com/definition/elaboration>.

Bachelor of Engineering

Project appendix

Winter semester 2019

Areas of Responsibility

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	SatLight
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains a list of each team members' area of responsibility and what has been done.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

Project Responsibilities

Common responsibilities

Chairperson (rolling)

- Sending meeting invitations and agenda
- Leading the meeting

Secretary (rolling)

- Writing minutes of the meeting

Rita Hogstad

Product owner

- Project planning: Gantt Diagram, Unified Process and the description of these. Overview of deadlines, when to do what.
- Main responsibility for maintaining contact between the team, the employer and our supervisors.
- Reminding the team that user stories needs to be made and followed up in the sprint board on Jira
- Arrange Sprint meetings every monday, keep track on how the progress is going, make sure that story estimation points are set for each user story.

Mechanical:

- Note: I need to emphasize that the mechanical work of the project has been a tight collaboration between Rita and Ming Kit. The individual work that has been done has been revised by each other throughout the whole process.
- Interface between electro and mechanical; make sure that the design is in line with the wishes of electro, make sure the dimensioning of the parts satisfy the needs for electrical equipment and at the same time satisfy space standards and is easily implemented in the assembly
- Tight collaboration with Ming Kit in designing the prototypes in SolidWorks and physical prototyping. Main responsibility of designing the lids on the side

of the 3D model, motor holder, found suitable gears for the assembly, connector adapter for the rigid part of the flex.

- 2D drawings of the parts in SolidWorks in collaboration with Ming Kit
- In our tight collaboration I have revised Ming Kits work and Ming Kit has revised my work, which has led to continuous changes in design at all times.
- Executed FEM analysis on critical parts(drivetrain) in SolidWorks and written about it in the main documentation. the FEM analysis contains 10G gravity force under launch, vibration analysis - which proved to be difficult, torque.
- Made user manual for the assembling of the prototype with a description of this
- Done research on materials, and written about materials in the documentation

Misc:

- Assisted in filling in requirements sheet into Google Disk and confluence. And did a big update on the VMD vs Confluence with Erica
- Made brochure for the EXPO, made the poster with Håvar

Thomas Mundal

Process manager

- Identify useful tools for managing project through process
- Setting up environments for communication and sprint management
- Educate team about agile process
- Ensure that team follows agile process
- Start daily stand-up as appropriate

Diagnostics system: Software layer

- Timeline planning
- Versioning, version control (git). Version planning, tagging and release

- Design
 - Modelling with UML
- Implementation
 - Serial communication handling towards arduino
 - WinForms UI layout
 - Gathering data on demand in real time
 - Data plotting of gathered data
 - Instruction system for instructing external devices
 - Support for hardware connection (arduino) or network connected devices (simulated SADM)
 - Automated tests of SADM unit and parameter control system (observable data values)
- Testing and verification
 - Define test parameters
 - Perform tests
 - Unit tests
 - Functional tests
 - Write verification reports
- Requirement analysis
 - Identify requirements
 - Form user stories from requirements
 - Ensure traceability from requirement to user story to verification
- Validation
 - Collab with external supervisors and support personnel to ensure that planning is going in the right direction

Miscellaneous

- SatLight XR prototype (Hololens) for validating the design of mechanical prototype.
- Simulated virtual device made with Unity for use when hardware layer lacks support for newer features.

- Tight collaboration with Jon for determining a unified communication protocol between hardware layer and software layer, and documenting this.

Erica Fegri

Requirement manager

- Develop procedure and documents for documenting and handling the requirements - educating the other team members
- Update and follow up
- Documentation

Electrical

- Control cabinet:
 - Design in Solidworks electrical
 - Find suppliers and order components for the cabinet
 - Build the cabinet
 - Documentation
- Power transfer system :
 - Leads and flex:
 - Number of leads
 - Derating
 - Connectors
 - Grounding
 - Interfaces
 - Find suppliers
 - Documentation
 - Testing connections
 - Testing - design and accomplish, review with commissioner
 - Find relative standards

- Review with suppliers and commissioner, price requests
- Bonding test of chassis
- Motor
 - Find suitable motor
 - Design the motor configuration and wiring
 - Calculate/analyze parameters suitable for configuration
 - Test
 - Documentation
- Position sensor
 - Find suitable sensor for implementation mechanically and with firmware
 - Test
 - Documentation

Håvar Østrem

Risk manager

- Making two iterations of risk management plan.
- Making a SWOT analysis.
- Making a risk calculation table

Mechatronics

- Twist Capsule
 - Collaborating with Erica, Ming Kit and Rita to agree on constraints regarding the twist capsule
 - Making a geometrical model for the flex, flex shaft and capsule housing
 - Write a mathematical formula for the length of the flex.
 - Making a calculator (the Flexculator), for handling the variable parameters of the minimum length of the flex based on the formula (radius and mounting angle of the flex shaft, radius and mounting angle of the capsule housing).

- Making a table for various flex lengths
- Making a table for various flex widths and number of flexes, based on required cross section of conductive copper
- Making a mechanical analogue of the flex for testing the approximate behavior inside of the twist capsule

Electrical

- Power transfer line
 - Calculating copper trace width and number of traces, for the flex, based on derating.
 - Drawing electrical schematic for flex, in OrCAD, Cadence CIS.
 - Calculating and design through hole connections for the flex, in Alegro Padstack editor.
 - Designing three iterations the actual flex PCB, in OrCAD, Alegro PCB editor.
 - Derating power transfer line with Erica
- Soldering wires to D-sub connectors, one flex and harness
- Bonding test of chassis and testing flex soldering with Erica

Miscellaneous

- Iteratively sketching logo designs and creating the actual logo design.
- Structuring and starting naming process, submitted name suggestions
- Designing expo-poster with Rita, expo planning
- Drawing sketches of SADM and twist capsule

Ming Kit Wong

Verification manager

- Developed a routine for verifications
- Create a system to get an overview of all the verifications and to ensure traceability in form of spreadsheet (Verification Management Document)
- Educate the team about how to follow the verification system properly
- Maintenance of the VMD

- Documentation about the verification and validation-process

Finance manager

- Bookkeeping
- Research and stay up to date about the prices in the marked within products similar to ours

Mechanical

- Made concept models in miniature and 1:1 sizes
- Designed and built Mark1 (the first prototype) for the first presentation. Had the parts cut with laser, 3D printed and taught Rita how to lathe the shaft.
- Responsible for the design of all three prototypes with Rita, where my main contribution was the structure, technical solution to the flex shaft, bearings and production of the physical parts for the prototypes..
- Collaboration with electro student/Håvar to develop a technical solution for the flex shaft, because the design of the twist capsule was based on the flex-design.
- Responsible for design and production of the test station
- Contribution of making 2D-drawings.
- Contact manufacturer for possibility for production of mechanical parts, such as Tronrud Engineering, Innovation Solution In Space (ISIS) , Mekanisk lab at KDA and Wayken.
- General mechanical consulting for the team.
- User manual

Miscellaneous

- Creating the Timesheet for the team to register working hours
- Specialized on CubeSat technology

Jon Skjelsbæk

Document manager

- Report template
- LaTeX Bibliography Guide
- Cover page template
- Appendix template
- Responsible for how deliverables are structured

Diagnostics system: Hardware layer

- Design
 - Decided to use the Arduino Mega 2560 microcontroller
 - UML class diagrams
 - C++ research
 - Involved in selecting components external to the HWL
 - Understanding how components works through datasheets
- Implementation
 - Serial communication handling towards SWL
 - Handshake protocol for connection establishment
 - Provide a list of available sensor data to the SWL
 - Provide a list of available instructions to the SWL
 - Control the SADM according to instructions received from SWL
 - Reading and providing sensor data according to requests from SWL
- Requirement analysis
 - Identify requirements
 - Form user stories from requirements
 - Ensure traceability from requirement to user story to verification
- Testing and verification
 - Define verification methods for each requirement
 - Perform defined verifications
 - Write verification reports
- Validation
 - Collab with supervisors and support personnel to ensure that planning is going in the right direction

Miscellaneous

- Tight collaboration with Thomas for determining a unified communication protocol between hardware layer and software layer, and documenting this.



Bachelor of Engineering

Project appendix

Winter semester 2019

Timesheet and time summary

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	SatLight
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains records of time spent in project. Includes individual time sheets, summary of hours spent per sprint, and a total amount of time spent in the project.

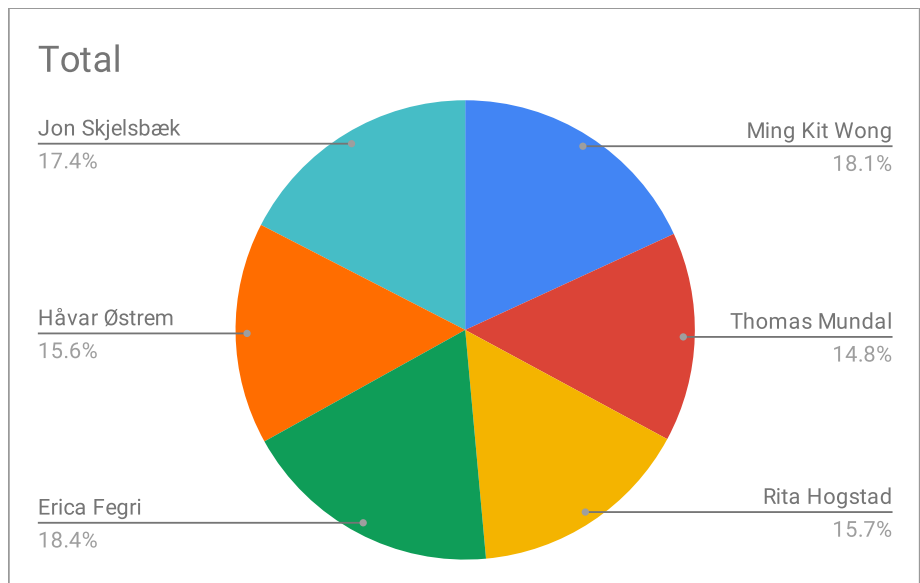
I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 22, 2019	Signature
---------------------------------------	-----------

Total in the project

	Total
Ming Kit Wong	640.5
Thomas Mundal	521
Rita Hogstad	553.5
Erica Fegri	649
Håvar Østrem	552
Jon Skjelsbæk	616
Total	3532

Hours from 2018 is added in the table above



Sprint 0

	Documentation	Technical	Presentation	Administrative	Total
Ming Kit Wong	0	0	0	12	12
Thomas Mundal	0	0	0	15	15
Rita Hogstad	0	0	0	13	13
Erica Fegri	0	0	0	13	13
Håvar Østrem	0	0	0	12	12
Jon Skjelsbæk	0	0	0	13.5	13.5
Total	0	0	0	78.5	78.5

Sprint 1

	Documentation	Technical	Presentation	Administrative	Total
Ming Kit Wong	8.5	5	5.5	14.5	33.5
Thomas Mundal	7	7	0	17.5	31.5
Rita Hogstad	3	1.5	0	22	26.5
Erica Fegri	6	0	1.5	25.5	33
Håvar Østrem	12	0	2	14	28
Jon Skjelsbæk	6	0	0	18.5	24.5
Total	42.5	13.5	9	112	177

Sprint 2

	Documentation	Technical	Presentation	Administrative	Total
Ming Kit Wong	18	7	5	10	40
Thomas Mundal	9.5	6.5	5	11	32
Rita Hogstad	4	5	2	19	30
Erica Fegri	8.5	1	1	20	30.5
Håvar Østrem	12.5	0.5	2.5	17.5	33
Jon Skjelsbæk	5	12.5	4	10.5	32
Total	57.5	32.5	19.5	88	197.5

Sprint 3

	Documentation	Technical	Presentation	Administrative	Total
Ming Kit Wong	12	10.5	17	6	45.5
Thomas Mundal	6	2	9.5	1	18.5
Rita Hogstad	21	4.5	20.5	0	46
Erica Fegri	5.5	2	25	8.5	41
Håvar Østrem	13	2	30.5	5	50.5
Jon Skjelsbæk	15	3	20	5	43
Total	72.5	24	122.5	25.5	244.5

Sprint 4

	Documentation	Technical	Presentation	Administrative	Total
Ming Kit Wong	5.5	13	3.5	2.5	24.5
Thomas Mundal	18	11.5	2	0	31.5
Rita Hogstad	9	8	3	1	21
Erica Fegri	21.5	3	3	1.5	29
Håvar Østrem	15	0	2	1	18
Jon Skjelsbæk	3	6.5	1	12.5	23
Total	72	42	14.5	18.5	147

Sprint 5

	Documentation	Technical	Presentation	Administrative	Total
Ming Kit Wong	2	12.5	0	11	25.5
Thomas Mundal	8.5	19	0	4	31.5
Rita Hogstad	3	12.5	0	5	20.5
Erica Fegri	11.5	0	0	11.5	23
Håvar Østrem	0	17.5	0	4.5	22
Jon Skjelsbæk	5	29	0	5	39
Total	30	90.5	0	41	161.5

Sprint 6

	Documentation	Technical	Presentation	Administrative	Total
Ming Kit Wong	7	22	0	2	31
Thomas Mundal	0	22	3	0	25
Rita Hogstad	2	12	0	0	14
Erica Fegri	11.5	11.5	0	9.5	32.5
Håvar Østrem	2.5	10	0	6	18.5
Jon Skjelsbæk	6	19	0	1	26
Total	29	96.5	3	18.5	147

Sprint 7

	Documentation	Technical	Presentation	Administrative	Total
Ming Kit Wong	2	24	0.5	4	30.5
Thomas Mundal	6.5	21.5	0	3	31
Rita Hogstad	3	24	0	0	27
Erica Fegri	0	22.5	0	1	23.5
Håvar Østrem	12	20	0	3	35
Jon Skjelsbæk	2	29	0	3	34
Total	25.5	141	0.5	14	181

Sprint 8 week 1

	Documentation	Technical	Presentation	Administrative	Total
Ming Kit Wong	13.5	14.5	13.5	4	45.5
Thomas Mundal	6.5	19.5	0	4	30
Rita Hogstad	9	10	15	2	36
Erica Fegri	13	18	8.5	1	40.5
Håvar Østrem	29.25	9	11	3.5	52.75
Jon Skjelsbæk	37.25	14.25	0	5	56.5
Total	108.5	85.25	48	19.5	261.25

Sprint 8 week 2

	Documentation	Technical	Presentation	Administrative	Total
Ming Kit Wong	2	9	15.5	2	28.5
Thomas Mundal	0	6.5	22	1	29.5
Rita Hogstad	4	3	30	0	37
Erica Fegri	0	4	28.5	3.5	36
Håvar Østrem	2	0	24.5	11	37.5
Jon Skjelsbæk	3	3.5	19.5	1	27
Total	11	26	140	18.5	195.5

Sprint 9

	Documentation	Technical	Presentation	Administrative	Total
Ming Kit Wong	5	2	0	11.5	18.5
Thomas Mundal	1.5	13.5	0	5	20
Rita Hogstad	0	13	0	4	17
Erica Fegri	0	14.5	0	11	25.5
Håvar Østrem	0	3	0	10.5	13.5
Jon Skjelsbæk	6.5	14.5	0	6	27
Total	13	60.5	0	48	121.5

Sprint 10 week 1 (Exams)

	Documentation	Technical	Presentation	Administrative	Total
Ming Kit Wong	0	8	0	0	8
Thomas Mundal	0	0	0	0	0
Rita Hogstad	0	0	0	2	2
Erica Fegri	0	14.5	0	13	27.5
Håvar Østrem	1	4	0	0	5
Jon Skjelsbæk	0	6	0	0	6
Total	1	32.5	0	15	48.5

Sprint 10 week 2 (Exams)

	Documentation	Technical	Presentation	Administrative	Total
Ming Kit Wong	0	0	0	0	0
Thomas Mundal	0	0	0	0	0
Rita Hogstad	0	0	0	0	0
Erica Fegri	0	3.5	0	0	3.5
Håvar Østrem	0	0	0	0	0
Jon Skjelsbæk	0	7.5	0	0	7.5
Total	0	11	0	0	11

Sprint 11

	Documentation	Technical	Presentation	Administrative	Total
Ming Kit Wong	0	38	0	3	41
Thomas Mundal	5	30	0	0	35
Rita Hogstad	0	38.5	0	0	38.5
Erica Fegri	2	16	0	12	30
Håvar Østrem	11	13.5	0	3	27.5
Jon Skjelsbæk	7.5	29.5	0	0	37
Total	25.5	165.5	0	18	209

Sprint 12

	Documentation	Technical	Presentation	Administrative	Total
Ming Kit Wong	7	18.5	0	7.5	33
Thomas Mundal	9	16	0	2	27
Rita Hogstad	0	18.5	0	0	18.5
Erica Fegri	11.5	7	0	8	26.5
Håvar Østrem	5	3.5	0	5	13.5
Jon Skjelsbæk	15	12.5	0	7	34.5
Total	47.5	76	0	29.5	153

Sprint 13

	Documentation	Technical	Presentation	Administrative	Total
Ming Kit Wong	9	36	0	3.5	48.5
Thomas Mundal	0	31.5	0	2	33.5
Rita Hogstad	21	24	0	0	45
Erica Fegri	10	11	0	11.5	32.5
Håvar Østrem	13	4	0	2	19
Jon Skjelsbæk	4.5	49	0	0	53.5
Total	57.5	155.5	0	19	232

Sprint 14

	Documentation	Technical	Presentation	Administrative	Total
Ming Kit Wong	9	19.5	11	0	39.5
Thomas Mundal	0	28.5	2	4	34.5
Rita Hogstad	5	32	3	0	40
Erica Fegri	5.5	12.5	14.5	3	35.5
Håvar Østrem	25.5	14	2	0.5	42
Jon Skjelsbæk	7	22.5	0	1	30.5
Total	52	129	32.5	8.5	222

Sprint 15

	Documentation	Technical	Presentation	Administrative	Total
Ming Kit Wong	14	29	0	3.5	46.5
Thomas Mundal	21	13	0	0	34
Rita Hogstad	50.5	8	0	0	58.5
Erica Fegri	50.5	28	0	0	78.5
Håvar Østrem	38.75	0	0	2	40.75
Jon Skjelsbæk	28.75	20.75	0	0	49.5
Total	203.5	98.75	0	5.5	307.75

Sprint 16

	Documentation	Technical	Presentation	Administrative	Total
Ming Kit Wong	15.5	47	0	1	63.5
Thomas Mundal	35	2	0	0.5	37.5
Rita Hogstad	42	0	0	0	42
Erica Fegri	40	22	0	0	62
Håvar Østrem	43	12	2	0	57
Jon Skjelsbæk	26.5	3.5	0	0	30
Total	202	86.5	2	1.5	292

Delivery week

	Documentation	Technical	Presentation	Administrative	Total
Ming Kit Wong	20	0	0	0	20
Thomas Mundal	24	0	0	0	24
Rita Hogstad	21	0	0	0	21
Erica Fegri	18.5	4	0	3	25.5
Håvar Østrem	18.25	7.5	0	0.75	26.5
Jon Skjelsbæk	22	0	0	0	22
Total	123.75	11.5	0	3.75	139

Ming Kit Wong

Ukedag	Dag	Måned	År	Sprint nr	Timer	Kommentar	Dokumentering	Teknisk arbeid	Presentasjon	Administrativt arbeid	
Tirsdag		1	Januar	2019		0.0					
Onsdag		2	Januar	2019		0.0					
Torsdag		3	Januar	2019		0.0					
Fredag		4	Januar	2019		0.0					
Lørdag		5	Januar	2019		0.0					
Søndag		6	Januar	2019		0.0					
Mandag		7	Januar	2019		1.0				1	
Tirsdag		8	Januar	2019		0.0					
Onsdag		9	Januar	2019		2.5				2.5	
Torsdag		10	Januar	2019		3.5				3.5	
Fredag		11	Januar	2019		5.0				5	
Lørdag		12	Januar	2019		0.0					
Søndag		13	Januar	2019		0.0					
Mandag		14	Januar	2019	1	7.5					
							3.5			4	
Tirsdag		15	Januar	2019	1	2.0					
							2				
Onsdag		16	Januar	2019	1	6.0					
							0		5	1	
Torsdag		17	Januar	2019	1	7.5					
							0	3	0.5	4	
Fredag		18	Januar	2019	1	7.5					
							0	2		5.5	
Lørdag		19	Januar	2019	1	3.0					
							3				
Søndag		20	Januar	2019	1	0.0					
							0				
Mandag		21	Januar	2019	2	8.0					
							3			5	
Tirsdag		22	Januar	2019	2	4.0					
								4			
Onsdag		23	Januar	2019	2	8.0					
									3	5	
Torsdag		24	Januar	2019	2	1.0					
								1			
Fredag		25	Januar	2019	2	13.0					
							13				
Lørdag		26	Januar	2019	2	4.0					
								2	2		
Søndag		27	Januar	2019	2	2.0					
							2				
Mandag		28	Januar	2019	3	11.0					
							5	3		3	
Tirsdag		29	Januar	2019	3	5.5					
							1	4.5			
Onsdag		30	Januar	2019	3	7.0					
							5			2	
Torsdag		31	Januar	2019	3	6.0					
							1		5		
Totalt i januar						115.0		38.5	19.5	15.5	41.5
Fredag		1	Februar	2019	3	6.0					
									5	1	
Lørdag		2	Februar	2019	3	5.0					
									5		
Søndag		3	Februar	2019	3	5.0					
								3	2		
Mandag		4	Februar	2019	4	4.5					
									3.5	1	
Tirsdag		5	Februar	2019	4	0.5					
										0.5	
Onsdag		6	Februar	2019	4	6.0					
								5		1	
Torsdag		7	Februar	2019	4	7.5					
							5.5	2			
Fredag		8	Februar	2019	4	5.0					
								5			
Lørdag		9	Februar	2019	4	1.0					
								1			
Søndag		10	Februar	2019	4	0.0					
Mandag		11	Februar	2019	5	9.0					
							2	2		5	
Tirsdag		12	Februar	2019	5	0.0					
Onsdag		13	Februar	2019	5	5.0					
										5	
Torsdag		14	Februar	2019	5	7.5					
								7.5			
Fredag		15	Februar	2019	5	4.0					
								3		1	
Lørdag		16	Februar	2019	5	0.0					
Søndag		17	Februar	2019	5	0.0					
Mandag		18	Februar	2019	6	7.5					
								7.5			
Tirsdag		19	Februar	2019	6	1.0					
								1			
Onsdag		20	Februar	2019	6	6.5					
							1	5.5			
Torsdag		21	Februar	2019	6	8.0					
							3	3		2	
Fredag		22	Februar	2019	6	8.0					
							3	5			
Lørdag		23	Februar	2019	6	0.0					
Søndag		24	Februar	2019	6	0.0					
Mandag		25	Februar	2019	7	7.5					
								7	0.5		
Tirsdag		26	Februar	2019	7	0.0					
Onsdag		27	Februar	2019	7	5.5					
							2			3.5	
Torsdag		28	Februar	2019	7	4.5					
								4		0.5	
Totalt i Februar						114.5		16.5	61.5	16	20.5
Fredag		1	Mars	2019	7	9.0					
									9		
Lørdag		2	Mars	2019	7	0.0					
Søndag		3	Mars	2019	7	4.0					
									4		
Mandag		4	Mars	2019	8	4.0					
									4		
Tirsdag		5	Mars	2019	8	7.5					
								7.5			
Onsdag		6	Mars	2019	8	6.5					
									3	3.5	
Torsdag		7	Mars	2019	8	8.5					
							2	2	4	0.5	
Fredag		8	Mars	2019	8	7.5					
							1.5		6		
Lørdag		9	Mars	2019	8	6.5					
							5	1	0.5		

Ming Kit Wong

Ukedag	Dag	Måned	År	Sprint nr	Timer	Kommentar	Dokumentering	Teknisk arbeid	Presentasjon	Administrativt arbeid	
Søndag		10	Mars	2019	8	5.0	[Description] Pugh-matrix, preparing for deliverables	5			
Mandag		11	Mars	2019	8	10.0	Presentation, Mark2		2	8	
Tirsdag		12	Mars	2019	8	10.0	Presentation, Mark2		6	4	
Onsdag		13	Mars	2019	8	3.5	Presentation			3.5	
Torsdag		14	Mars	2019	8	0.0					
Fredag		15	Mars	2019	8	4.0	VMD, Group meeting	2		2	
Lørdag		16	Mars	2019	8	0.0					
Søndag		17	Mars	2019	8	1.0	Moment of Inertia		1		
Mandag		18	Mars	2019	9	7.5	Sprint meeting, planning the progress for the rest of the project, VMD	2.5		5	
Tirsdag		19	Mars	2019	9	0.0					
Onsdag		20	Mars	2019	9	6.5	Meeting at KIC and group work			6.5	
Torsdag		21	Mars	2019	9	2.0	Researching gear		2		
Fredag		22	Mars	2019	9	0.0					
Lørdag		23	Mars	2019	9	2.5	Background for design of Twist capsule	2.5			
Søndag		24	Mars	2019	9	0.0					
Mandag		25	Mars	2019	10	0.0					
Tirsdag		26	Mars	2019	10	4.0	Design of Twist capsule for M3		4		
Onsdag		27	Mars	2019	10	1.0	Contact with Tronrud/Structure, research		1		
Torsdag		28	Mars	2019	10	1.0	research on common 2U structures		1		
Fredag		29	Mars	2019	10	2.0	research on common 2U structures		2		
Lørdag		30	Mars	2019	10	0.0	Mechatronics Exam				
Søndag		31	Mars	2019	10	0.0	Mechatronics Exam				
Totalt i Mars						113.5		75.5	127.5	60.5	79.5
Mandag		1	April	2019	10	0.0	Mechatronics Exam				
Tirsdag		2	April	2019	10	0.0	Mechatronics Exam				
Onsdag		3	April	2019	10	0.0	Mechatronics Exam				
Torsdag		4	April	2019	10	0.0	Mechatronics Exam				
Fredag		5	April	2019	10	0.0	Mechatronics Exam				
Lørdag		6	April	2019	10	0.0					
Søndag		7	April	2019	10	0.0					
Mandag		8	April	2019	11	8.0	Research and design of structure for M3		8		
Tirsdag		9	April	2019	11	8.0	Design of M3		8		
Onsdag		10	April	2019	11	6.0	Meetings with supervisors and consulting with Mek. Lab at KDA		4	2	
Torsdag		11	April	2019	11	8.5	Design of M3, mainly flex shaft and details		8.5		
Fredag		12	April	2019	11	6.0	Meeting with Jose and design of M3		5	1	
Lørdag		13	April	2019	11	2.0	Research		2		
Søndag		14	April	2019	11	2.5	redesign of flex shaft		2.5		
Mandag		15	April	2019	12	7.5	Sprint, group (report) and transparency meetings, redesign of shaft and other CAD		5.5	2	
Tirsdag		16	April	2019	12	0.0	CAD M3				
Onsdag		17	April	2019	12	6.5	Meetings with supervisors and consulting with Mek. Lab at KDA, editing at M3		5	1.5	
Torsdag		18	April	2019	12	7.5	Meetings with team and Jose about the report, CAD		3.5	4	
Fredag		19	April	2019	12	7.5	Research (bearings) and documentation (Validation and verification)	5	2.5		
Lørdag		20	April	2019	12	0.0					
Søndag		21	April	2019	12	4.0	Review of documentations and research about bearing	2	2		
Mandag		22	April	2019		4.0	Research for the chapter State of the art		4		
Tirsdag		23	April	2019		8.5	Sprint meeting, last editing of M3 before sending to Mek Lab for revision, documentation in the report	5	3	0.5	
Onsdag		24	April	2019		5.5	Meeting with DSS, consulting with Mek Lab, redesign		4.5	1	
Torsdag		25	April	2019		7.5	Optimizing structure of M2 for 3D printing at Tronrud Engineering and other redesigns		7.5		
Fredag		26	April	2019		10.0	Expo-meeting, consulting with Richard Thue, redesigns, assembling plastic M3		8	2	
Lørdag		27	April	2019		5.0	2D to Waiken Rapid Manufacturing, new structure to Tronrud Engineering		5		
Søndag		28	April	2019		0.0					
Mandag		29	April	2019		8.0	State of Art - documentation and research	4	4		
Tirsdag		30	April	2019		7.0	Space Night meeting and preparing			7	
Totalt i april						129.5		91.5	220	67.5	93.5
Onsdag		1	Mai	2019		6.0	Space Night rehearsal, making Arduino bracket and walls of sheet metal		4	2	
Torsdag		2	Mai	2019		7.0	Assembling M3, promo of project at Space Night presentation, Expo talk with marianne		5	2	
Fredag		3	Mai	2019		2.5	redesign model for production, documentation	1	1.5		
Lørdag		4	Mai	2019		4.0	Documentation of Background	4			
Søndag		5	Mai	2019		5.0	thermal analysis, bearing research		5		
Mandag		6	Mai	2019		7.5	Mek Lab, thermal analysis, edit screw holes		7.5		
Tirsdag		7	Mai	2019		7.5	thermal analysis, bearing research, report (M2 structure)		7.5		
Onsdag		8	Mai	2019		6.0	Thermal analysis, Meeting with externals, making drawings to Mek Lab for production		4	2	
Torsdag		9	Mai	2019		10.0	Documentation of the structure, redesign, correction of mistakes in design	6	4		
Fredag		10	Mai	2019		6.5	Meeting with Aage about the future, edit drawings for MekLab, design test station		5	1.5	
Lørdag		11	Mai	2019		5.0	Documenting Design Description	5			
Søndag		12	Mai	2019		4.0	Reviewing and producing documents and designing test station	3	1		
Mandag		13	Mai	2019		8.0	Documentation, meeting with Jose, converting old documents to latex, motion simulations	6	1	1	
Tirsdag		14	Mai	2019		10.0	Assembling Mark3 Alu and motion simulation at SW.		10		
Onsdag		15	Mai	2019		7.5	Modifying the sheet metal used as structure, making test station, converting old documents to latex	1.5	6		

Ming Kit Wong

Ukedag	Dag	Måned	År	Sprint nr	Timer	Kommentar	Dokumentering	Teknisk arbeid	Presentasjon	Administrativ arbeid	
Torsdag		16	Mai	2019		10.0	Simulation of motor torque and testing the prototype		10		
Fredag		17	Mai	2019		10.0	Made a mock up of solar arrays for testing, making test station for positiontesting		10		
Lørdag		18	Mai	2019		10.0	Simulation and calculation on inertia, making test station for torque testing		10		
Søndag		19	Mai	2019		8.0	Making the assembly manual, small fixes on the test station	8			
Mandag		20	Mai	2019		8.0	Documentation	8			
Tirsdag		21	Mai	2019		12.0	Documentation - Finance	12			
Onsdag		22	Mai	2019		0.0					
Torsdag		23	Mai	2019		0.0					
Fredag		24	Mai	2019		0.0					
Lørdag		25	Mai	2019							
Søndag		26	Mai	2019							
Mandag		27	Mai	2019							
Tirsdag		28	Mai	2019							
Onsdag		29	Mai	2019							
Torsdag		30	Mai	2019							
Fredag		31	Mai	2019							
Totalt i mai						154.5		54.5	91.5	4	4.5
Lørdag		1	Juni	2019							
Søndag		2	Juni	2019							
Mandag		3	Juni	2019							
Tirsdag		4	Juni	2019							
Onsdag		5	Juni	2019							
Torsdag		6	Juni	2019							
Fredag		7	Juni	2019							

Thomas Mundal

Ukedag	Dag	Måned	År	Sprint nr	Timer	Kommentar	Dokumentering	Teknisk arbeid	Presentasjon	Administrativt arbeid
Tirsdag	1	Januar	2019		0					
Onsdag	2	Januar	2019		0					
Torsdag	3	Januar	2019		0					
Fredag	4	Januar	2019		0					
Lørdag	5	Januar	2019		0					
Søndag	6	Januar	2019		0					
Mandag	7	Januar	2019		1	Planning first meeting				1
Tirsdag	8	Januar	2019		3	Project software tools				3
Onsdag	9	Januar	2019		2.5	Project model				2.5
Torsdag	10	Januar	2019		1.5	Planning project model				1.5
Fredag	11	Januar	2019		7	Discussion about projectmodel, definitions and delegating roles + project software tools				7
Lørdag	12	Januar	2019		0					
Søndag	13	Januar	2019		0					
Mandag	14	Januar	2019	1	8	Done forming project model, not ready for report. Group contract. Procedure for documenting requirements and risk in confluence.	5			3
Tirsdag	15	Januar	2019	1	0					
Onsdag	16	Januar	2019	1	6	Meeting with Eirik, discussed progress in project.				6
Torsdag	17	Januar	2019	1	11	Explored libraries for statistics and plotting. Other administrative work	2	6		3
Fredag	18	Januar	2019	1	6.5	Internal supervisor meeting. Retrospective, referent, shared documents with Jose. Made diagram for software		1		5.5
Lørdag	19	Januar	2019	1	0					
Søndag	20	Januar	2019	1	0					
Mandag	21	Januar	2019	2	7.5	Arduino SADE emulating, software interface with arduino. Meeting agenda for next meeting. Sprint meeting, sprint planning. Retrospective documentation. Other administrative work	2	3		2.5
Tirsdag	22	Januar	2019	2	8.5	External supervisor meeting. Work on first presentation		1	2	5.5
Onsdag	23	Januar	2019	2	7.5	Detailed info about retrospective, research into unit testing, form and send agenda for internal supervisor meeting. Work on presenting tool	1	1.5	3	2
Torsdag	24	Januar	2019	2	7.5	Revised description of project model. Grouped hours in sprints, set up cell-connections between cells in spreadsheet for requirements and test. Meeting with internal supervisor	6.5			1
Fredag	25	Januar	2019	2	0					
Lørdag	26	Januar	2019	2	0					
Søndag	27	Januar	2019	2	1	Helping with google sheets		1		
Mandag	28	Januar	2019	3	7.5	Sprint meeting, presentation, documenting software requirements	5		1.5	1
Tirsdag	29	Januar	2019	3	0	Sick				
Onsdag	30	Januar	2019	3	0	Sick				
Torsdag	31	Januar	2019	3	1	Presentasjon script + slides, Sick				1
Totalt i januar					87		21.5	13.5	7.5	44.5
Fredag	1	Februar	2019	3	2	Meeting with Jose. Slides and script	1		1	
Lørdag	2	Februar	2019	3	5	Pres script and slides + rehearsal		2	3	
Søndag	3	Februar	2019	3	3	Rehearsal for presentation			3	
Mandag	4	Februar	2019	4	6	Presentation 1, working on diagnostic system		4	2	
Tirsdag	5	Februar	2019	4	0					
Onsdag	6	Februar	2019	4	7.5	Sprint planning, requirement analysis, verification definitions	7.5			
Torsdag	7	Februar	2019	4	9.5	Requirement analysis, verification definitions, document structures. Preliminary design software layer	7.5	2		
Fredag	8	Februar	2019	4	8.5	Version definitions SW layer, timeline for alpha version progress, design SW layer serial I/O interface. Started implementation	3	5.5		
Lørdag	9	Februar	2019	4	0					
Søndag	10	Februar	2019	4	0					
Mandag	11	Februar	2019	5	8.5	Sprintmeeting, Retrospektiv, Design SWlayer JSON implementation and generic interface	5.5			3
Tirsdag	12	Februar	2019	5	0					
Onsdag	13	Februar	2019	5	8.5	Finished design for input data, defined unit tests for input data, verified through unit tests. One failed. Meeting with internal supervisor. Updated requirement document for SW layer	1	6.5		1
Torsdag	14	Februar	2019	5	8.5	Verification through Unit Testing for input and output data. Written verification report, finished UML design for generic I/O system at software layer	2	6.5		
Fredag	15	Februar	2019	5	6	Meeting with internal supervisor. Discovered and defined more SW layer requirements. UI design		6		
Lørdag	16	Februar	2019	5	0					
Søndag	17	Februar	2019	5	0					
Mandag	18	Februar	2019	6	11	Finalized version Alpha1, planning communication protocol with HW layer. Rewrite code to support arbitrary datatype from HW layer. Rewrite and rerun tests for verification. Started implementing stream simulator for testing without hw layer connected		11		
Tirsdag	19	Februar	2019	6	0					
Onsdag	20	Februar	2019	6	8	Supervisor meeting, tour of some useful facilities at Kongsberg. Set up unity scene for using hololens in presentation, some testing and minor coding		5	3	
Torsdag	21	Februar	2019	6	3	Stream simulator now work with UI elements + tweaks / sick		3		
Fredag	22	Februar	2019	6	3	General development / sick		3		
Lørdag	23	Februar	2019	6	0					
Søndag	24	Februar	2019	6	0					
Mandag	25	Februar	2019	7	7.5	Sprint planning meeting, code refactor, UML design com settings, review class diagrams, added documentation comments	6.5			1
Tirsdag	26	Februar	2019	7	0					
Onsdag	27	Februar	2019	7	7.5	Supervisor meeting. Work on AR prototype demonstration		6.5		1
Torsdag	28	Februar	2019	7	7.5	Work in communication protocol. Expand stream simulator to support protocol handshake for testing		7.5		
Totalt i februar					120.5		34	68.5	12	6
Fredag	1	Mars	2019	7	8.5	Internal supervisor meeting. Work on implementing communication protocol on SWL		7.5		1
Lørdag	2	Mars	2019	7	0					
Søndag	3	Mars	2019	7	0					

Thomas Mundal

Ukedag	Dag	Måned	År	Sprint nr	Timer	Kommentar	Dokumentering	Teknisk arbeid	Presentasjon	Administrativt arbeid
Mandag	4	Mars	2019	8	7.5	Update project report. Preparing for versioning alpha2 for software layer	6.5	1		
Tirsdag	5	Mars	2019	8	0					
Onsdag	6	Mars	2019	8	6.5	External supervisor meeting. Work on HoloLens presentation		4.5		2
Torsdag	7	Mars	2019	8	7.5	Patched Alpha2 and released Alpha2.a. Chosen LiteDB as database library. WIP Proof of concept database saving of plot data implemented on Alpha3 branch		7.5		
Fredag	8	Mars	2019	8	8.5	Internal supervisor meeting, work on second presentation		6.5		2
Lørdag	9	Mars	2019	8	0					
Søndag	10	Mars	2019	8	0					
Mandag	11	Mars	2019	8	12	Work on presentation 2			12	
Tirsdag	12	Mars	2019	8	5	Work on presentation 2			5	
Onsdag	13	Mars	2019	8	5	Second presentation			5	
Torsdag	14	Mars	2019	8	0					
Fredag	15	Mars	2019	8	7.5	Work on implementing database saving and viewing for Alpha3 + group status meeting and discussion		6.5		1
Lørdag	16	Mars	2019	8	0					
Søndag	17	Mars	2019	8	0					
Mandag	18	Mars	2019	9	8.5	Work on timeline forward, sync team work. Work on socket communication	1.5	4		3
Tirsdag	19	Mars	2019	9	0					
Onsdag	20	Mars	2019	9	6.5	External sup meeting. Started on second progress plan for SWL and beta milestones. Walktrough of SWL code with Jon. Work on mark3 exploded view animation++		4.5		2
Torsdag	21	Mars	2019	9	5	Plot optimizations, changed to OxyPlot		5		
Fredag	22	Mars	2019	9	0	Exam period				
Lørdag	23	Mars	2019	9	0	Exam period				
Søndag	24	Mars	2019	9	0	Exam period				
Mandag	25	Mars	2019	10	0	Exam period				
Tirsdag	26	Mars	2019	10	0	Exam period				
Onsdag	27	Mars	2019	10	0	Exam period				
Torsdag	28	Mars	2019	10	0	Exam period				
Fredag	29	Mars	2019	10	0	Exam period				
Lørdag	30	Mars	2019	10	0	Exam period				
Søndag	31	Mars	2019	10	0	Exam period				
Totalt i mars					88		8	47	22	11
Mandag	1	April	2019	10		Exam period				
Tirsdag	2	April	2019	10		Exam period				
Onsdag	3	April	2019	10		Exam period				
Torsdag	4	April	2019	10		Exam period				
Fredag	5	April	2019	10		Exam period				
Lørdag	6	April	2019	10		Exam period				
Søndag	7	April	2019	10		Exam period				
Mandag	8	April	2019	11	8	Save and load previous COM settings + related tests. Database viewer can display several entries in one plot. Fix for plotmodel update from separate thread by locking data. API docs updates. Design documents development	2	6		
Tirsdag	9	April	2019	11	6.5	Start work on live data view with validation parameters	1	5.5		
Onsdag	10	April	2019	11	8	Work on observable values. Work on deploying hololens presentation	1	7		
Torsdag	11	April	2019	11	6	Finish observable variables with responsive UI	1	6		
Fredag	12	April	2019	11	5.5	Save/load observable variables template		5.5		
Lørdag	13	April	2019	11	0					
Søndag	14	April	2019	11	0					
Mandag	15	April	2019	12	6.5	Sprint meeting, transparency meeting, testing and finalizing of Beta1, Update verification document and requirement documents	2	3.5		1
Tirsdag	16	April	2019	12	7	Work on report and other documentation	7			
Onsdag	17	April	2019	12	7.5	Supervisor meeting + hololens work		6.5		1
Torsdag	18	April	2019	12	6	Work on test system + SADM simulator		6		
Fredag	19	April	2019	12	0					
Lørdag	20	April	2019	12	0					
Søndag	21	April	2019	12	0					
Mandag	22	April	2019	13	7.5	Work on test system UI and execution + simulator		7.5		
Tirsdag	23	April	2019	13	7.5	Work on test system		7.5		
Onsdag	24	April	2019	13	1	Supervisor meeting, validation of software and UI, continue refining test execution		5		1
Torsdag	25	April	2019	13	5	Working on user stories		5		
Fredag	26	April	2019	13	7.5	Supervisor meeting, Some code refactoring. Continue automated testing development		6.5		1
Lørdag	27	April	2019	13	0					
Søndag	28	April	2019	13	0					
Mandag	29	April	2019	14	5	Sprint eval, automated testing		4		1
Tirsdag	30	April	2019	14	7	Automated testing + refactoring, work with external simulation		7		
Totalt i april					101.5		22	135.5	22	16
Onsdag	1	Mai	2019	14	7.5	Refactoring of some code, automated testing. Prepare for space night presentation		6.5	1	
Torsdag	2	Mai	2019	14	7.5	Space night presentation		3.5	1	3
Fredag	3	Mai	2019	14	7.5	Working on user stories		7.5		
Lørdag	4	Mai	2019	14	0					
Søndag	5	Mai	2019	14	0					
Mandag	6	Mai	2019	15	7.5	Documentation, some optimizations for software layer, started user manual for software layer	6.5	1		
Tirsdag	7	Mai	2019	15	7.5	Research thread interrupts. Write SatStat user manual, Rebuild automated test architecture, refactor some code by creating custom UI controls and moving event handlers	2	5.5		
Onsdag	8	Mai	2019	15	0	Job Interview				
Torsdag	9	Mai	2019	15	7.5	Documentation	7.5			
Fredag	10	Mai	2019	15	7.5	Some docs, Refactoring some code, fixed bugs with automated testing	1	6.5		

Thomas Mundal

Ukedag	Dag	Måned	År	Sprint nr	Timer	Kommentar	Dokumentering	Teknisk arbeid	Presentasjon	Administrativ arbeid
Lørdag	11	Mai	2019	15	0					
Søndag	12	Mai	2019	15	4	Documentation	4			
Mandag	13	Mai	2019		7.5	Internal supervisor meeting. Documentation	7			0.5
Tirsdag	14	Mai	2019		7.5	Docs	7.5			
Onsdag	15	Mai	2019		8	Documentation, finished satstat user manual, finalized beta2 last stories, fixed small bugs. Latex technical support. Defined some risks	7	1		
Torsdag	16	Mai	2019		7	Documenting	7			
Fredag	17	Mai	2019		7.5	Documentation, abstract, proof reading. Release SatStat v1	6.5	1		
Lørdag	18	Mai	2019		0					
Søndag	19	Mai	2019		0					
Mandag	20	Mai	2019		12	Documentation, proof reading, attachement management	12			
Tirsdag	21	Mai	2019		12	Documentation, report, attachement management, proof reading	12			
Onsdag	22	Mai	2019		0					
Torsdag	23	Mai	2019		0					
Fredag	24	Mai	2019		0					
Lørdag	25	Mai	2019		0					
Søndag	26	Mai	2019		0					
Mandag	27	Mai	2019		0					
Tirsdag	28	Mai	2019		0					
Onsdag	29	Mai	2019		0					
Torsdag	30	Mai	2019		0					
Fredag	31	Mai	2019		0					
Totalt i mai					118		80	32.5	2	3.5
Lørdag	1	Juni	2019							
Søndag	2	Juni	2019							
Mandag	3	Juni	2019							
Tirsdag	4	Juni	2019							
Onsdag	5	Juni	2019							
Torsdag	6	Juni	2019							
Fredag	7	Juni	2019							

Rita Hogstad

Ukedag	Dag	Måned	År	Sprint nr	Timer	Kommentar	Dokumentering	Teknisk arbeid	Presentasjon	Administrativt arbeid	
Tirsdag		1	Januar	2019		0					
Onsdag		2	Januar	2019		0					
Torsdag		3	Januar	2019		0					
Fredag		4	Januar	2019		0					
Lørdag		5	Januar	2019		0					
Søndag		6	Januar	2019		0					
Mandag		7	Januar	2019		2				2	
Tirsdag		8	Januar	2019		0					
Onsdag		9	Januar	2019		2.5				2.5	
Torsdag		10	Januar	2019		3.5				3.5	
Fredag		11	Januar	2019		5				5	
Lørdag		12	Januar	2019		0					
Søndag		13	Januar	2019		0					
Mandag		14	Januar	2019	1	7				7	
Tirsdag		15	Januar	2019	1	0					
Onsdag		16	Januar	2019	1	6				6	
Torsdag		17	Januar	2019	1	7				4	
Fredag		18	Januar	2019	1	6.5				5	
Lørdag		19	Januar	2019	1	0					
Søndag		20	Januar	2019	1	0					
Mandag		21	Januar	2019	2	8				4	
Tirsdag		22	Januar	2019	2	0					
Onsdag		23	Januar	2019	2	8				3	
Torsdag		24	Januar	2019	2	7				5	
Fredag		25	Januar	2019	2	7				7	
Lørdag		26	Januar	2019	2	0					
Søndag		27	Januar	2019	2	0					
Mandag		28	Januar	2019	3	12					
Tirsdag		29	Januar	2019	3	6.5					
Onsdag		30	Januar	2019	3	7					
Torsdag		31	Januar	2019	3	7.5				7.5	
Totalt i januar						102.5		28	11	9.5	54
Fredag		1	Februar	2019	3	6				6	
Lørdag		2	Februar	2019	3	5				5	
Søndag		3	Februar	2019	3	2				2	
Mandag		4	Februar	2019	4	3				3	
Tirsdag		5	Februar	2019	4	0					
Onsdag		6	Februar	2019	4	6				6	
Torsdag		7	Februar	2019	4	8					
Fredag		8	Februar	2019	4	4				1	
Lørdag		9	Februar	2019	4	0					
Søndag		10	Februar	2019	4	0					
Mandag		11	Februar	2019	5	7					
Tirsdag		12	Februar	2019	5	0					
Onsdag		13	Februar	2019	5	7				5	
Torsdag		14	Februar	2019	5	6.5				6.5	
Fredag		15	Februar	2019	5	0					
Lørdag		16	Februar	2019	5	0					
Søndag		17	Februar	2019	5	0					
Mandag		18	Februar	2019	6	0					
Tirsdag		19	Februar	2019	6	0					
Onsdag		20	Februar	2019	6	0					
Torsdag		21	Februar	2019	6	7				7	
Fredag		22	Februar	2019	6	5				3	
Lørdag		23	Februar	2019	6	2				2	
Søndag		24	Februar	2019	6	0					
Mandag		25	Februar	2019	7	7				7	
Tirsdag		26	Februar	2019	7	0					
Onsdag		27	Februar	2019	7	6				4	
Torsdag		28	Februar	2019	7	7				7	
Totalt i februar						88.5		16	50.5	16	6
Fredag		1	Mars	2019	7	7				6	
Lørdag		2	Mars	2019	7	0					
Søndag		3	Mars	2019	7	0					
Mandag		4	Mars	2019	8	3				3	
Tirsdag		5	Mars	2019	8	7				7	
Onsdag		6	Mars	2019	8	2				2	
Torsdag		7	Mars	2019	8	7				7	
Fredag		8	Mars	2019	8	7				7	
Lørdag		9	Mars	2019	8	6					
Søndag		10	Mars	2019	8	4				1	
Mandag		11	Mars	2019	8	13				13	
Tirsdag		12	Mars	2019	8	14				14	
Onsdag		13	Mars	2019	8	3				3	
Torsdag		14	Mars	2019	8	0					
Fredag		15	Mars	2019	8	7				3	
Lørdag		16	Mars	2019	8	0					
Søndag		17	Mars	2019	8	0					
Mandag		18	Mars	2019	9	10				4	
Tirsdag		19	Mars	2019	9	0					
Onsdag		20	Mars	2019	9	7				7	

Rita Hogstad

Ukedag	Dag	Måned	År	Sprint nr	Timer	Kommentar	Dokumentering	Teknisk arbeid	Presentasjon	Administrativt arbeid	
Torsdag		21	Mars	2019	9	0					
Fredag		22	Mars	2019	9	0					
Lørdag		23	Mars	2019	9	0					
Søndag		24	Mars	2019	9	0					
Mandag		25	Mars	2019	10	2	Discussion in group			2	
Tirsdag		26	Mars	2019	10	0	Mechatronics exam prep				
Onsdag		27	Mars	2019	10	0	Mechatronics exam prep				
Torsdag		28	Mars	2019	10	0	Mechatronics exam prep				
Fredag		29	Mars	2019	10	0	Mechatronics exam prep				
Lørdag		30	Mars	2019	10	0	Mechatronics exam prep				
Søndag		31	Mars	2019	10	0	Mechatronics exam prep				
Totalt i mars						99		14	32	45	8
Mandag		1	April	2019	10	0	Mechatronics exam prep				
Tirsdag		2	April	2019	10	0	Mechatronics exam prep				
Onsdag		3	April	2019	10	0	Mechatronics exam prep				
Torsdag		4	April	2019	10	0	Mechatronics exam prep				
Fredag		5	April	2019	10	0	Mechatronics exam				
Lørdag		6	April	2019	10	0					
Søndag		7	April	2019	10	0					
Mandag		8	April	2019	11	7	Made brochure for EXPO, Mark3 Design		7		
Tirsdag		9	April	2019	11	8.5	SolidWorks - Mark3 parts		8.5		
Onsdag		10	April	2019	11	7	SolidWorks - Mark3 parts		7		
Torsdag		11	April	2019	11	10.5	SolidWorks - Mark3 parts		10.5		
Fredag		12	April	2019	11	5.5	SolidWorks - Mark3 parts		5.5		
Lørdag		13	April	2019	11	0					
Søndag		14	April	2019	11	0					
Mandag		15	April	2019	12	5.5	Solidworks		5.5		
Tirsdag		16	April	2019	12	6.5	Solidworks		6.5		
Onsdag		17	April	2019	12	6.5	Solidworks		6.5		
Torsdag		18	April	2019	12	0					
Fredag		19	April	2019	12	0					
Lørdag		20	April	2019	12	0					
Søndag		21	April	2019	12	0					
Mandag		22	April	2019	13	5	Written about Mark3 in the report, research on gears	3	2		
Tirsdag		23	April	2019	13	8	Written "Gantt", "Working time" in the report, modified two versions of Mark3 in Solidworks	5	3		
Onsdag		24	April	2019	13	7	Visited Mech lab, documentation	6	1		
Torsdag		25	April	2019	13	8	Office at home; Written documentation on M3, solidworks analysis	7	1		
Fredag		26	April	2019	13	12	Optimized SolidWorks parts, 2D drawings, FEM simulation, research on bearings		12		
Lørdag		27	April	2019	13	5	SolidWorks . Stepper motor adapter, FEM		5		
Søndag		28	April	2019	13	0					
Mandag		29	April	2019	14	8	Updated Gantt diagram, Material research, Simulations		8		
Tirsdag		30	April	2019	14	8	Preparations for Space Night		8		
Totalt i april						118		35	129	45	8
Onsdag		1	Mai	2019	14	8	Space Night rehearsals, Prezi and documentation of gears, materials	4	4		
Torsdag		2	Mai	2019	14	8	Space Night, Mark3 Assembly		5	3	
Fredag		3	Mai	2019	14	3	Gear investigation and research, bit of documentation	1	2		
Lørdag		4	Mai	2019	14	1	Research on FEM		1		
Søndag		5	Mai	2019	14	4	Reading about thermal expansion on materials, Vibration testing, materials		4		
Mandag		6	Mai	2019	15	9	Documentation of materials, FEM, Gantt	9			
Tirsdag		7	Mai	2019	15	8.5	FEM documentation, Mark3 Assembly	4.5	4		
Onsdag		8	Mai	2019	15	10	Supervisor meeting, FEM simulations, Documentation on FEM and materials	6	4		
Torsdag		9	Mai	2019	15	10	Written: Materials, User Manual, About USN and Space, Project Plan	10			
Fredag		10	Mai	2019	15	6	Meeting with external sensor, written about project challenges, Work(Bistro)	6			
Lørdag		11	Mai	2019	15	5	Design Description, User manual, Flexh and Main Shafts. Work(Bistro)	5			
Søndag		12	Mai	2019	15	10	User manual, Flex Shaft and Main Shafts	10			
Mandag		13	Mai	2019		7	Design Description, placed attachments into report, translated timesheet	7			
Tirsdag		14	Mai	2019		10	Written "Tools" in the report, Started making A0 poster, translated all meeting notes in Confluence to english	10			
Onsdag		15	Mai	2019		7	Written in the Risk table, Made sure that we include all attachments in the report, translated documents	7			
Torsdag		16	Mai	2019		6	Updated 2D Drawings in Design Description, Updated item numbers in DD, Started on writing about bearings in report, proofreading	6			
Fredag		17	Mai	2019		0					
Lørdag		18	Mai	2019		3	Proofreading of report	3			
Søndag		19	Mai	2019		9	Updated VMD and Confluence requirements, Design Description and User Manual, Proofreading	9			
Mandag		20	Mai	2019		9	Added to and proofreading of report	9			
Tirsdag		21	Mai	2019		12	Mark3 assembly, attachments, finishing and printing report	12			
Onsdag		22	Mai	2019		0					
Torsdag		23	Mai	2019		0					
Fredag		24	Mai	2019		0					
Lørdag		25	Mai	2019		0					
Søndag		26	Mai	2019		0					
Mandag		27	Mai	2019		0					
Tirsdag		28	Mai	2019		0					
Onsdag		29	Mai	2019		0					
Torsdag		30	Mai	2019		0					
Fredag		31	Mai	2019		0					
Totalt i mai						145.5		118.5	24	3	0
Lørdag		1	Juni	2019							

Rita Hogstad

Ukedag	Dag	Måned	År	Sprint nr	Timer	Kommentar	Dokumentering	Teknisk arbeid	Presentasjon	Administrativ arbeid
Søndag		2 Juni	2019							
Mandag		3 Juni	2019							
Tirsdag		4 Juni	2019							
Onsdag		5 Juni	2019							
Torsdag		6 Juni	2019							
Fredag		7 Juni	2019							

Erica Fegri

Ukedag	Dag	Måned	År	Sprint nr	Timer	Kommentar	Dokumentering	Teknisk arbeid	Presentasjon	Administrativt arbeid	
Tirsdag		1	Januar	2019		0					
Onsdag		2	Januar	2019		0					
Torsdag		3	Januar	2019		0					
Fredag		4	Januar	2019		0					
Lørdag		5	Januar	2019		0					
Søndag		6	Januar	2019		0					
Mandag		7	Januar	2019		2				2	
Tirsdag		8	Januar	2019		0					
Onsdag		9	Januar	2019		2.5				2.5	
Torsdag		10	Januar	2019		3.5				3.5	
Fredag		11	Januar	2019		5				5	
Lørdag		12	Januar	2019		0					
Søndag		13	Januar	2019		0					
Mandag		14	Januar	2019	1	8				8	
Tirsdag		15	Januar	2019	1	0					
Onsdag		16	Januar	2019	1	9	KIC, møte med ekstern veileder, 1. presentasjo	2		1.5	5.5
Torsdag		17	Januar	2019	1	8.5	Spekk, kravbehandling	4			4.5
Fredag		18	Januar	2019	1	6.5					6.5
Lørdag		19	Januar	2019	1	0					
Søndag		20	Januar	2019	1	1					1
Mandag		21	Januar	2019	2	7.5					7.5
Tirsdag		22	Januar	2019	2	0					
Onsdag		23	Januar	2019	2	8.5	Møte med esktern veileder, mm.				8.5
Torsdag		24	Januar	2019	2	7	Req.research, system mapping, presenting pro	1		1	4
Fredag		25	Januar	2019	2	7.5		7.5			
Lørdag		26	Januar	2019	2	0					
Søndag		27	Januar	2019	2	0					
Mandag		28	Januar	2019	3	8		4			4
Tirsdag		29	Januar	2019	3	1.5		1.5			
Onsdag		30	Januar	2019	3	6.5			2	1	3.5
Torsdag		31	Januar	2019	3	7.5				6.5	1
Totalt i januar						100		20	3	10	67
Fredag		1	Februar	2019	3	7.5				7.5	
Lørdag		2	Februar	2019	3	5				5	
Søndag		3	Februar	2019	3	5				5	
Mandag		4	Februar	2019	4	3				3	
Tirsdag		5	Februar	2019	4	0					
Onsdag		6	Februar	2019	4	7.5		3	3		1.5
Torsdag		7	Februar	2019	4	8.5		8.5			
Fredag		8	Februar	2019	4	4		4			
Lørdag		9	Februar	2019	4	6		6			
Søndag		10	Februar	2019	4	0					
Mandag		11	Februar	2019	5	8.5					8.5
Tirsdag		12	Februar	2019	5	0					
Onsdag		13	Februar	2019	5	7.5		7.5			
Torsdag		14	Februar	2019	5	7		4			3
Fredag		15	Februar	2019	5	0					
Lørdag		16	Februar	2019	5	0					
Søndag		17	Februar	2019	5	0					
Mandag		18	Februar	2019	6	8.5	Skaptegninger, fremdriftsplan, scrummøe	3	3.5		2
Tirsdag		19	Februar	2019	6	0					
Onsdag		20	Februar	2019	6	8.5	Skaptegninger	4.5	4		
Torsdag		21	Februar	2019	6	8	Diskusjon, research motor, flexdesign, tegning	4	4		
Fredag		22	Februar	2019	6	7.5					7.5
Lørdag		23	Februar	2019	6	0					
Søndag		24	Februar	2019	6	0					
Mandag		25	Februar	2019	7	7.5			7.5		
Tirsdag		26	Februar	2019	7	0					
Onsdag		27	Februar	2019	7	7.5			7.5		
Torsdag		28	Februar	2019	7	1	Sykehus				1
Totalt i februar						118		44.5	29.5	20.5	23.5
Fredag		1	Mars	2019	7	7.5			7.5		
Lørdag		2	Mars	2019	7	0					
Søndag		3	Mars	2019	7	0					
Mandag		4	Mars	2019	8	1			1		
Tirsdag		5	Mars	2019	8	9			9		
Onsdag		6	Mars	2019	8	8			8		
Torsdag		7	Mars	2019	8	9	Føre inn reviderte krav/review med Eirik, Motor	5.5		2.5	1
Fredag		8	Mars	2019	8	7.5		5.5		2	
Lørdag		9	Mars	2019	8	0					
Søndag		10	Mars	2019	8	6	Forberede 2. presentasjon	2		4	
Mandag		11	Mars	2019	8	13	Forberede 2. presentasjon			13	
Tirsdag		12	Mars	2019	8	12	Forberede 2. presentasjon			12	

Erica Fegri

Ukedag	Dag	Måned	År	Sprint nr	Timer	Kommentar	Dokumentering	Teknisk arbeid	Presentasjon	Administrativt arbeid
Onsdag	13	Mars	2019	8	3.5	2. presentation			3.5	
Torsdag	14	Mars	2019	8	0					
Fredag	15	Mars	2019	8	7.5	Retrospektiv,		4		3.5
Lørdag	16	Mars	2019	8	0					
Søndag	17	Mars	2019	8	0					
Mandag	18	Mars	2019	9	7.5					7.5
Tirsdag	19	Mars	2019	9	0					
Onsdag	20	Mars	2019	9	7.5	Møte med Lars Helge, utstyr og design, prisforespørsel		7.5		
Torsdag	21	Mars	2019	9	3	Finne varer og leverandører		3		
Fredag	22	Mars	2019	9	7.5	Flexdesign, innhente tilbud		4		3.5
Lørdag	23	Mars	2019	9	0					
Søndag	24	Mars	2019	9	0					
Mandag	25	Mars	2019	10	6.5	Mailkorrespondanse, diskusjon, Jira				6.5
Tirsdag	26	Mars	2019	10	7.5	Design connector, bestille varer		7.5		
Onsdag	27	Mars	2019	10	7.5	Møte med ekstern veileder, connectordesign		3		4.5
Torsdag	28	Mars	2019	10	6	Flex, connectordesign, research		4		2
Fredag	29	Mars	2019	10	0					
Lørdag	30	Mars	2019	10	0					
Søndag	31	Mars	2019	10	0					
Totalt i mars					137		13	58.5	37	28.5
Mandag	1	April	2019	10	3.5	Posisjon sensor research		3.5		
Tirsdag	2	April	2019	10	0					
Onsdag	3	April	2019	10	0					
Torsdag	4	April	2019	10	0	Eksamenslesing				
Fredag	5	April	2019	10	0	Eksamen				
Lørdag	6	April	2019	10	0					
Søndag	7	April	2019	10	0					
Mandag	8	April	2019	11	0	Borte på jobb...				
Tirsdag	9	April	2019	11	7.5	Hente varer, samarbeid om connector		4.5		3
Onsdag	10	April	2019	11	8	Sensor, flexdesign, finne leverandør, sende forespørsler, mechla		4		4
Torsdag	11	April	2019	11	7.5	Sensor, flexdesign, sende forespørsler, admin	2	3.5		2
Fredag	12	April	2019	11	7	Mailkorrespondanse, bestilling av varer,		4		3
Lørdag	13	April	2019	11	0					
Søndag	14	April	2019	11	0					
Mandag	15	April	2019	12	7.5	Møte og diskusjon, skriftelig, sensor research	1	3		3.5
Tirsdag	16	April	2019	12	3	Mailkorrespondanse, dokumentasjon	3			
Onsdag	17	April	2019	12	7.5	Møte, dokumentasjon, verifikasjon Meklab	2	3		2.5
Torsdag	18	April	2019	12	6.5	Møte, dokumentasjon	3.5	1		2
Fredag	19	April	2019	12	0					
Lørdag	20	April	2019	12	2		2			
Søndag	21	April	2019	12	0					
Mandag	22	April	2019		0					
Tirsdag	23	April	2019		7.5		1	3		3.5
Onsdag	24	April	2019		7.5	Veiledermøte, 3D-printing, dokumentasjon	2	2		3.5
Torsdag	25	April	2019		2.5	3D-printing		2.5		
Fredag	26	April	2019		11	Veiledermøte, Møte med Karoline, plan B flex	5	1.5		4.5
Lørdag	27	April	2019		0					
Søndag	28	April	2019		4		2	2		
Mandag	29	April	2019		7.5	Flex plan B, Macaos software,		7.5		
Tirsdag	30	April	2019		7.5	Flex plan B fra Danmark og C, steppermotor	1.5	4	2	
Totalt i april					107.5		38	107.5	39	60
Onsdag	1	Mai	2019		6	Presentasjon Spacenight, Elprint mail (flex)		1	5	
Torsdag	2	Mai	2019		7.5	Spacenight			7.5	
Fredag	3	Mai	2019		7	Tegninger, Solidworks Electrical	4			3
Lørdag	4	Mai	2019		0					
Søndag	5	Mai	2019		0					
Mandag	6	Mai	2019		8.5	Flexprint bestilling, oppkobling og test motor		8.5		
Tirsdag	7	Mai	2019		9.5	Motoroppkobling og programmering		9.5		
Onsdag	8	Mai	2019		11	Møte med ekstern veileder, motortest dokumen	5	6		
Torsdag	9	Mai	2019		12.5	Dokumentasjon - requirement, sensortest	11.5	1		
Fredag	10	Mai	2019		14	Dokumentasjon: Power mark 2, MatLab motor	13	1		
Lørdag	11	Mai	2019		12	KIC: motorparametre, motordokumentasjon,	10	2		
Søndag	12	Mai	2019		11	Dokumentasjon; sensor, PT-schematic	11			
Mandag	13	Mai	2019		10	Designe styrestrømskrets, koble skap		10		
Tirsdag	14	Mai	2019		13	KIC/Meklab: Teknisk arbeid SADM, motordoku	8	5		
Onsdag	15	Mai	2019		10	Deratingdokumentasjon	10			
Torsdag	16	Mai	2019		7	Connectors, ground og sensor		7		
Fredag	17	Mai	2019		9	Motorcalculations, drawings update	9			
Lørdag	18	Mai	2019		3	drawings of cabinet update	3			
Søndag	19	Mai	2019		10	Update Req-sheet + VMD. Motor calculations	10			
Mandag	20	Mai	2019		12.5	Groupmeeting, bonding test/documentation	6.5	3		3
Tirsdag	21	Mai	2019		13	Motor calculations, documentation correction, n	12	1		
Onsdag	22	Mai	2019		0					

Erica Fegri

Ukedag	Dag	Måned	År	Sprint nr	Timer	Kommentar	Dokumentering	Teknisk arbeid	Presentasjon	Administrativt arbeid
Torsdag	23	Mai	2019		0					
Fredag	24	Mai	2019		0					
Lørdag	25	Mai	2019		0					
Søndag	26	Mai	2019		0					
Mandag	27	Mai	2019		0					
Tirsdag	28	Mai	2019		0					
Onsdag	29	Mai	2019		0					
Torsdag	30	Mai	2019		0					
Fredag	31	Mai	2019		0					
Totalt i mai					186.5		113	55	12.5	6
Lørdag	1	Juni	2019							
Søndag	2	Juni	2019							
Mandag	3	Juni	2019							
Tirsdag	4	Juni	2019							
Onsdag	5	Juni	2019							
Torsdag	6	Juni	2019							
Fredag	7	Juni	2019							

Håvar Østrem

Ukedag	Dag	Måned	År	Sprint nr	Timer	Kommentar	Dokumentering	Teknisk arbeid	Presentasjon	Administrativt arbeid	
Tirsdag		1	Januar	2019		0					
Onsdag		2	Januar	2019		0					
Torsdag		3	Januar	2019		0					
Fredag		4	Januar	2019		0					
Lørdag		5	Januar	2019		0					
Søndag		6	Januar	2019		0					
Mandag		7	Januar	2019		1	Gruppemøte			1	
Tirsdag		8	Januar	2019		0					
Onsdag		9	Januar	2019		2.5	Prosessmodell			2.5	
Torsdag		10	Januar	2019		3.5	Prosessmodell			3.5	
Fredag		11	Januar	2019		5	Prosessmodell, Rollefordeling			5	
Lørdag		12	Januar	2019		0					
Søndag		13	Januar	2019		0					
Mandag		14	Januar	2019	1	8	Testing av funksjoner i Jira, generell diskusjon og bestemmelser, Satt meg inn i rollen som Risikoansvarlig med research om risikohåndtering	6		2	
Tirsdag		15	Januar	2019	1	0					
Onsdag		16	Januar	2019	1	6	1. Veiledermøte, gruppemøte med presentasjonsplanlegging		2	4	
Torsdag		17	Januar	2019	1	8	Risk management plan, administrativt	6		2	
Fredag		18	Januar	2019	1	6	Intært veiledermøte, retrospekt, administrativt			6	
Lørdag		19	Januar	2019	1	0					
Søndag		20	Januar	2019	1	0					
Mandag		21	Januar	2019	2	10.5	SWOT-analyse, logo design, administrativt			10.5	
Tirsdag		22	Januar	2019	2	1.5	logo design			1.5	
Onsdag		23	Januar	2019	2	4.5	veiledermøte, presentasjon		1.5	3	
Torsdag		24	Januar	2019	2	11.5	Project risk identification, blockdiagram, Presentasjonsarbeid, processmodell review	9.5	0.5	0.5	
Fredag		25	Januar	2019	2	5	int.veil.møte, Project risk identification, presentasjonsarbeid, annet administrativt	3		0.5	
Lørdag		26	Januar	2019	2	0					
Søndag		27	Januar	2019	2	0					
Mandag		28	Januar	2019	3	7.5	logodesign, dokumentering	4		3.5	
Tirsdag		29	Januar	2019	3	0					
Onsdag		30	Januar	2019	3	10.5	Veiledermøte, dokumentering, møtereferat	9		1.5	
Torsdag		31	Januar	2019	3	9			9		
Totalt i januar						100		37.5	0.5	13.5	48.5
Fredag		1	Februar	2019	3	10	presentasjonsforberedelser, prototyping		2	8	
Lørdag		2	Februar	2019	3	5	presentasjonsforberedelser			5	
Søndag		3	Februar	2019	3	8.5	presentasjonsforberedelser			8.5	
Mandag		4	Februar	2019	4	2	Første presentasjon			2	
Tirsdag		5	Februar	2019	4	0					
Onsdag		6	Februar	2019	4	5	requirements	5			
Torsdag		7	Februar	2019	4	7.5	requirements	7.5			
Fredag		8	Februar	2019	4	3.5	veiledermøte, requirements, diskusjon	2.5		1	
Lørdag		9	Februar	2019	4	0					
Søndag		10	Februar	2019	4	0					
Mandag		11	Februar	2019	5	7			5	2	
Tirsdag		12	Februar	2019	5	0					
Onsdag		13	Februar	2019	5	8	veilmøte, flex-research		6	2	
Torsdag		14	Februar	2019	5	0.5	flex-research, electro framdriftsplan			0.5	
Fredag		15	Februar	2019	5	6.5			6.5		
Lørdag		16	Februar	2019	5	0					
Søndag		17	Februar	2019	5	0					
Mandag		18	Februar	2019	6	7.5	scrummøte, processplan elektro, flex-research	1.5	4	2	
Tirsdag		19	Februar	2019	6	0					
Onsdag		20	Februar	2019	6	4	Omvisning på KDA			4	
Torsdag		21	Februar	2019	6	5.5	conceptual flex-design, interfaces	1	4.5		
Fredag		22	Februar	2019	6	1.5	flex		1.5		
Lørdag		23	Februar	2019	6	0					
Søndag		24	Februar	2019	6	0					
Mandag		25	Februar	2019	7	10	flex	3	7		
Tirsdag		26	Februar	2019	7	1		1			
Onsdag		27	Februar	2019	7	6	flexpertrådgivning, veiledemøte, flex	1.5	2.5	2	
Torsdag		28	Februar	2019	7	9.5	flex	3.5	6		
Totalt i februar						108.5		26.5	45	23.5	13.5
Fredag		1	Mars	2019	7	8.5	flexculator, meeting with Jose, flex table	3	4.5	1	
Lørdag		2	Mars	2019	7	0					
Søndag		3	Mars	2019	7	0					
Mandag		4	Mars	2019	8	6	flex design	1.5	4	0.5	
Tirsdag		5	Mars	2019	8	9	flex design, flex dokumentasjon, risk dokumentasjon	5	4		
Onsdag		6	Mars	2019	8	11	møte med Eirik, og Johan, flex scrutinizing, presentasjons forberedelser, rapport	5.5	1	1.5	
Torsdag		7	Mars	2019	8	7.5	rapportskrivning, presentasjon prescript	5		2.5	
Fredag		8	Mars	2019	8	6	meeting with internal supervisor, presentation, documentation	3		3	
Lørdag		9	Mars	2019	8	4	script			4	
Søndag		10	Mars	2019	8	9.25	RMP2, flex dokumentasjon.	9.25			
Mandag		11	Mars	2019	8	12.5	Dokumentering til pres, presøving	2		10.5	

Håvar Østrem

Ukedag	Dag	Måned	År	Sprint nr	Timer	Kommentar	Dokumentering	Teknisk arbeid	Presentasjon	Administrativt arbeid	
Tirsdag		12	Mars	2019	8	10.5	presøving			10.5	
Onsdag		13	Mars	2019	8	3.5	presentasjon 2 og utspørring			3.5	
Torsdag		14	Mars	2019	8	0					
Fredag		15	Mars	2019	8	3.5	Retrospektiv og planlegging			3.5	
Lørdag		16	Mars	2019	8	0					
Søndag		17	Mars	2019	8	0					
Mandag		18	Mars	2019	9	7.5	Sprintplanleggingsmøte			7.5	
Tirsdag		19	Mars	2019	9	0					
Onsdag		20	Mars	2019	9	6	møte med Eirik og Lars Helge, flex design		3	3	
Torsdag		21	Mars	2019	9	0					
Fredag		22	Mars	2019	9	0					
Lørdag		23	Mars	2019	9	0					
Søndag		24	Mars	2019	9	0					
Mandag		25	Mars	2019	10	0					
Tirsdag		26	Mars	2019	10	5	Flexdesign	1	4		
Onsdag		27	Mars	2019	10	0					
Torsdag		28	Mars	2019	10	0					
Fredag		29	Mars	2019	10	0					
Lørdag		30	Mars	2019	10	0					
Søndag		31	Mars	2019	10	0					
Totalt i mars						109.75		35.25	20.5	35.5	18.5
Mandag		1	April	2019	10	0					
Tirsdag		2	April	2019	10	0					
Onsdag		3	April	2019	10	0					
Torsdag		4	April	2019	10	0					
Fredag		5	April	2019	10	0					
Lørdag		6	April	2019	10	0					
Søndag		7	April	2019	10	0					
Mandag		8	April	2019	11	6	Risk Table	6			
Tirsdag		9	April	2019	11	6.5	Kort møte, Risk Table, Sendt Flex design og parametre til KS&S, Sensor	3	2.5	1	
Onsdag		10	April	2019	11	4	Møte med Lars Helge, twist capsule		2	2	
Torsdag		11	April	2019	11	6	Flex_gerber_filer, posisjonssensor, TS-interfaces		6		
Fredag		12	April	2019	11	5	Dokumentasjon til Elmatica	2	3		
Lørdag		13	April	2019	11	0					
Søndag		14	April	2019	11	0					
Mandag		15	April	2019	12	3.5	transparicy meeting, sprint meeting, Twist capsule connectors		1.5	2	
Tirsdag		16	April	2019	12	3	risk description, risk sheet	3			
Onsdag		17	April	2019	12	7	møte med Eirik og Lars Helge, elektroplanlegging, dokumentasjon	2	2	3	
Torsdag		18	April	2019	12	0					
Fredag		19	April	2019	12	0					
Lørdag		20	April	2019	12	0					
Søndag		21	April	2019	12	0					
Mandag		22	April	2019		3	Dokumentasjon og referanser	3			
Tirsdag		23	April	2019		3	Sprintmøte dokumentasjon og referanser	2.5		0.5	
Onsdag		24	April	2019		3	veiledermøte, dokumentasjon	2		1	
Torsdag		25	April	2019		0					
Fredag		26	April	2019		5	Veiledermøte, dokumentasjon, alternativ til flex	3.5	1	0.5	
Lørdag		27	April	2019		0					
Søndag		28	April	2019		5	alternativt til flex	2	3		
Mandag		29	April	2019		7.5	sprint slutt, alternativt til flex, flexbestillingsarbeid	1	6	0.5	
Tirsdag		30	April	2019		10.5	Space Night-møte, flexbestillingsarbeid, logodokumentasjon	9	0.5	1	
Totalt i april						78		39	27.5	1	10.5
Onsdag		1	Mai	2019		6.5	logodokumentasjon, flexdokumentasjon, elektroteknisk, presentasjonskritikk	4.5	1	1	
Torsdag		2	Mai	2019		7.5	Power test bench	1	6.5		
Fredag		3	Mai	2019		4	logodokumentasjon	4			
Lørdag		4	Mai	2019		1	riskdokumentasjon	1			
Søndag		5	Mai	2019		5	logodokumentasjon, flexdokumentasjon, riskdokumentasjon	5			
Mandag		6	Mai	2019		4	reaktivereing av dokumenter, dokumentasjonsplanlegging og referanseinnsamling	4			
Tirsdag		7	Mai	2019		0					
Onsdag		8	Mai	2019		4	Møte med Eirik, LH og Karten, flex dokumentasjon	2		2	
Torsdag		9	Mai	2019		8.25	flexdokumentasjon, logodokumentasjon	8.25			
Fredag		10	Mai	2019		10.5	flexdokumentasjon, logodokumentasjon	10.5			
Lørdag		11	Mai	2019		7	flexdokumentasjon	7			
Søndag		12	Mai	2019		7	flexdokumentasjon	7			
Mandag		13	Mai	2019		9.5	flexdokumentasjon	9.5			
Tirsdag		14	Mai	2019		10	gif til Expo/P3, Bilder til Expo/P3, flexdokumentasjon	8		2	
Onsdag		15	Mai	2019		9	Design Description, solder prep, flexchematics, leiting etter kabelsko og krympestrømer for bestilling, flexdokumentasjon, risk dokumentasjon	4	5		
Torsdag		16	Mai	2019		6	produksjon av mekanisk flexanalog, testing av flexanalog, flexdokumentasjon, design description proofreading	2	4		
Fredag		17	Mai	2019		10.5	Flexdokumentasjon, testing av flexanalog, design description, diverse dokumentasjon	7.5	3		

Håvar Østrem

Ukedag	Dag	Måned	År	Sprint nr	Timer	Kommentar	Dokumentering	Teknisk arbeid	Presentasjon	Administrativ arbeid
Lørdag		18	Mai	2019		4 Dokumentering	4			
Søndag		19	Mai	2019		8 Flex dokumentasjon	8			
Mandag		20	Mai	2019		12.25 Dokumentasjon, lodding D-sub, lodding flex.	4	7.5		0.75
Tirsdag		21	Mai	2019		15.25 Dokumentasjon, poster	15		0.25	
Onsdag		22	Mai	2019		0				
Torsdag		23	Mai	2019		0				
Fredag		24	Mai	2019		0				
Lørdag		25	Mai	2019		0				
Søndag		26	Mai	2019		0				
Mandag		27	Mai	2019		0				
Tirsdag		28	Mai	2019		0				
Onsdag		29	Mai	2019		0				
Torsdag		30	Mai	2019		0				
Fredag		31	Mai	2019		0				
Totalt i mai						149.25	116.25	27	3.25	2.75
Lørdag		1	Juni	2019						
Søndag		2	Juni	2019						
Mandag		3	Juni	2019						
Tirsdag		4	Juni	2019						
Onsdag		5	Juni	2019						
Torsdag		6	Juni	2019						
Fredag		7	Juni	2019						

Jon Skjelsbæk

Ukedag	Dag	Måned	År	Sprint nr	Timer	Kommentar	Dokumentering	Teknisk arbeid	Presentasjon	Administrativt arbeid
Tirsdag		1	Januar	2019		0				
Onsdag		2	Januar	2019		0				
Torsdag		3	Januar	2019		0				
Fredag		4	Januar	2019		0				
Lørdag		5	Januar	2019		0				
Søndag		6	Januar	2019		0				
Mandag		7	Januar	2019		1				1
Tirsdag		8	Januar	2019		0				
Onsdag		9	Januar	2019		2.5				2.5
Torsdag		10	Januar	2019		3.5				3.5
Fredag		11	Januar	2019		5				5
Lørdag		12	Januar	2019		1.5				1.5
Søndag		13	Januar	2019		0				
Mandag		14	Januar	2019	1	5.5				2.5
Tirsdag		15	Januar	2019	1	0				
Onsdag		16	Januar	2019	1	5.5				5.5
Torsdag		17	Januar	2019	1	7				4
Fredag		18	Januar	2019	1	6.5				6.5
Lørdag		19	Januar	2019	1	0				
Søndag		20	Januar	2019	1	0				
Mandag		21	Januar	2019	2	7.5				1.5
Tirsdag		22	Januar	2019	2	0				
Onsdag		23	Januar	2019	2	7.5				4.5
Torsdag		24	Januar	2019	2	7.5				2
Fredag		25	Januar	2019	2	7.5				2.5
Lørdag		26	Januar	2019	2	0				
Søndag		27	Januar	2019	2	2				
Mandag		28	Januar	2019	3	8				3
Tirsdag		29	Januar	2019	3	0				
Onsdag		30	Januar	2019	3	11				1
Torsdag		31	Januar	2019	3	9.5				9.5
Totalt i januar						98.5	26	12.5	13.5	46.5
Fredag		1	Februar	2019	3	6.5				5.5
Lørdag		2	Februar	2019	3	4				
Søndag		3	Februar	2019	3	4				
Mandag		4	Februar	2019	4	5				
Tirsdag		5	Februar	2019	4	0				
Onsdag		6	Februar	2019	4	7.5				7.5
Torsdag		7	Februar	2019	4	6				3
Fredag		8	Februar	2019	4	4.5				2
Lørdag		9	Februar	2019	4	0				
Søndag		10	Februar	2019	4	0				
Mandag		11	Februar	2019	5	6.5				3
Tirsdag		12	Februar	2019	5	0				
Onsdag		13	Februar	2019	5	8.5				1
Torsdag		14	Februar	2019	5	6.5				4.5
Fredag		15	Februar	2019	5	8.5				1
Lørdag		16	Februar	2019	5	6.5				6.5
Søndag		17	Februar	2019	5	2.5				2.5
Mandag		18	Februar	2019	6	8.5				7.5
Tirsdag		19	Februar	2019	6	0				
Onsdag		20	Februar	2019	6	3				3
Torsdag		21	Februar	2019	6	6.5				4.5
Fredag		22	Februar	2019	6	5				1.5
Lørdag		23	Februar	2019	6	3				2.5
Søndag		24	Februar	2019	6	0				
Mandag		25	Februar	2019	7	7.5				7.5
Tirsdag		26	Februar	2019	7	0				
Onsdag		27	Februar	2019	7	9				7
Torsdag		28	Februar	2019	7	9				7
Totalt i februar						128	16	79	11.5	21.5
Fredag		1	Mars	2019	7	8.5				7.5
Lørdag		2	Mars	2019	7	0				
Søndag		3	Mars	2019	7	0				
Mandag		4	Mars	2019	8	11				9
Tirsdag		5	Mars	2019	8	3				1.25
Onsdag		6	Mars	2019	8	11				4
Torsdag		7	Mars	2019	8	7.5				7.5
Fredag		8	Mars	2019	8	7.5				6
Lørdag		9	Mars	2019	8	10.5				10.5

Jon Skjelsbæk

Ukedag	Dag	Måned	År	Sprint nr	Timer	Kommentar	Dokumentering	Teknisk arbeid	Presentasjon	Administrativt arbeid
Søndag		10 Mars	2019	8	6	Strukturert Alpha 2 for generering av dokumentasjon, kommentert og generert dokumentasjon for Alpha 2, lag til dokumenter i deliverables på drive	6			
Mandag		11 Mars	2019	8	11	Forberedelser til presentasjon 2			11	
Tirsdag		12 Mars	2019	8	5	Forberedelser til presentasjon 2			5	
Onsdag		13 Mars	2019	8	3.5	Presentasjon 2 og utspørring			3.5	
Torsdag		14 Mars	2019	8	0					
Fredag		15 Mars	2019	8	7.5	Sett Get A Grip presentasjon, feilsøkt og fiksa bug i HWL som førte til systemkræs, definert requirement og lagt til i respektive dokumenter	3	3.5		1
Lørdag		16 Mars	2019	8	0					
Søndag		17 Mars	2019	8	2	Lagt til requirements for HWL Alpha 3	2			
Mandag		18 Mars	2019	9	9.5	Sprint møte, progresjonsplanlegging, laget use case diagram og begynt på Diagnostics System problem domain dokument	6.5			3
Tirsdag		19 Mars	2019	9	1.5	Testet en mulig løsning på subscriber system, men sett at det kanskje kan lønne seg å bruke linked list i stedet			1.5	
Onsdag		20 Mars	2019	9	10	Møte Eirik og Lars Helge, gjennomgått SWL med Thomas, sett videre på subscriber system og fått på plass error handling bortsett fra i handshake protokoll			7	3
Torsdag		21 Mars	2019	9	0					
Fredag		22 Mars	2019	9	6	Klargjort HWL for Alpha 3 og sett videre på subscriber system løsning			6	
Lørdag		23 Mars	2019	9	0					
Søndag		24 Mars	2019	9	0					
Mandag		25 Mars	2019	10	0					
Tirsdag		26 Mars	2019	10	0					
Onsdag		27 Mars	2019	10	6	sett videre på subscriber system løsning og lest litt om DHT22 sensor			6	
Torsdag		28 Mars	2019	10	0					
Fredag		29 Mars	2019	10	0					
Lørdag		30 Mars	2019	10	0					
Søndag		31 Mars	2019	10	0					
Totalt i mars					127		48.75	45.75	19.5	13
Mandag		1 April	2019	10	0					
Tirsdag		2 April	2019	10	0					
Onsdag		3 April	2019	10	7.5	Avsluttende forskning på subscriber system. Klar til implementering.			7.5	
Torsdag		4 April	2019	10	0					
Fredag		5 April	2019	10	0					
Lørdag		6 April	2019	10	0					
Søndag		7 April	2019	10	0					
Mandag		8 April	2019	11	8	Delvis integrert Subscriber System Template Library (SSTL) med resten av HWL.			8	
Tirsdag		9 April	2019	11	10	Fikset problemer med SSTL og gjort klart for implementering av request håndtering			10	
Onsdag		10 April	2019	11	7.5	Møte Lars Helge og Karsten. Laget klassediagram for SSTL. Implementert og lagt til subscribe og unsubscribe funksjoner i instruction interpreter. Funnet og sett på feil i LinkedList.	3		4.5	
Torsdag		11 April	2019	11	3	Sett på alternativ løsning for å gjøre SSTL kompatibel med både Arduino og standard C++. Oppdatert timelister.	1		2	
Fredag		12 April	2019	11	7	Møte José. Begynt på LaTeX mal for vedlegg. Sett på encoder spesifikasjon og lest om forskjellige outputs.	2		5	
Lørdag		13 April	2019	11	0					
Søndag		14 April	2019	11	1.5	Gjort ferdig LaTeX mal for vedlegg og fiksa warning i rapporten.	1.5			
Mandag		15 April	2019	12	7.5	Sprint møte. Transparancy møte. Diskusjon rapportoppsett. Gjennomført tester og skrevet verifikasjonsrapport for Alpha 3 og Beta 1.	4.5			3
Tirsdag		16 April	2019	12	6.5	Skrevet dokumenter fra Drive over i Overleaf. Fikset klassediagrammer. Reformulert DS beskrivelse i rapporten.	6.5			
Onsdag		17 April	2019	12	6.5	Møte med Eirik og Lars Helge. Funnet og fikset memory leak i insert instruction. Sett etter flere memory leaks, men fant ingen. Lagt til subscriber functions i klassediagram.	1		4.5	1
Torsdag		18 April	2019	12	6	Møte José. Diskusjon rapportoppsett. Begynt design fase på available instructions.			3	3
Fredag		19 April	2019	12	4	Kjørt tester på forskjellige biblioteker for temperatur sensor på DHT11 og DHT22. Optimalisert kode for DHT22.			4	
Lørdag		20 April	2019	12	0					
Søndag		21 April	2019	12	4	Kjørt tester på DHT11 og DHT22 med DHTlib V0.1.00. Lagt til resultater av tester i HWL Decisions dokument. Lagt til kilder i HWL Decisions dokument.	3		1	
Mandag		22 April	2019	13	7.5	Fikset kildeformat i rapport, appendix- og decisions dokument. Fylt ut decisions dokument med sensor og bibliotek valg. Testet sending av tilgjengelige instruksjoner i separat prosjekt.	2.5		5	
Tirsdag		23 April	2019	13	7.5	Implementert sekvensiell kjøring av instruksjoner for å kunne protothrede rotate instruksjoner, og klargjort for implementering av tilgjengelig instruksjoner og beskjed når instruksjon er ferdig kjørt.			7.5	
Onsdag		24 April	2019	13	9	Implementert sneding av tilgjengelig instruksjoner i separat prosjekt.			9	
Torsdag		25 April	2019	13	10	Forbedret Json_container og fjerna Json_object_container og Json_array container. Laget Instruction_container, og endret SADM_functions for å fungere med nye Json_container og Instruction_container.			10	
Fredag		26 April	2019	13	6	Begynt migrering inn i HWL prosjekt.			6	
Lørdag		27 April	2019	13	7.5	Migrering inn i HWL prosjekt			7.5	
Søndag		28 April	2019	13	6	Ferdigstilt migrering. Oppdatert klassediagram.	2		4	
Mandag		29 April	2019	14	9.5	Ferdigstilt sending av available instructions. Endret så available data og available instructions blir sendt uten request etter fullført handshake. Lagt til ping. Oppdatert klassediagram.	1		8.5	
Tirsdag		30 April	2019	14	7.5	SpaceNight prat. Sett på datablader for motor og driver. Begynt på Request_container.			6.5	1
Totalt i april					149.5		28	113.5	0	8
Onsdag		1 Mai	2019	14	6	Minor changes to error messages. Added contains method to LinkedList. Added Func_ptr and Request_container. Renamed Instruction_handler to Message_handler and changed it to work with Request_container. Renamed rotate_deg to rotate_degrees and it's parameter identifier is now degrees rather than deg. Created separate lists for requests and instructions in Message_handler. This is to enable instructions to be queued and requests to be executed at once.			6	
Torsdag		2 Mai	2019	14	7.5	Små endringer på HWL for å fungere optimalt med SWL. Kommentert alle header filer.	6		1.5	
Fredag		3 Mai	2019	14	0					
Lørdag		4 Mai	2019	14	0					
Søndag		5 Mai	2019	14	0					
Mandag		6 Mai	2019	15	7.5	API Doc HWL, LinkedList og SSTL. Oppdatert klassediagram.	7.5			
Tirsdag		7 Mai	2019	15	7	Oppdatert forsider på dokumentasjon for eldre versjoner. Gått igjennom og endret HWL Decisions, lagt til nye overskrifter.	7			
Onsdag		8 Mai	2019	15	7.5	Ny motor driver test			7.5	

Jon Skjelsbæk

Ukedag	Dag	Måned	År	Sprint nr	Timer	Kommentar	Dokumentering	Teknisk arbeid	Presentasjon	Administrativt arbeid
Torsdag		9 Mai	2019	15	7.5	Implementert kode for ny motor i HWL.		7.5		
Fredag		10 Mai	2019	15	7.5	Dokumentasjon av HWL og test av diverse sensorer.	5.5	2		
Lørdag		11 Mai	2019	15	7.5	Dokumentasjon av HWL og test av ny motor med DS.	3.75	3.75		
Søndag		12 Mai	2019	15	5	Dokumentasjon av HWL	5			
Mandag		13 Mai	2019	16	7.5	Fikset på attachments. Generert kode dokumentasjon for Beta 1. Lagt til attachments i rapporten.	7.5			
Tirsdag		14 Mai	2019	16	0					
Onsdag		15 Mai	2019	16	7.5	Gone through references. Fixed appendicies. Reduced serial listener delay to almost nothing. Added two different modes, automatic and manual. Implemented functionality for swapping between the two when recieving input from switch.	4	3.5		
Torsdag		16 Mai	2019	16	0					
Fredag		17 Mai	2019	16	7.5	Gått igjennom Josés kommentarer i rapport og rettet opp.	7.5			
Lørdag		18 Mai	2019	16	0					
Søndag		19 Mai	2019	16	7.5	Fikset attachements og rapport layout.	7.5			
Mandag		20 Mai	2019		13	Gått igjennom rapport og rettet opp. Lagt til forside på vedlegg som mangler. Fikset feil i nesten alle forsider.	13			
Tirsdag		21 Mai	2019		9	Fikset feil i resten av forsidene. Gått igjennom kommentarer i rapport og rettet opp. Testet i eksternt prosjekt og lagt til lastcelle og posisjonssensor i HWL.	9			
Onsdag		22 Mai	2019		0					
Torsdag		23 Mai	2019		0					
Fredag		24 Mai	2019		0					
Lørdag		25 Mai	2019		0					
Søndag		26 Mai	2019		0					
Mandag		27 Mai	2019		0					
Tirsdag		28 Mai	2019		0					
Onsdag		29 Mai	2019		0					
Torsdag		30 Mai	2019		0					
Fredag		31 Mai	2019		0					
Totalt i mai					115		83.25	31.75	0	0
Lørdag		1 Juni	2019							
Søndag		2 Juni	2019							
Mandag		3 Juni	2019							
Tirsdag		4 Juni	2019							
Onsdag		5 Juni	2019							
Torsdag		6 Juni	2019							
Fredag		7 Juni	2019							

Bachelor of Engineering

Project appendix

Winter semester 2019

Group contract

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	SatLight
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document is a contract concerning each and everyone one of the group members. It comprises rules and guidelines that has to be adhered by everyone over the course of this project.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

Group Contract

This contract regulates the cooperation between the members of the USN Bachelor group 5.

The undersigned undertakes to comply with the guidelines listed in paragraphs 1-5. The agreement is valid until June 14, 2019, unless all members wish to adjust the contract.

Working Hours

Core Time	09.00 – 15.00
Working Days	Mon, Tue (After Easter), Wed, Thu, Fri
Stand-up	09.00 – 09.15
Sprint Meeting	Mondays 09.00 – 10.00

An 8-hour working day is expected. four days a week until the examination in the students' respective subjects

1. Objectives of the project

- 1.1 Give good answers to the problem and produce high quality results.
- 1.2 Achieve interdisciplinarity and a product all group members can relate to their own field of study.
- 1.3 Gain greater insight into each other's subject areas, and group dynamics in general.

2. Orders

- 2.1 The group members must meet at. 09.00 every Monday, Wednesday, Thursday and Friday, unless otherwise agreed.
- 2.2 Absence must be notified to the other group members in a timely manner. This also applies to arrivals over ten minutes.
- 2.3 Each work session begins with a 15-minute stand-up. Here everyone can present their work since last. Any issues are pointed out and added to the agenda for discussion outside stand-up.
- 2.4 Each work session shall be conducted in an agreed hourly system.
- 2.5 We undertake to meet prepared according to current agreements.
- 2.6 We shall use confluence actively to document the progress of the others at the group and undertake to make all relevant information available to all team members.
- 2.7 The role that rolls between the team members. We shall keep records with important secretaries and the chairman of the meeting. confluence The protocol is written within the end of the week by the secretary.
- 2.8 As a group member, one is committed to keeping up to date with and understanding of what the others are doing.

3. Decision rules

3.1 Decisions that affect the group, the project or the whole of the product should be discussed and decided jointly. These are brought in protocol. See point 2.7.

3.2 In the event of absence, you lose your right to influence and must abide by what is adopted.

3.3 If no agreement is reached after several attempts, advice shall be sought from external persons, preferably supervisors.

4. Work process

4.1 The group will set specific sub-goals with clear time frames during the work.

4.2 When a sub-goal is reached, the work must be documented with a level of detail that is sufficiently later report writing. This applies to both product and process report.

4.3 If a sub-goal is not reached, the cause is documented, and a new time frame is added.

4.4 It is the duty of all group members to ensure that plagiarism or cheating does not occur, intentionally or without. If a student discovers plagiarism or cheats on other forms that can affect the outcome of the whole group, this must be notified. It should be attempted at the lowest possible level before warning sensors.

5. Environment

5.1 We shall maintain a good tone within the group and be good at giving each other feedback and praise.

5.2 Criticism is allowed, but it must always be objective.

5.3 Each person is responsible for addressing issues along the way if something should appear.

5.4 The group will meet as far as possible to socialize at regular intervals.

Kongsberg, 14. Januar 2019

Thomas Mundal

Erica Fegri

Ming Kit Wong

Jon Skjelsbæk

Rita Hogstad

Håvar Østrem

Bachelor of Engineering**Project appendix****Winter semester 2019****LaTeX Bibliography Guide**

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

The LaTeX Bibliography Guide is a document describing how to replicate the IEEE standard for citation when adding references to LaTeX documents.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

LaTeX Bibliography Guide

A LaTeX bibliography is a (.bib) file in the LaTeX repository, which holds references that can be cited within tex documents. It consists of two elements, entry- and field types, where field types are contained within an entry, and a single entry represents a single reference. The figure below show an example implementation of a reference in a bib file.

```

13 @article{
14     bib1,
15     author = {D. Ince},
16     title = {Acoustic coupler},
17     journal = {A Dictionary of the Internet},
18     year = {2006},
19     volume = {1},
20     number = {2},
21     pages = {70--160},
22     month = {May},
23     note = {Oxford Reference Online,
24     | \url{http://www.oxfordreference.com} [Accessed: May 24, 2007].}
25 }
```

Entry types

In the figure above the field types (author, title, journal and so on) are encapsulated in an entry, which in this case is of type article. The entry is defined as follows: “@entryType{” where the corresponding field types are defined inside the curly brackets.

Different entry types:

- **article**
 - An article from a journal or magazine.
- **book**
 - A book with an explicit publisher.
- **booklet**
 - A work that is printed and bound, but without a named publisher or sponsoring institution.
- **conference**
 - The same as `inproceedings`, included for Scribe compatibility.
- **inbook**
 - A part of a book, usually untitled. May be a chapter (or section, etc.) and/or a range of pages.
- **incollection**
 - A part of a book having its own title.

- **inproceedings**
 - An article in a conference proceedings.
- **manual**
 - Technical documentation.
- **mastersthesis**
 - A Master's thesis.
- **misc**
 - For use when nothing else fits.
- **phdthesis**
 - A Ph.D. thesis.
- **proceedings**
 - The proceedings of a conference.
- **techreport**
 - A report published by a school or other institution, usually numbered within a series.
- **unpublished**
 - A document having an author and title, but not formally published.

Field types

Field types are used to define the reference by adding e.g. the name and author of an article, or in general information that's to be displayed when printing the bibliography.

Different field types:

- **address**
 - Publisher's address (usually just the city, but can be the full address for lesser-known publishers)
- **annotate**
 - An annotation for annotated bibliography styles (not typical)
- **author**
 - The name(s) of the author(s) (in the case of more than one author, separated by and)
- **booktitle**
 - The title of the book, if only part of it is being cited
- **chapter**
 - The chapter number
- **crossref**
 - The key of the cross-referenced entry
- **doi**
 - digital object identifier
- **edition**
 - The edition of a book, long form (such as "First" or "Second")
- **editor**

- The name(s) of the editor(s)
- **howpublished**
 - How it was published, if the publishing method is nonstandard
- **institution**
 - The institution that was involved in the publishing, but not necessarily the publisher
- **journal**
 - The journal or magazine the work was published in
- **key**
 - A hidden field used for specifying or overriding the alphabetical order of entries (when the "author" and "editor" fields are missing). Note that this is very different from the key (mentioned just after this list) that is used to cite or cross-reference the entry.
- **month**
 - The month of publication (or, if unpublished, the month of creation)
- **note**
 - Miscellaneous extra information
- **number**
 - The "(issue) number" of a journal, magazine, or tech-report, if applicable. Note that this is not the "article number" assigned by some journals.
- **organization**
 - The conference sponsor
- **pages**
 - Page numbers, separated either by commas or double-hyphens.
- **publisher**
 - The publisher's name
- **school**
 - The school where the thesis was written
- **series**
 - The series of books the book was published in (e.g. "[The Hardy Boys](#)" or "[Lecture Notes in Computer Science](#)")
- **title**
 - The title of the work
- **type**
 - The field overriding the default type of publication (e.g. "Research Note" for techreport, "{PhD} dissertation" for phdthesis, "Section" for inbook/incollection)
- **volume**
 - The volume of a journal or multi-volume book
- **year**
 - The year of publication (or, if unpublished, the year of creation)

Composition

Every entry type has different field types that are either required for the entry type, or optional. If the entry type you're using don't support a field type you want to define, you might have to find an alternative entry type that does.

Entry types and their required/optional field types:

- **article**
 - Required fields: author, title, journal, year, volume
 - Optional fields: number, pages, month, doi, note, key
- **book**
 - Required fields: author/editor, title, publisher, year
 - Optional fields: volume/number, series, address, edition, month, note, key, url
- **booklet**
 - Required fields: title
 - Optional fields: author, howpublished, address, month, year, note, key
- **inbook**
 - Required fields: author/editor, title, chapter/pages, publisher, year
 - Optional fields: volume/number, series, type, address, edition, month, note, key
- **incollection**
 - Required fields: author, title, booktitle, publisher, year
 - Optional fields: editor, volume/number, series, type, chapter, pages, address, edition, month, note, key
- **inproceedings**
 - Required fields: author, title, booktitle, year
 - Optional fields: editor, volume/number, series, pages, address, month, organization, publisher, note, key
- **manual**
 - Required fields: title
 - Optional fields: author, organization, address, edition, month, year, note, key
- **mastersthesis**
 - Required fields: author, title, school, year
 - Optional fields: type, address, month, note, key
- **misc**
 - Required fields: none
 - Optional fields: author, title, howpublished, month, year, note, key
- **phdthesis**
 - Required fields: author, title, school, year
 - Optional fields: type, address, month, note, key
- **proceedings**
 - Required fields: title, year
 - Optional fields: editor, volume/number, series, address, month, publisher, organization, note, key
- **techreport**

- Required fields: author, title, institution, year
- Optional fields: type, number, address, month, note, key
- **unpublished**
 - Required fields: author, title, note
 - Optional fields: month, year, key

Replication of IEEE Citation Guidelines in LaTeX

The document “[IEEE Citation Guidelines](#)” can be found on the [Drive](#) under [Support material/Standards](#). We want to list and cite every reference according to the guidelines and examples in this document, so it’s very important to read through it carefully at least once, and verify that every reference and citation in every document meet the criterias stated in the standard.

The figures below show an example on how to reference an article from an online encyclopedia in LaTeX by replicating the “[IEEE Citation Guidelines](#)”. What we’re trying to achieve when referencing to different reference types is to have them look as similar as possible to the examples in the guidelines. Which entry- and field field types that are used to achieve this doesn’t really matter as long as it replicates the corresponding example to the best of their ability.

Online article reference from “[IEEE Citation Guidelines](#)”:

- [2] D. Ince, “Acoustic coupler,” in *A Dictionary of the Internet*. Oxford University Press, [online document], 2001. Available: Oxford Reference Online, <http://www.oxfordreference.com> [Accessed: May 24, 2007].

Online reference LaTeX implementation:

```

13 @article{
14     bib1,
15     author = {D. Ince},
16     title = {Acoustic coupler},
17     journal = {A Dictionary of the Internet},
18     year = {2006},
19     volume = {1},
20     number = {2},
21     pages = {70--160},
22     month = {May},
23     note = {Oxford Reference Online,
24     | \url{http://www.oxfordreference.com} [Accessed: May 24, 2007].}
25 }
```

LaTeX output of the above implementation:

- [2] D. Ince. Acoustic coupler. *A Dictionary of the Internet*, 1(2):70–160, May 2006. Oxford Reference Online, <http://www.oxfordreference.com> [Accessed: May 24, 2007].

Bachelor of Engineering

Project appendix

Winter semester 2019

V-Model inspired workflow

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Erica Fegri
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

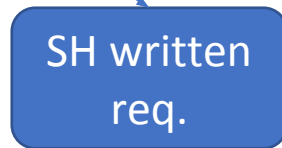
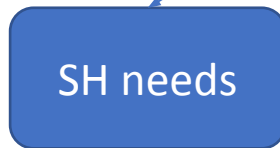
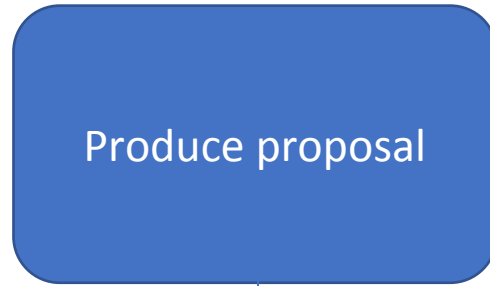
Abstract

This document visualizes and describes in short terms the core of the workflow implemented in the agile development process.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

Input:
Oppgave

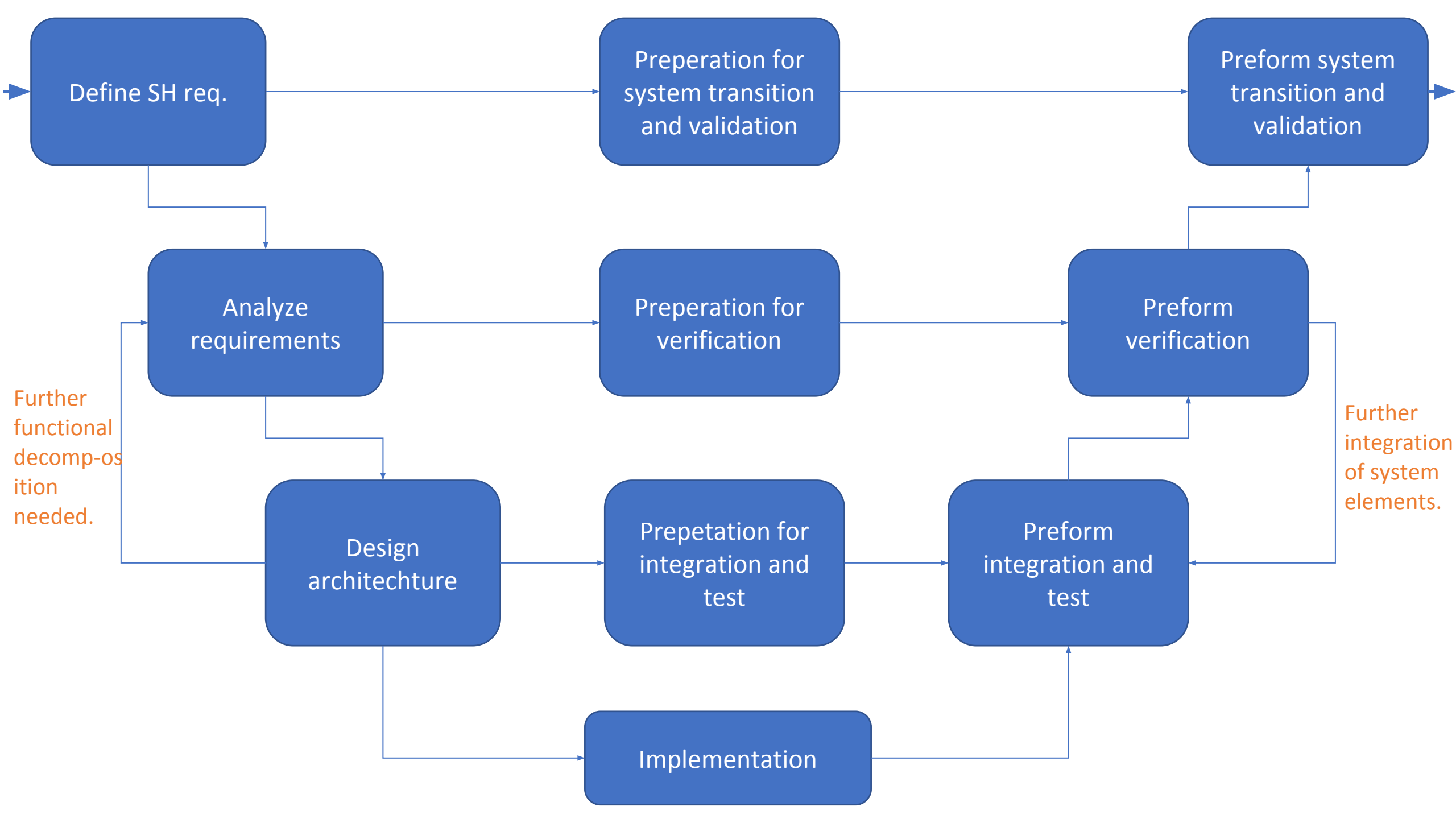


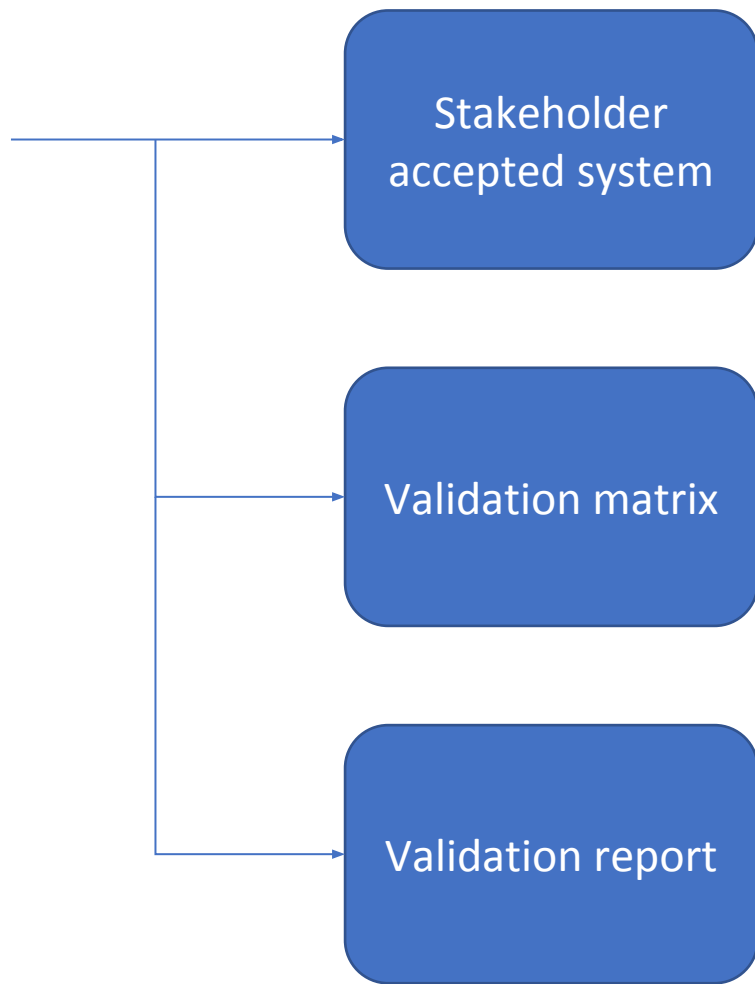
- Action plan: How we plan to respond to RFP.
- Develop proposal
- Proposal Review and Risk analysis
- Requirement Review
- Fianancial Review

- Analyse requirements and expectations
- Analyse needs
- Possible conflicts between stakeholders of interest?

- SH needs may be more than the written requirements.
- Identified through discussion with SH.

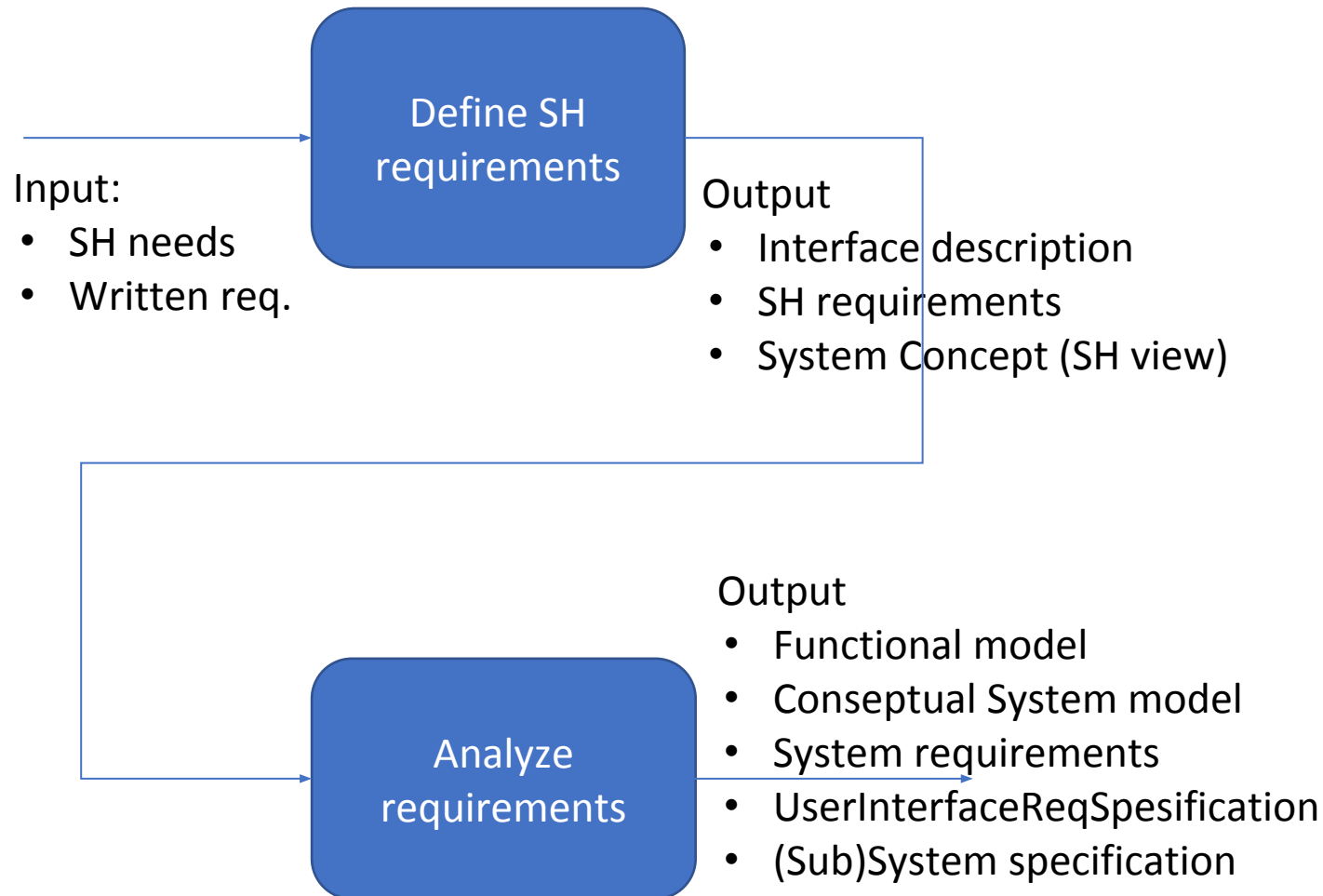
RFP: Request For Proposal realised from costumer





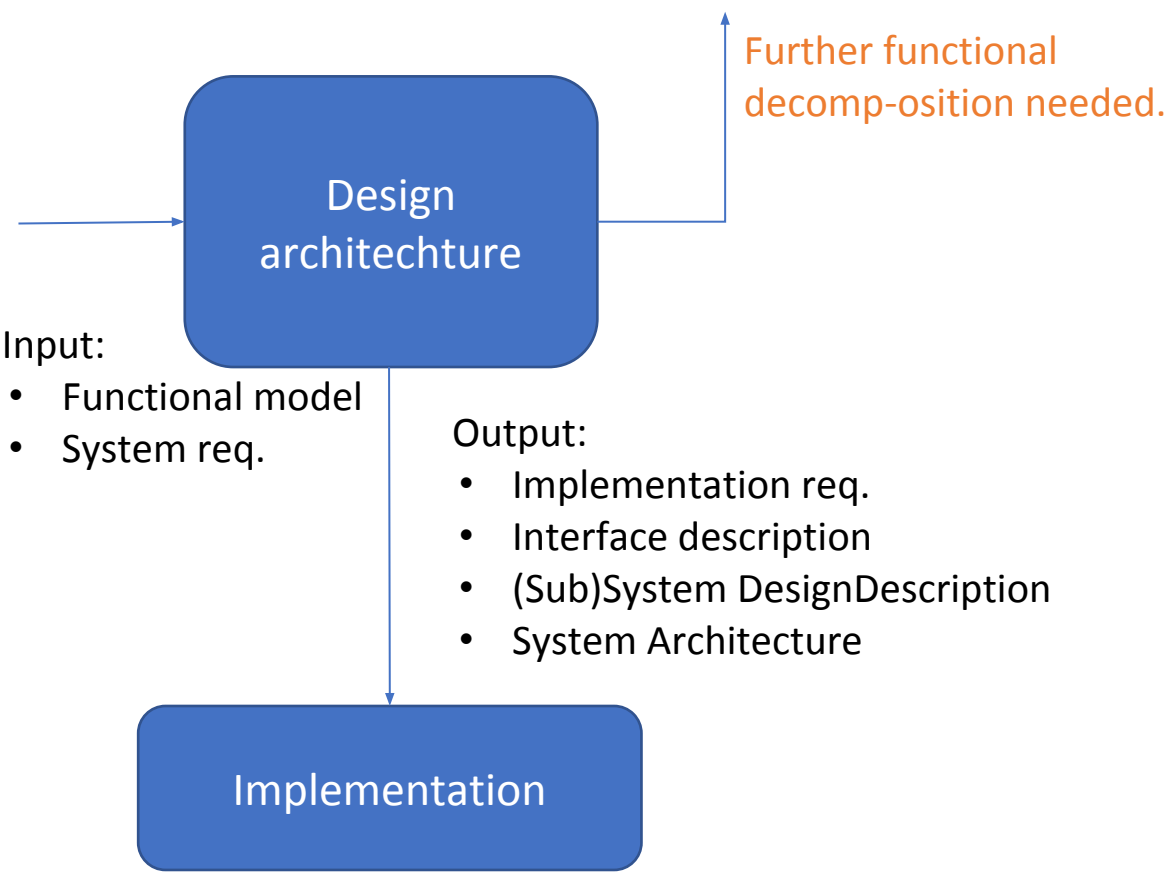
Kanskje ikke aktuelt for oss?

Forklaringer:



- Define the requirements a system shall be validated against
- Assure it can provide the services needed
- Define environment

- SH requirements transferred to system functional and non-functional.
- Shall not implies any specific implementations
- May be reused
- Each system req. will be traced to SH-req or higher level



- Overall system concepts and –requirements analysed by refining and structuring.
- Evaluating concepts and solutions.
- Purpose: Split system into designable elements – meet expectations and SH req.
- Find implementation req. Needed to meet the system req.

Preparation for system transition and validation

Input:
-Interface description.
-SH req.
-System Concept (SH view)

Output:
-SystemValidation Procedure
-ValidationEnvironment Description
-Validation Matrix

- Selecting the system/system elements for validation
- Specification of validation environment, procedures and criteria
- Plan and specify the validation of SH req. And needs

Preparation for verification

Input:
-Functional model
-System Req.

Output:
-Verification matrix and description
-Verification environments description

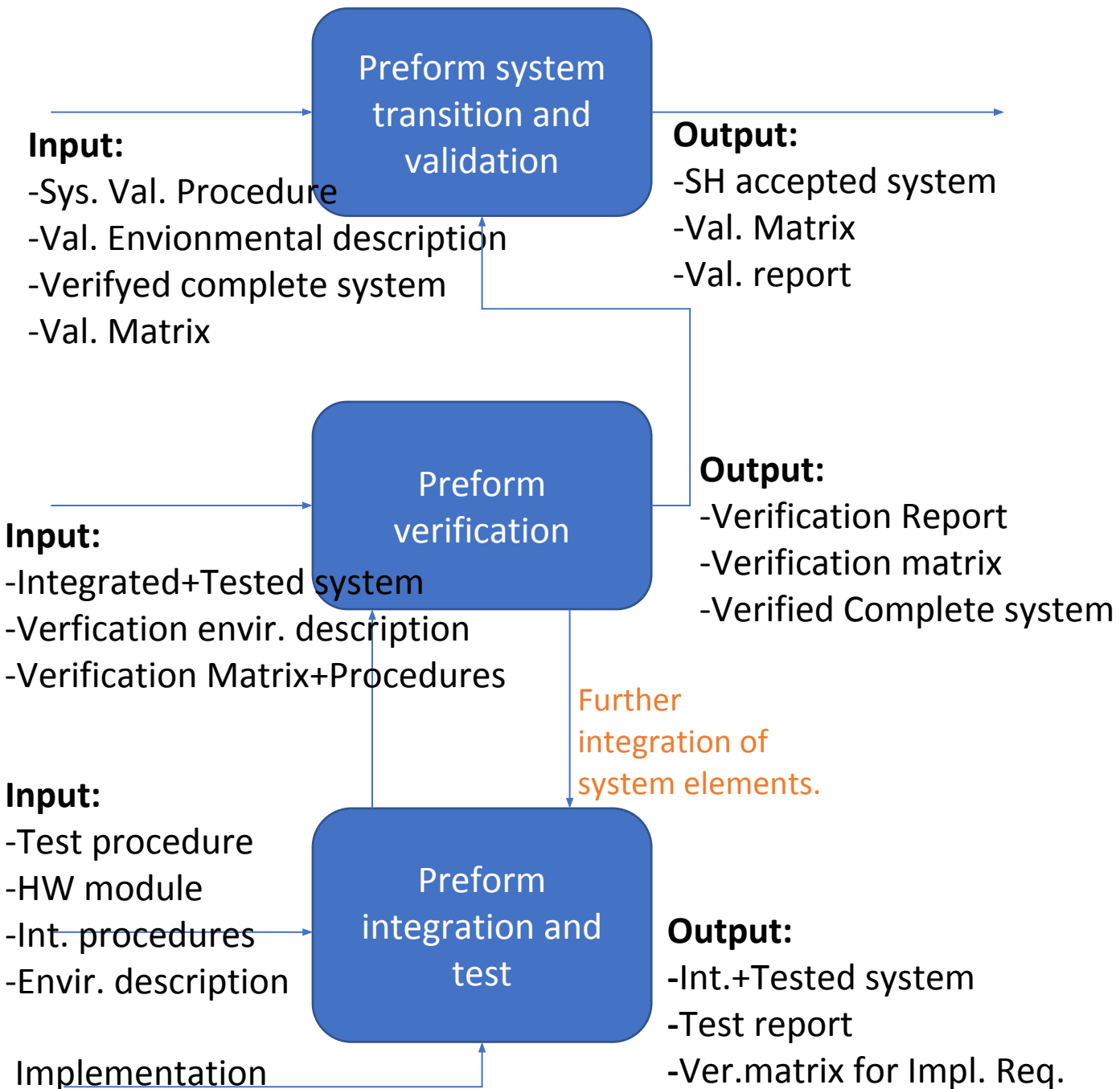
Identify and describe verification procedures. Includes planning → system meets req. Verification procedures and criteria Support tools, test equipment and SW, simulations, prototypes.

Preparation for integration and test

Input:
-Behav. Model
-Implementation Req.
-Interface Description
-System architecture

Output:
-Integration&Test environment description
-Integration + Test procedure
-Verification matrix for implement req.

- Various forms: SW integration, Electrical + mechanical integration, System integration.
- How and what sequence integrations and test are performed to:
 - Meet milestones and efficient production
 - Ensure quality
 - Reduce risks early as possible



- Propose: Demonstrate complete system fulfill intended use in intended enviroment
- → **Right system was built!**

- System Verification: that implemented system meets requirements assosiated with system or system element.

- Assemble a system that is consisted with architectural design.
- Combines system elements to complete/partial system config.
- Tested to reveal error/design weakness.

Bachelor of Engineering**Project appendix****Winter semester 2019****Procedure for agenda and minutes**

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Ming Kit Wong
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document describes the routine for handling of documents related to agenda for meetings and minutes after the meeting. This document will also describe the routine and order of who is the chairman of the meeting (CoM) and referee.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

Procedure for agenda and minutes

This document describes the routine for handling of documents related to agenda for meetings and minutes after the meeting. This document will also describe the routine and order of who is the chairman of the meeting (CoM) and referee.

Agenda

The agenda is to be sent to supervisor at least one day before the meeting, preferably two days in advance. It is the responsibility of the CoM to create and send the agenda to the supervisor, on behalf of the project team. Following is the procedure for creating agenda and meeting request as an document

1. **Open the file Agenda** in Google Drive > Bachelor 2019 > Meetings > Agenda
2. **Make new folder** and name it with the date of the meeting
3. Open the new folder, **create a Google Doc** and name it with the date of the meeting
4. **Make the team aware** of the new document so everyone can add topics to the meeting

Attachments that is relevant or related to the agenda must me added to the folder for that certain meeting. When all the attachments are ready to be sent, they must be joined together as one PDF.

Agenda for external supervisor is sent to eirik.demmo.normann@kongsberg.com

Agenda for internal supervisor is sent to josemmf.usn@gmail.com

Minutes

Minutes, also called Meeting notes, are created on Confluence by the referee. Following is the procedure for creating a Meeting note

1. **Open Confluence**
2. Click on the **plus-sign (+)** on the left side
3. Click on the **icon named Meeting notes and Create**
4. **Fill out the form** and edit if necessarily
5. **Click publish** when finished

Export the document as PDF when published (click on the three dots on top right side) and send to the all the people who were invited to the meeting.

The order of the roles

The roles are changed weekly to ensure that everyone gets to lead the meetings and to be referee. The person who have been referee is CoM in the following week. The order is as following:

1. Erica
2. Thomas
3. Rita
4. Håvar
5. Ming Kit
6. Jon

The responsibilities of the CoM is to send the agenda ahead of the meeting and to distribute the time, so that all the topics in the agenda are discussed within the meeting time.

The responsibilities of the referee is to take meeting notes during the meeting and send it to everyone who was invited to the meeting.



Bachelor of Engineering

Project appendix

Winter semester 2019

Team Naming and Logo Design

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Håvar Østrem
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document explains the process of naming the project team and the process of designing the team's logo, with preliminary sketches and iterations of designs included.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

Contents

- 1 Naming process** **2**

- 2 Logo Design Process** **2**
 - 2.1 Multidisciplinary Engineering Concepts 2
 - 2.2 Text concepts 3
 - 2.3 SADM Assembly Concepts 4
 - 2.4 Space Concepts 4
 - 2.5 Sun and Satellite logo concepts 5

- 3 Final Design** **7**

- 4 References** **8**

1 Naming process

The process of finding a fitting project team name and suitable logo was started early in the project. The process of naming was done democratically in a period of about two weeks. All members was allowed to submit as many name suggestions as they wanted, which accumulated to a total of twenty-nine suggestions. All members had three votes each to give to their preferred name or names. Changing votes was allowed as new suggestions were submitted. The twenty-ninth and last suggestion was SatLight. It was immediately more popular than any previous suggestion, resulting in us ending the election and adapting it as our team name. With the exception of SatLight, some of the most popular names includes KONSAP (Kongsberg Nanosat Solar Array Project), N-SAP (NanoSat Solar Array Project) and KONSADA (Kongsberg Nanosat Solar Array Driving Assembly).

2 Logo Design Process

The first logo design sketches were drawn on paper around the same time as the naming process began. A few of them are name specific, while most of them are not. A few of the sketches were also drawn with computer graphics tools, usually starting with MS Paint, and then edited with Gimp. [1]

2.1 Multidisciplinary Engineering Concepts

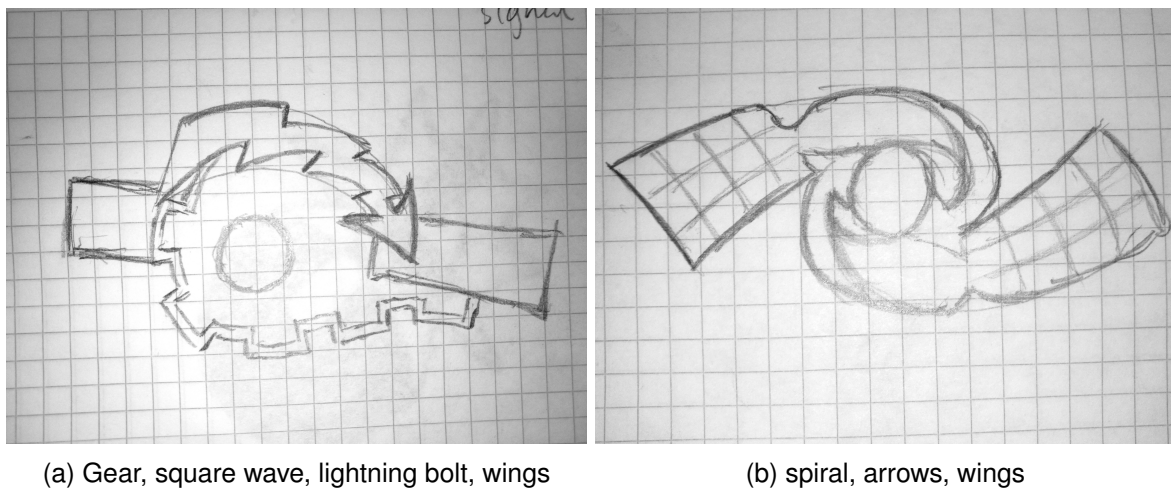


Figure 1: Sketches of multidisciplinary logo themes

As we are a multidisciplinary team, a few attempts were made to depict this in the logo design. In figure 1 (a) there is a gear in the center representing mechanical engineering, where the outline of the gear departs from the imaginary circle of the gear and becomes a square wave which is supposed to represent software engineering (although it is really a hardware aspect of computers), a lightning bolt across the top represents electronics engineering. Figure 1 (b) were also an attempt to represent the disciplines. However, the sketch had become too complex, even before the any disciplines were represented. Both sketches also have satellite

solar array wings. This concept was difficult to design simply enough for a logo and was quickly abandoned.

2.2 Text concepts

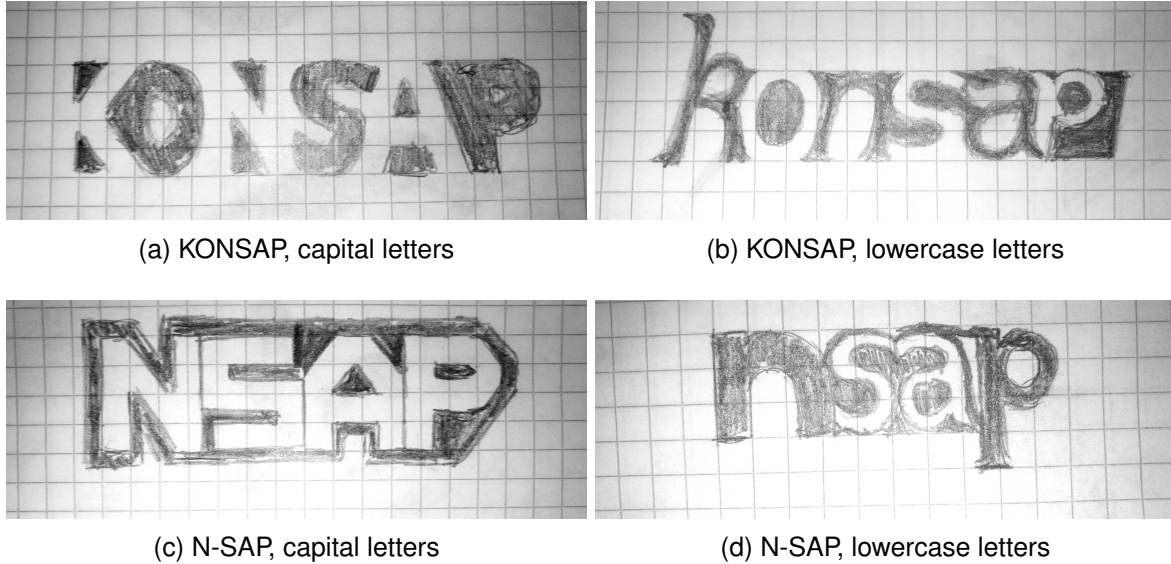


Figure 2: Sketches of text logo themes

The sketches in figure 2 shows various, pure text based logo designs. KONSAP and N-SAP was some of the more popular names, at the time when these were drawn.



Figure 3: Computer graphic

Figure 3 shows the same design as the sketch in figure 2, made with computer graphics and with the addition of a color inverted version. Both of them are using letters depicted in negative and positive space in an alternating pattern. [2] Many recognizable logo designs are purely text based, such as Google and Coca-Cola. However, it is difficult to display the team's values or mission, exclusively with text.

2.3 SADM Assembly Concepts

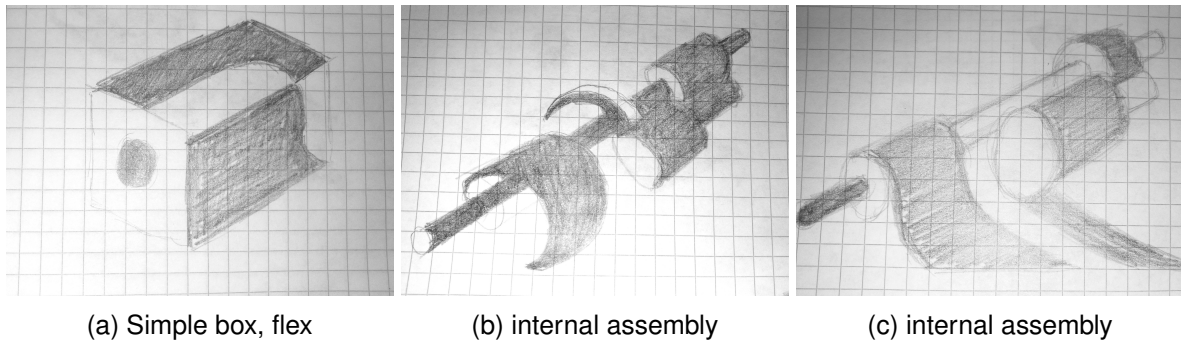


Figure 4: Sketches of SADM assembly logo themes

Since the scope of the project was fairly well defined early in the project, one possible logo concept was a simplified drawing of the actual assembly as shown in figure 4. These turned out either too vague and conservative, like in (a) or too complex, like in (b) and (c).

2.4 Space Concepts

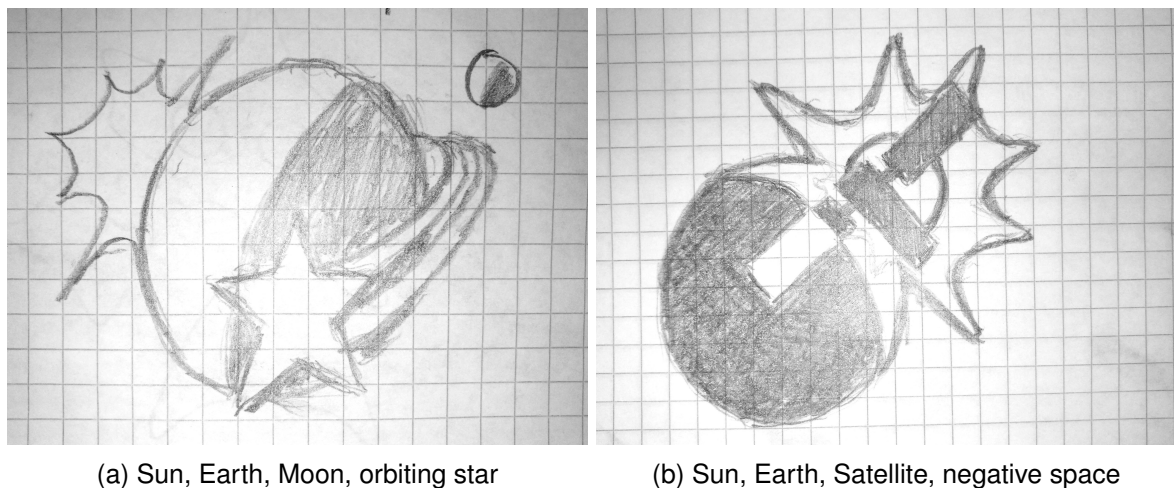


Figure 5: Sketches of space logo themes

Using elements associated with space and space technology is a more general approach than the actual assembly, for describing the team's mission. Figure 5 shows two sketches of this sort. (a) depicts the Earth partially eclipsing the Sun, the Moon and a star in place of a satellite, orbiting the earth. In (b) The Earth is again eclipsing the Sun and in front of them is a satellite.

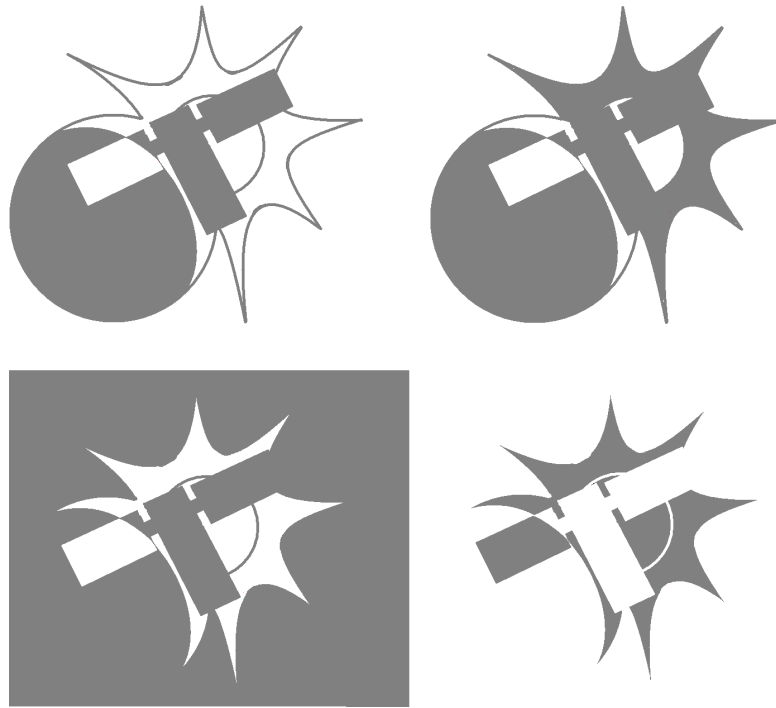


Figure 6: Computer graphic

Figure 6 shows a computer graphics design of the sketch in 5 (b), as well as some modifications.

2.5 Sun and Satellite logo concepts

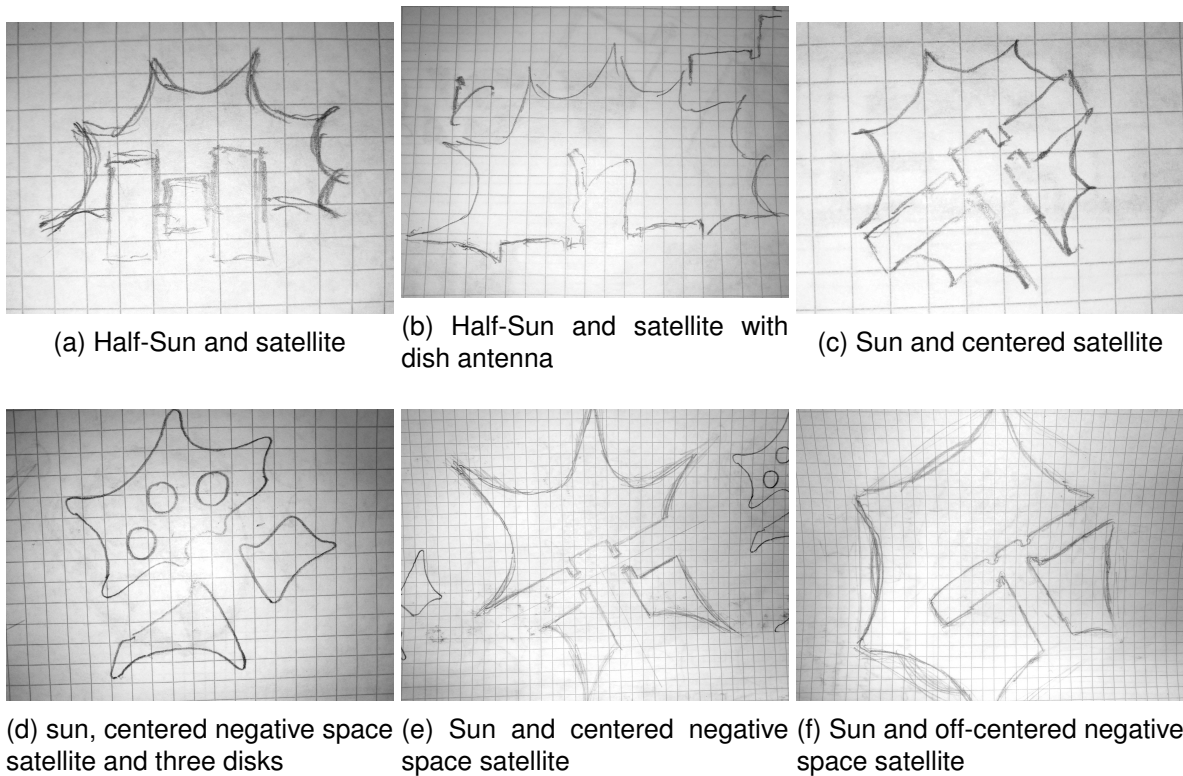


Figure 7: Sketches of Sun-Satellite logo themes

Figure 7 shows the sketches with a “Sun and satellite”-theme. This is a narrower and simplified concept of the space-theme, which after some modification, eventually lead to the final logo design. The three disks in (d) was meant to represent the three disciplines of the team. (d), (e) and (f) are six-pointed stars to represent the six members of the team. (f) is the one closest to the final logo design.

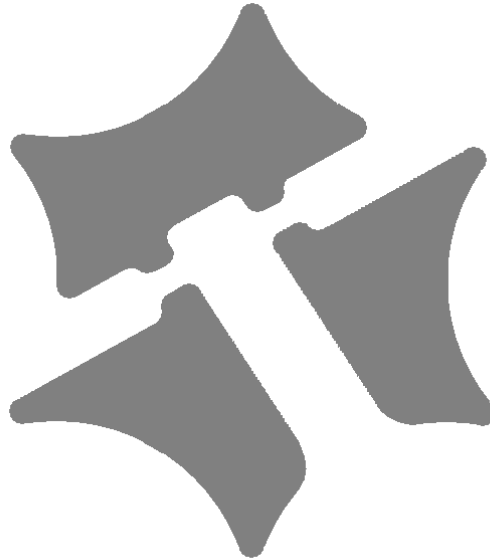


Figure 8: Computer graphic

Figure 8 shows the first logo of this concept made with computer graphics.

3 Final Design



SatLight

(a) Name placed beneath logo



(b) Name placed to the right of the logo

Figure 9: Official logo

The final logo shown in figure 9, shows a satellite in the negative space in front of the sun, indicating the field of our production. The sun is depicted as a six-pointed star which represent the six members of the team. Both the Sun and the satellite has their corners slightly rounded to give the logo a softer look and also for it to be more scalable. The colour orange was chosen, as it is said to represent things like youth, cheerfulness and optimism by various websites and online forums. However, very few of these have references and some of the claims contradicts each other. The faultiness of these claims are discussed in a report by Zena O'Connor. [3] Regardless, orange is as much of a color as any and well received by all members of the team. The exact hue of the color is #FF6D0D in Hex or (255, 109, 13) in RGB. The font used for the logo text is calibri as its rounded letters matches the roundness of the logo. The letters are kerned, meaning the spacing between some of letters are slightly shortened to make the text more compact and consistent. [4]

4 References

- [1] The GIMP Team, “GNU image manipulation program,” <http://www.gimp.org/> [Accessed: October 27, 2017].
- [2] M. Boddy-Evans, “What is negative space in art?” 2019, <https://www.thoughtco.com/negative-space-definition-2573838> [Accessed: April 28, 2019].
- [3] Z. O’Connor, “Colour psychology and colour therapy: Caveat emptor,” 2011, <https://onlinelibrary.wiley.com/doi/full/10.1002/col.20597/> [Accessed: May 9, 2019].
- [4] Øyvin Rammen, “Kerning,” February 2018, <https://snl.no/kerning> [Accessed: April 28, 2019].

Bachelor of Engineering**Project appendix****Winter semester 2019****Description - Requirement sheet v3.1**

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Erica Fegri
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document is a description of the requirement sheet. The purpose of this documents is to enlighten readers and writers of the requirement sheet how to properly use the sheet.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

Description - Requirement sheet v3.1

Link to Requirements sheet:

<https://docs.google.com/spreadsheets/d/1sxhgZIxREtATWt48BzuWZCH-MJr8LqtyCqZKEIUhgwM/edit#gid=0>

Requirement are organized by groups:

1. **R1 - Equipment breakdown**
2. **R2 - Functional requirements**
3. **R3 - Physical Requirements**
4. **R4 - Interface Requirements**
5. **R5 - Environmental Requirements**
6. **R6 - Space Material Compatibility Requirements**
7. **R7 - Reliability Requirements**
8. **R8 - Lifetime Requirements**

Card color:

The color is ment to indicate the state of the requirement.

White: OK - no issues found.

Orange: Under revision.

Red: The requirement is not met or cannot be fulfilled

Purple: The req. is no longer valid, and has been replaced with another revision.

ID	Rev	Suc ces sor	Us er St ory -ID	V.-I D	Owner	W R-I D	S A D M/ D S	Description	Pri orit y	Sig n
R2-0 1			US N5- 119	AV-0 1	KSS	SA DM- RE Q-0 002 0	SA DM	The SADM shall link the solar array wing mechanically to the spacecraft structure during launch and all operational phases	Disc arde d	EF
R2-0 2				TV-0 1	KSS	SA DM- RE Q-0	SA DM	The SADM shall orient the solar arrays as specified herein	Disc arde d	EF

						002 0				
R2-0 3			US N5- 120	TV-0 2	KSS	SA DM- RE Q-0 002 0	SA DM	The SADM shall hold the solar array wings in position when not powered	Sho uld	EF
R2-0 4			US N5- 121	TV-0 3	KSS	SA DM- RE Q-0 002 0	SA DM	The SADM shall provide angular position information The DS shall provide angular position information of the SADM.	Mus t	EF
R2-0 5			US N5- 122	TV-0 4	KSS	SA DM- RE Q-0 002 0	SA DM	The SADM shall transfer solar array grounding lines	Mus t	EF
R2-0 6					PG		DS	The DS shall imitate the spacecraft main computer	Can Disc ard ed	EF
R2-0 7				TV-0 6	PG		DS	The DS shall provide information about the SADM, such as; - Input values - Output values	Disc ard ed	EF

Requirement: Short description of the requirement. This description can be as technically as necessary.

R-ID: Requirement ID consist of Requirement group followed by number in group.
Example: R1-4. This is a requirement that belongs in the group R1 - Functional Requirement and is listed as number 4 in this group.

- If an requirement has a predecessor this is implemented in the R-ID. Ex: R1-01.01. R1-01 is the predecessor of this requirement.

Rev: If this requirement is a revision, it will be indicated with A for the first revision, B for the second....

OBS! Remember to update the date and sign.

Example of how to update revision:

ID	Rev	Successor	UserStory-ID	V-ID	Owner	WR-ID	SADM/DS	Description	Priority	Sign
R2-01	A				KSS	SADM-REQ-0020	SADM	<p>The SADM shall link the solar array wing mechanically to the spacecraft structure during launch and all operational phases</p> <p>The new version of the requirement.</p>	Must	EF-EF

UserStory-ID: The ID of the userstory that connects this requirement to the userstories in Jira.

Priority:

Four groups: Must, should, can or discarded.

Owner: This is typically stakeholder, KSS or USN, or it can be the project team depending on the origin of the requirement.

WR-ID: Stands for Written Requirement ID. The written requirement from stakeholder (KSS) has an ID in document XXX-RS-01 NanoSat SADM Requirement Specifications.

Test/Test-ID:

Letter that indicates how verification method is defined and Test-ID that points to the specific ID of the test.

Verification letter definition:

T = Test	Verification by test consists of measuring, product performance and functions under representative or specific simulated environmental worst case conditions that show that the requirements are met.
A = Analysis	Verification by analysis consists of performing theoretical calculation or empirical evaluation using techniques agreed with the commissioner/KSS
R = Review of Design	Verification by Review of Design (ROD) consists of using evidence in records, drawings or documents that unambiguously shows that the requirements are met.
I = Inspection	Verification by inspection consists for visual determination of physical characteristics.

SADM/DS:

SADM - Solar Array Drive Mechanism

DS: Diagnostic System

This column is to indicate if the requirement is related to the Diagnostic System or the SADM.

Sign:

The initials of the owner.

EF - Erica Fegri

HØ - Håvar Østrem

JS - Jon Skjælsbek

TM - Thomas Mundal

MKW - Ming Kit Wong

RH - Rita Hogstad

Date: Creation date.**Standard-ID from written requirements:**

SD#	Reference	Issue/rev	Title
[SD1]	ECSS-E-ST-33-01C	15 February 2017	Mechanisms
[SD2]	ECSS-E-ST-10-03C	1 June 2012	Testing
[SD3]	ECSS-E-ST-20-07C	7 February 2012	Electromagnetic Compatibility
[SD4]	MIL-STD-202G	28/06/13	Test Method Standard, Electronic and Electrical Component Parts
[SD5]	ECSS-Q-ST-70C	6 March 2009	Materials, mechanical parts and processes
[SD6]	ECSS-Q-ST-70-71C	15 October 2014	Materials, processes and their data selection
[SD7]	ECSS-E-ST-32-08C	31 July 2008	Materials
[SD8]	ECSS-Q-ST-30-11C	Rev 1 4 October 2011	Derating – EEE Components
[SD9]	ESCC 3901/001		Polyimide Insulated Wires and Cables Low Frequency 600V -100 to +200 degree C

[SD10]	ESCC 3901/002		Polyimide Insulated Wires and Cables Low Frequency 600V -100 to +200 degree C
[SD11]	ESCC3401/001	8	CONNECTORS, ELECTRICAL, RECTANGULAR, NON-REMOVABLE SOLDER BUCKET, PCB AND WIRE-WRAP CONTACTS AND REMOVABLE COAXIAL AND POWER CONTACTS, BASED ON TYPE D*M
[SD12]	ESCC3401/044	2	CONNECTORS, ELECTRICAL, CIRCULAR, BAYONET COUPLING, REMOVABLE CRIMP CONTACTS, BASED ON MIL-C-38999 SERIES II
[SD13]	ECSS-E-HB-31-03	A	Space Engineering, Thermal Analysis Handbook
[SD14]	ESCC 3901/019	2	POLYIMIDE INSULATED WIRES AND CABLES, LOW FREQUENCY, 600V, -200 TO +200°, BASED ON TYPE SPL
[SD15]	ECSS-Q-ST-70-01C	15 November 2008	Cleanliness and contamination control



Bachelor of Engineering

Project appendix

Winter semester 2019

Requirements

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	SatLight
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains all requirements, both self defined and those from DSS.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

ID	Rev	Successor	UserStory-ID	V.-ID	Owner	WR-ID	SADM/DS	Description	Priority	Sign	Date	Responsible	Parent
R1-01			USN5-204	IV-01 RV-01	KSS	SADM-REQ-0010	SADM	<p>The SADM will be composed of the following componets:</p> <ul style="list-style-type: none"> - Drive line: Gear, motor - Main shaft and bearings - Housing - Signal and power transfer unit 	Must	EF	25.01.19	Mech Elec	
R1-02	A		USN5-205	IV-59	PG		DS	<p>The DS will be composed of the following components:</p> <ul style="list-style-type: none"> - FPGA as motor controller - Realy controlled by FPGA to transfer power to motor - Position sensor - Temprature Sensor -Diagnostics output <p>Rev A)</p> <ul style="list-style-type: none"> -HMI software for transmitting test instructions and visualizing results - Motor and sensor controller (Microcontroller / FPGA) -Relays to switch on or off the different motor phases? -Different purpose sensors to measure outputs -Diagnostics output -etc TO BE CONTINUED <p>Rev B)</p> <ul style="list-style-type: none"> - HMI software for transmitting test instructions and visualizing results - Diagnostics output <p>Cabinet for the DS with following equipment:</p> <ul style="list-style-type: none"> - DC Power supply - Motor driver and sensor controller (Microcontroller / FPGA) - Stepper motor Drive - Angular position sensor - Fuse(s) 	Must	EF Rev A) JS Rev B) EF	25.01.19 Rev A) 06.02.19 Rev B) 06.02.19	Elec Mech Data	

ID	Rev	Successor	UserStory-ID	V.-ID	Owner	WR-ID	SADM/DS	Description	Priority	Sign	Date	Responsible	Parent
R2-01			USN5-119	AV-01	KSS	SADM-REQ-00020	SADM	The SADM shall link the solar array wing mechanically to the spacecraft structure during launch and all operational phases	Discarded	EF	25.01.19	Mech Elec	
R2-02				TV-01	KSS	SADM-REQ-00020	SADM	The SADM shall orient the solar arrays as specified herein	Discarded	EF	25.01.19	Mech Elec Data	
R2-03			USN5-120	TV-02	KSS	SADM-REQ-00020	SADM	The SADM shall hold the solar array wings in position when not powered	Should	EF	25.01.19	Mech	
R2-04			USN5-121	TV-03	KSS	SADM-REQ-00020	SADM	The SADM shall provide angular position information The DS shall provide angular position information of the SADM.	Must	EF	25.01.19	Mech Elec Data	
R2-05			USN5-122	TV-04	KSS	SADM-REQ-00020	SADM	The SADM shall transfer solar array grounding lines	Must	EF	25.01.19	Elec Mech	
R2-06					PG		DS	The DS shall imitate the spacecraft maincomputer	Can Discarded	EF	25.01.19	Data	
R2-07				TV-06	PG		DS	The DS shall provide information about the SADM, such as; - Input values - Output values	Discarded	EF	25.01.19	Data Elec	
R2-08			USN5-123	RV-02	KSS	SADM-REQ-00030	SADM	The SADM physical reference frame shall be as defined below and depicted in following figure: - The origin shall be at the intersection of the SADM mounting plane with the S/C interface - The X-axis shall be in point - The Y-axis shall be coincident - The Z-axis shall be in the mounting plane and point through a mounting point	Discarded	RH MKW RH	25.01.19 06.02.2019 20.04.19	Mech	
R2-09			USN5-124	AV-02	KSS	SADM-REQ-00090	SADM	The SADM shall be compliant with the solar array wing inertia as follows: - Ixx = up to 0.013 kgm ² - Iyy = up to 15 kgm ² - Izz = up to 15 kgm ² Note: The values for the solar array Moment of Inertia are provided at the SADM Interface	SHOULD	RH MKW RH	25.01.19 06.02.2019 14.02.2019	Mech	
R2-10			USN5-125	TV-07 AV-07	KSS	SADM-REQ-00100	SADM	The SADM functional performance and motorization margin shall be determined considering the following solar array characteristics when attached to the SADM: - 1st torsional mode of the SA: 0,2 Hz with inertia of 31 kgm ² - 2nd torsional mode of the SA: 0,7 Hz with inertia of 30 kgm ² - 3rd torsional mode of the SA: 3,7 Hz with inertia of 33 kgm ² - Considered SA modal dampning 0,3% or Q = 166 - SA residual inertia: 16 kgm ² - Mass 110 kg +/- 5% - Mass: 0.40kg +/- 5%	SHOULD Discarded	RH MKW RH	25.01.19 06.02.2019 07.02.19	Mech	

ID	Rev	Successor	UserStory-ID	V.-ID	Owner	WR-ID	SADM/DS	Description	Priority	Sign	Date	Responsible	Parent
R2-11			USN5-130	TV-08 RV-04	KSS	SADM-REQ-00040	SADM	The SADM shall be able to oscillate or rotate around the X- axis in both directions (CW or CCW), with a total angular range of minimum +/- 180 degrees	MUST	RH EF	25.01.19 07.02.19	Mech	
R2-12			USN5-126	AV-03	KSS	SADM-REQ-00050	SADM	The SADM shall satisfy motorization margin requirement as specified in §4.7.5.3 per ECSS-E-ST-33-01, [SD1]	CAN Discarded	RH MKW EF	25.01.19 06.02.2019 19.05.19	Mech Elec	
R2-13			USN5-127	TV-09 AV-08	KSS	SADM-REQ-00060	SADM	The unpowered holding torque of the SADM shall be greater than or equal to 2 (TBC) Nm	CAN	RH EF	25.01.19 13.02.19 06.04.19	Elec Mech	
R2-14			USN5.128	TV-10 AV-09	KSS	SADM-REQ-00070	SADM	The SADM shall be able to deliver a minimum holding torque of 8 Nm with a motor running at a driving current of 150 mA The SADM shall be able to deliver a minimum holding torque of 0.04 Nm with a motor running at a driving current of 1 A	SHOULD	EF	25.01.19	Elec Mech	
R2-15			USN5-129	TV-11 AV-10	KSS	SADM-REQ-00080	SADM	The SADM irreversibility quasi-static torque shall be less than 70 (TBC) Nm Note: it means that if the quasi-static torque exceeds this maximum value, the SADM must rotate in order to avoid any degradation	Should	RH	25.01.19	Mech	
R2-16			USN5-208	TV-12	KSS	SADM-REQ-00110	SADM	Frequency Range on ground (Hz): - 0,1 - 1 - 0,001 - 1 - 10 - 0,01 - 10 - 200 - 0,1 - 200 - 1000 - 0,1 All forces and torques are peak-to-peak at SADM/Spacecraft interface, taking into account all phenomena (step, acceleration, backlash)	Discarded	RH MKW	25.01.19 06.02.2019	Mech Elec	
R2-17			USN5-141	TV-13	KSS	SADM-REQ-00120	SADM	This is NA The SADM shall have enough torque margins to withstand a short circuit of motor winding without loss of functionality at 0,1 degrees/s (nominal speed) Note: using the redundant interface in case of failure in nominal winding and vice versa	Discarded	RH	25.01.19	Mech Elec	
R2-18			USN5-134 USN5-136	AV-04	KSS	SADM-REQ-00125	SADM	The SADM shall keep its position under the following spacecraft rotational accelerations: - Powered: +/- 0,02 rad/s ² (TBC) - Unpowered: +/- 0,01 rad/s ² (TBC) An amplification factor of 2 shall be considered	CAN Discarded	RH	25.01.19	Mech Elec	
R2-19			USN5-151	TV-14	KSS	SADM-REQ-00130	SADM	The SADM shall be able to operate in normal angular speed mode between 0,0 X degrees/s to 5 degrees/s. The maximum output angular velocity capability shall be at least 3 degrees/s	MUST	RH	25.01.2019	Elec Mech Data	

ID	Rev	Successor	UserStory-ID	V.-ID	Owner	WR-ID	SADM/DS	Description	Priority	Sign	Date	Responsible	Parent
R2-20			USN5-133	TV-15	KSS	SADM-REQ-00140	SADM	The SADM angular speed at the output shaft shall have a stability performance better than: - 30 arcsec/s for frequency lower than 0,5 Hz - 0,5 arcsec/s in the frequency range from 0,5 Hz to 3 Hz - 1000 arcsec/s for frequency higher than 3 Hz	GAN Discarded	RH	25.01.19	Mech Elec	
R2-21				AV-05	KSS	SADM-REQ-00150	SADM	The SADM shall provide a full step resolution at output shaft <1 degrees	SHOULD	RH EF	25.01.19	Elec Mech Data	
R2-22			USN5-142	TV-16 AV-13	KSS	SADM-REQ-00210	SADM	The SADM shall provide an axial translation stiffness of >- 1.0 x 10 ⁸ N/m	GAN Discarded	RH	25.01.19 19.05.19	Mech	
R2-23			USN5-143	TV-17 AV-14	KSS	SADM-REQ-00220	SADM	The SADM shall provide a radial translation stiffness of >- 1.0 x 10 ⁸ N/m with the solar array interface constrained in cross axis bending (interface plane remains parallel under loading)	Discarded	RH	25.01.19	Mech	
R2-24			USN5-144	TV-18 AV-15	KSS	SADM-REQ-00230	SADM	The SADM shall provide a cross axis bending stiffness of >- 1.0 X 10 ⁵ Nm/rad	Discarded	RH	25.01.19	Mech	
R2-25			USN5-145	TV-19 AV-16	KSS	SADM-REQ-00240	SADM	The SADM shall provide a torsional stiffness of >_ 10 ⁴ Nm/rad (assuming a blocked motor shaft, at 5 to 15 Nm external load on the output shaft) In the SADM backlash angular range (SADM-REQ-00170) with low loading (only overcoming frictional torque) an area with "zero stiffness" is expected and therefore not considered as a deviation to the requirement	Discarded	RH	25.01.19	Mech	
R2-26			USN5-147	TV-20 AV-17	KSS	SADM-REQ-00250	SADM	The SADM shall have a first Eigen-frequency > 250 Hz when its stator part is mounted on rigid interfaces and when its rotor part is connected to a dummy mass of minimum 2,5 kg with CoG placed at minimum 20 mm from the SADM I/F	Discarded	RH MKW	25.01.19 07.02.2019	Mech	
R2-27			USN5-74 USN5-275	IV06 AV-34 TV-50	PG		DS-SW	The DS Software Layer shall display diagnostics data in real time	Must	TM	28.01.19	Data	
R2-28			USN5-77	IV-16	PG		DS-SW	The DS Software Layer shall have a modular design for easy addition of diagnostic components	Must	TM	29.01.19	Data	
R2-29			USN5-118	UTV-1 UTV-2	PG		DS-SW	The DS Software Layer shall save diagnostic data to a database	Should	TM	30.01.19	Data	
R2-30			USN5-96	TV-39	PG		DS-HW	The DS Hardware layer must be able to control the SADM for testing purposes.	Must	JS	06.02.19	Data	
R2-31			USN5-97	TV-40	PG		DS-HW	The DS Hardware layer must be able to transmit sensor data to the Software layer for comparison and visualization purposes.	Must	JS	06.02.19	Data	
R2-32		R4-20	USN5-131	IV-13	PG		DS-SW	The DS Software layer shall be able to calibrate position and speed of the SADM drive shaft	Must	TM	07.02.19	Data	
R2-33			USN5-146	UTV-24 UTV-6 IV-15	PG		DS-SW	The DS Software layer should be able to remember previous COM settings and try to establish a connection to HW layer with these	Should	TM	07.02.19	Data	

ID	Rev	Successor	UserStory-ID	V.-ID	Owner	WR-ID	SADM/DS	Description	Priority	Sign	Date	Responsible	Parent
R2-34		R2-34.01 R2-34.02 R4-22 R4-23	USN5-279	UTV-26	PG		DS-SW	DS SW should be able to configure desired maximum and minimum limits for operational sensor values to display status indicator on live input values during diagnostics run. Values outside accepted values should indicate deviation	Should	TM		Data	
R2-34.01			USN5-281		PG		DS-SW	DS SW should provide functionality to save diagnostics settings to a template	Should	TM		Data	R2-34
R2-34.02					PG		DS-SW	DS SW should be able to load and apply settings from a diagnostics settings template	Should	TM		Data	R2-34
R2-35		R2-35.01 R2-35.02 R4-24	USN5-304		PG		DS-SW	Should be able to run acceptance test on applied test parameters and generate test report	Should	TM		Data	
R2-35.01			USN5-302		PG		DS-SW	Should be able to define acceptance test parameters	Should	TM		Data	R2-35
R2-35.02			USN5-303		PG		DS-SW	Should provide functionality for saving acceptance test parameters to an acceptance test template	Should	TM		Data	R2-35
R2-36			USN5-131	TV-44	PG		DS-HW	The DS Hardware layer shall be able to interpret and execute instructions received from the software layer.	Must	JS	22.02.19	Data	
R2-37					PG		DS-SW	Should provide capabilities for communicate trough ethernet network	Should	TM		Data	
R2-38			-	R	PG		DS-HW	The sensor for angular position must fulfill : - Electrical angle of 360 degrees - Mechanical angle of <360 degrees - Suitable for 24VDC or 5VDC voltage level - Provide a output signal suitable for Arduino Mega	Must	EF	17.04.19	Elec	

ID	Rev	Successor	UserStory-ID	V.-ID	Owner	WR-ID	SADM/DS	Description	Priority	Sign	Date	Responsible	Parent
R3-01			UNS5-153	AV-06 TV-21	KSS	SADM-REQ-00260		The mass of the SADM shall not exceed 2kg (Harness not included)	Must	RH	25.01.19	Mech Elec	
R3-02			USN5-154	RV-03 TV-22	KSS	SADM-REQ-00270		The SADM shall not exceed an envelope of 100 X 100 X 200 (millimeters)	Must	RH	25.01.19	Mech	

ID	Rev	Successor	UserStory-ID	V.-ID	Owner	WR-ID	SADM/DS	Description	Priority	Sign	Date	Responsible	Parent
R4-01			USN5-157	RV-05	KSS	SADM-REQ-00280	SADM	The SADM mechanical interfaces to S/C and Solar Array shall be according to the SADM supplier	Must	EF	25.01.19	Mech	
R4-02			USN5-158	RV-06	KSS	SADM-REQ-00285	SADM	The zero reference position shall be clearly marked on the SADM rotor and stator and included on the Interface Control Drawing	ISCARDE	EF	25.01.19	Mech	
R4-03			USN5-159	RV-07	KSS	SADM-REQ-00300	SADM	The length of the rotor harness shall be 500 mm flying leads	Must	EF	25.01.19	Elec	
R4-04			USN5-163	RV-08	KSS	SADM-REQ-00310	SADM	There shall be integrated connectors on the stator side, connector size and type are to be determined by SADM supplier	Must	EF RH	25.01.19	Mech Elec	
R4-05			Work in progress...	TV-23	KSS	SADM-REQ-00340	SADM	The maximum total current transfer in the power lines shall be 20A forward and 20A return; - 10A each direction each solar array	Must	EF	25.01.19	Elec	
R4-06			USN5-164	RV-09	KSS	SADM-REQ-00420	SADM	Electrical grounding between SADM shaft and housing shall be redundant	SHOULD	EF	25.01.19	Elec	
R4-07			USN5-160	RV-10	KSS	SADM-REQ-00430	SADM	Solar Array grounding lines shall be insulated from SADM grounding.	SHOULD	EF	25.01.19	Elec	
R4-08			USN5-161	RV-11	KSS	SADM-REQ-00470	SADM	The stepper motor electrical shall have the following nominal electrical interface: - 2 phase, bipolar communication with 1.8 degrees full step angle - Phase resistance: 18.5 Ohm + - 10% - Phase inductance: 37mH + - 20% - Power: < 5W The stepper motor electrical shall have the following nominal electrical interface: - 2 phase, bipolar communication with 1.8 degrees full step angle - Phase resistance: 3.5 ohm/phase - Phase inductance: 1.2 mH/phase	Should	EF	25.01.19	Elec Mech	
R4-09			USN5-162	RV-12	KSS	SADM-REQ-00490	SADM	The motor interface shall be according to standard NEMA dimensions.	CAN	EF	25.01.19	Elec Mech	
R4-10				-	KSS	SADM-REQ-00510	SADM	The SADM emissivity shall be above 0.8. The materials with stable thermo-optical properties are required.	ISCARDE	EF	25.01.19	Mech	
R4-11			USN5-165	AV-19	KSS	SADM-REQ-00520	SADM	The SADM power consumption shall not exceed XX Watts.	SHOULD	EF	25.10.19	Elec	
R4-12	A		USN5-77	TV-35	PG		DS-SW	The DS Software Layer should have standardized interface for input data The DS Software Layer should use JSON for input data	Must	TM	28.01.19	Data	
R4-13	A		USN5-77	TV-36	PG		DS-SW	The DS Software Layer should have standardized interface for output data The DS Software Layer should use JSON for output data	Must	TM	28.01.19	Data	
R4-14			USN5-76	IV-07	PG		DS-SW	DS Software layer GUI shall support changing underlying COM port settings	Should	TM	29.01.19	Data	

ID	Rev	Succesor	UserStory-ID	V.-ID	Owner	WR-ID	SADM/DS	Description	Priority	Sign	Date	Responsible	Parent
R4-15			USN5-87	TV-37	PG		DS-SW	DS Software layer GUI shall support selection of diagnostic data output view	Must	TM	30.01.19	Data	
R4-16			USN5-75	TV-38	PG		DS-SW	DS Software layer GUI shall support displaying historical data	Should	TM	31.01.19	Data	
R4-17	A	R4-17.01 R4-17.02	USN5-98	IV-08	PG		DS-HW	DS Hardware layer shall have a standardized-expandable Input handler, buffering input data according to input type. Rev. A) DS Hardware layer shall have a general input handler responsible for every kind of input.	Must	JS Rev A) JS	06.02.19 Rev. A) 21.02.19	Data	
R4-18	A	R4-18.01 R4-18.02	USN5-99	IV-09	PG		DS-HW	DS Hardware layer shall have a standardized-expandable output handler translating the input data passed from a specific controller, and output the translated information to the device to be controlled. Rev. A) DS Hardware layer shall have a general output handler supporting the different protocols/formats needed in order to communicate with the different connected devices.	Must	JS Rev A) JS	06.02.19 Rev. A) 21.02.19	Data	
R4-19	A		USN5-100	IV-10	PG		DS-HW	There shall be an interface between the input handler and the output handler. Preferably one for each device to be controlled. Rev. A) There shall be an interface responsible for fetching from the input handler and passing to the output handler.	Must	JS Rev A) JS	06.02.19 Rev. A) 21.02.19	Data	
R4-17.01	B		USN5-101	TV-41	PG		DS-HW	The input handler of the DS Hardware layer must be compatible with the software layer of the diagnostic system. Rev. A) The input handler must support JSON input received from the software layer. Rev. B) The HWL must support JSON input received from the software layer.	Must	JS Rev. A) JS Rev. B) JS	06.02.19 Rev. A) 21.02.19 Rev. B) 07.03.19	Data	R4-17
R4-18.01	B		USN5-102	TV-42	PG		DS-HW	The output handler of the DS Hardware layer shall support the different protocols needed in order to communicate with the different connected devices. Rev. A) The output handler shall incorporate serial printing of JSON formatted data. Rev. B) The HWL shall incorporate serial printing of JSON formatted data.	Must	JS Rev. A) JS Rev. B) JS	06.02.19 Rev. A) 21.02.19 Rev. B) 07.03.19	Data	R4-18
R4-20			USN5-131	IV-14	PG		DS-SW	The DS Software Layer should provide an interface for adjusting calibration settings such as position and speed of the SADM motor	Must	TM	07.02.19	Data	R2-33
R4-21					PG		DS-HW	DS HW Layer needs to encode double data types as strings with comma sign (",") as decimal delimiter in JSON data that is being delivered to SW layer	Must	TM	13.02.19	Data	
R4-22					PG		DS-SW	DS SW should display alert information about sensor readings that are not within the configured range according to the applied diagnostics settings	Should	TM	15.02.2019	Data	R2-34
R4-23					PG		DS-SW	Should provide a graphical user interface for defining diagnostics settings parameters	Should	TM	15.02.2019	Data	R2-34
R4-24					PG		DS-SW	Should provide a graphical user interface for defining acceptance test parameters	Should	TM	15.02.2019	Data	R2-35
R4-17.02					PG		DS-HW	The input handler shall provide given sensor readings when prompted.	Must	JS	21.02.19	Data	R4-17

ID	Rev	Successor	UserStory-ID	V.-ID	Owner	WR-ID	SADM/DS	Description	Priority	Sign	Date	Responsible	Parent
R4-18.02					PG		DS-HW	The output handler shall utilize JSON formatted instructions to control the SADM.	Must	JS	21.02.19	Data	R4-18
R4-25			USN5-226	IV-17 TV-43	PG		DS	Shall follow a unified communication protocol	Must	TM	21.02.19	Data	
R4-26			USN5-251	TV-47	PG		DS-HW	The HWL should be able to enable or disable different sensors according to received instructions.	Should	JS	15.03.2019	Data	
R4-27			USN5-311	TV-52	PG		DS-HW	The HWL should provide a list of available instructions to the SWL when prompted.	Should	JS	24.04.2019	Data	

ID	Rev	Successor	UserStory-ID	V.-ID	Owner	WR-ID	SADM/DS	Description	Priority	Sign	Date	Responsible	Parent
R5-01			USN5-166	RV-13	KSS	SADM-REQ-00530		The SADM shall be designed to meet the following requirements for temperature under the following environmental conditions: - On-Ground: 23 +- 5 degrees - Launch: 20 +- 5 degrees - In-orbit: see SADM-REQ-00610 Note: This requirement does not exist. - In-orbit: See SADM-REQ-00600	Should	EF RH	25.01.19 28.04.19	Mech Elec	
R5-02			USN5-168	RV-14	KSS	SADM-REQ-00530		The SADM shall be designed to meet the following requirements for pressure under the following environmental conditions: - On-Ground: P Atmos - Launch: P Atmos to 10 ⁻⁵ mbar - In-orbit: 10 ⁻⁵ mbar	DISCARDED	EF	25.01.19	Mech Elec	
R5-03			USN5-167	RV-15	KSS	SADM-REQ-00530		The SADM shall be designed to meet the following requirements for humidity under the following environmental conditions: - On-Ground: 30% < Hr < 65% - Launch: 0% < Hr < 65% - In-orbit: Vacuum	Can DISCARDED	EF	25.01.19	Mech Elec	
R5-04			USN5-169	RV-16	KSS	SADM-REQ-00540		The SADM structural analysis shall take into account the margins and factors described in §4.7.5.2 of ECSS-E-ST-33-01 [STD1]	Should	EF	25.01.19	Mech	
R5-05			USN5-170	RV-17	KSS	SADM-REQ-00550		The SADM and GSE design shall allow mounting of accelerometers (flat areas aligned with the coordinate system). Note: Not necessary for prototype, but for "Test Prediction Analysis" we should define where the output comes from	Can Discarded	EF	25.01.19	Mech	
R5-06			USN5-171	TV-25 AV-24	KSS	SADM-REQ-00560		The SADM shall be able to withstand the following quasi-static loads applied in the SADM solar array interface and with the SADM constrained by its normal screws in a rigid (infinitely stiff) spacecraft interface Maximum moment: Axial: +- 0 N Radial: +- 1500 N Bending: +- 300 N Maximum radial and axial loads: Axial: +- 2000 N Radial: +- 1500 N Bending: +- 260 Nm	Can	RH	25.01.19	Mech	

ID	Rev	Successor	UserStory-ID	V.-ID	Owner	WR-ID	SADM/DS	Description	Priority	Sign	Date	Responsible	Parent
R5-07			USN5-172	TV-26 AV-25		SADM-REQ-00570		<p>The SADM shall be able to withstand the following sinusoidal vibration qualification levels for all three orthogonal axis. The sweep rate is specified by ECSS-E-ST-10-03C [SD2]</p> <p>All axis: Frequency (Hz): Acceleration (g): - 5 - Max shaker capacity (max 24g) - 22 - Max shaker capacity (max 24g) - 22 - 24 - 140 - 24</p>	Should	RH	25.01.19	Mech	
R5-08			USN5-173	TV-27 AV-26		SADM-REQ-00580		<p>The SADM shall be able to withstand the following random vibration environment in a single axis in any direction (X and Z-axis and out of plane Y-axis). The duration is specified by ECSS-E-ST-10-03C [SD2]. Notching is allowed in order to protect SADM sensitive components and can be incorporated with the customer's approval. Mechanical analysis shall set the basis for maximum allowed gRMS for sensitive components.</p> <p>Out of Plane: FREQ(Hz): ASD(g^2/Hz): Db/OCT: gRMS: - 20 - 0,1008 - * - 19,3 - 100 - 0,5000 - 3,00 - 19,3 - 400 - 0,5000 - 0,00 - 19,3 - 2000 - 0,0345 - (-5.00) - 19,3</p> <p>In Plane FREQ(Hz): ASD(g^2/Hz): Db/OCT: gRMS: - 20 - 0,0422 - * - 12,5 - 100 - 0,2100 - 3,00 - 12,5 - 400 - 0,2100 - 0,00 - 12,5 - 2000 - 0,01450 - (-5.00) - 12,5</p>	Can	RH	25.01.19	Mech Elec	
R5-09			USN5-174	TV-28		SADM-REQ-00590		<p>The SADM shall withstand the following shock profile:</p> <p>Frequency (Hz): SRS (Q = 10): - 100 - 20 g - 500 - 500 g - 2000 - 2000 g - 10 000 - 2000 g</p>	CAN DISCARDED	RH	25.01.19	Mech Elec	
R5-10			USN5-175	TV-29 AV-27		SADM-REQ-00600		<p>The SADM shall be compatible with qualification temperatures specified below:</p> <p>Temperature range: Non-operating Operating min max min max</p> <p>1. S/C conductive interface (SADM TRP) -40 105 -20 70 2. S/C radiative interface -20 70 -20 75 3. S/C connectors -40 105 -20 95 4. SA conductive interface -40 125 -40 95 5. SA connectors -40 105 -35 95</p> <p>All values are in degrees per celcius</p>	Can	RH	25.01.19	Mech Elec	

ID	Rev	Successor	UserStory-ID	V.-ID	Owner	WR-ID	SADM/DS	Description	Priority	Sign	Date	Responsible	Parent
R5-11			USN5-176	AV-20	KSS	SADM-REQ-00620		The total heat flow to Spacecraft (S/C) including <ul style="list-style-type: none"> - The radiative heat flow to the S/C internal environment - The conductive heat flow to the S/C payload panel - The conductive heat flow to the S/C harness connector, shall be in the following range (positive values are from the SADM to the S/C): <ul style="list-style-type: none"> - Hot case: $-5W < HF < 90W$ (TBC) - Cold case: $-5W < HF < 90W$ (TBC) 	Can	EF	25.01.19	Mech	
R5-12			USN5-177	AV-21	KSS	SADM-REQ-00630		The total heat flow to SA including <ul style="list-style-type: none"> - The radiative heat flow to the SA internal environment - The conductive heat flow to the SA payload panel - The conductive heat flow to the SA harness connector shall be in the following range (positive values are from SA to the SADM): <ul style="list-style-type: none"> - Hot case: $-5W < HF < 5W$ (TBC) - Cold case: $-5W < HF < 5W$ (TBC) 	Can	EF	25.01.19	Mech	
R5-13			USN5-178	AV-22	KSS	SADM-REQ-00640		The SADM should be coupled to the Solar Array yoke trough a thermal washer. The washer should provide a thermal coupling of maximum 0.1W/K (TBC). Note: Relevant if we are to perform a thermal analysis	Can	EF RH	25.01.19 06.02.19	Mech	
R5-14			USN5-179	AV-23	KSS	SADM-REQ-00650		The SADM shall be compatible with thermal conductance between SADM and spacecraft conductive interface up to 6 W/K	Can	EF RH	25.01.19 06.02.19	Mech	
R5-15			USN5-180	TV-30	KSS	SADM-REQ-00660		Metallic parts of each electrical equipment chassis (case) shall be mutually bonded together by direct metal contact (preferred method). Bonding interfaces shall be designed to achieve contact resistance as follow: <ul style="list-style-type: none"> - < 2.5 mOhm for 1 junction - < 4.3 mOhm for 2 junctions - < 5.9 mOhm for 3 junctions - < 7.5 mOhm for 4 junctions - < 9.0 mOhm for 5 junctions - < 10.6 mOhm for 6 junctions 	Should	EF	25.01.19	Mech Elec	
R5-16			USN5-181	TV-31	KSS	SADM-REQ-00670		All external metallic parts without area consideration (such as metallic labels, baseplates, straps, insulated electrical circuits etc) and intrinsically conductive parts (such as carbon) that do not perform any electrical function shall be grounded to the equipment structure by a DC resistance of less than 1kOhm. Floating metallic parts are strictly prohibited without any area consideration	Can	EF	25.01.19	Elec	
R5-17			USN5-182	RV-18	KSS	SADM-REQ-00680		The bonding test shall be performed with the SADM switched off. The measurements shall be done with the four-wire method and 1A and with both directions of polarity, ref §5.3.10 [SD3].	Should	EF	25.01.19	Elec	

ID	Rev	Successor	UserStory-ID	V.-ID	Owner	WR-ID	SADM/DS	Description	Priority	Sign	Date	Responsible	Parent
R5-18			USN5-183	RV-19	KSS	SADM-REQ-00690		The bonding test shall be performed without any harness connection.	Should	EF	25.01.19	Elec	
R5-19			USN5-184	Rv-20	KSS	SADM-REQ-00810		The SADM shall ensure by design the required shielding according to EEE parts sensitivity (TIDL & TNIDL)	Must	EF	25.01.19	Elec	
R5-20			USN5-185	RV-21	KSS	SADM-REQ-00820		For materials, the impact and effect of electron,proton and gamma radiation shall be taken into account in the material degradation assessment	Must	EF	25.01.19	Mech	
R5-21			USN5-150	RV-22	KSS	SADM-REQ-00830		Assuming the SADM being an internal equipment, external materials shall withstand a TID level of 10 Mrad.	Must	EF	25.01.19	Mech	
R5-22			USN5-103		PG		DS	The Diagnostics Systems shall not interfere with the SADM.	Must	JS	06.02.19	Data	

ID	Rev	Successor	UserStory-ID	V.-ID	Owner	WR-ID	SADM/DS	Description	Priority	Sign	Date	Responsible	Parent
R6-01			USN5-187	RV-23	KSS	SADM-REQ-00840	SADM	The materials should be selected in the line with ECSS-Q-ST-70C ([SD5]), ECSS-Q-ST-70-71C ([SD6]) and ECSS-E-ST-32-08C ([SD7]) to ensure vacuum compability.	Should	EF	25.01.19	Mech	
R6-02			USN5-188	RV-24	KSS	SADM-REQ-00850	SADM	Materials should be selected in conformance with Table 5-1 in ECSS-Q-ST-70C ([SD5]) for bimetallic compability.	Should	EF	25.01.19	Mech	
R6-03			USN5-149	RV-25	KSS	SADM-REQ-00860	SADM	The SADM needs to include preferred venting path to allow for the typical depressurization profiles.	Can Must	EF RH	25.01.19 06.02.19	Mech	
R6-04			USN5-189	AV-28	KSS	SADM-REQ-00865	SADM	The impact of radiation shall be taken into consideration when selecting materials. Selected materials shall be able to withstand the radion levels specified for the orbit and lifetime of the SADM (ref. radiation requirements in §4.5.5 and lifetime requirement SADM-REQ-00970 in §4.8).	Must	EF	25.01.19	Mech	

ID	Rev	Successor	UserStory-ID	V.-ID	Owner	WR-ID	SADM/DS	Description	Priority	Sign	Date	Responsible	Parent
R8-01			USN5-135	A, T		SADM-REQ-00970	SADM	<p>The SADM shall have a minimum operational life time capability of</p> <p>- 5 years in space for 600km SSO(Sun-synchronous orbit) application</p> <p>13 000 cycles (life time factors according to ECSS-E-ST-33-01 [SD1] can be applied for qualification</p>	SHOULD	RH	07.02.19	<p>Mech</p> <p>Elec</p> <p>Data</p>	

ID	Rev	Successor	UserStory-ID	V.-ID	Owner	WR-ID	SADM/DS	Description	Priority	Sign	Date	Responsible	Parent
R7-01			USN5-194	AV-29	KSS	SADM-REQ-00870		The SADM shall be compliant with a reliability requirement of 0,9 for its operational lifetime. This reliability figure is applicable to EEE parts and mechanical rotational parts	Can	RH	25.01.19	Mech Elec	
R7-02			USN5-195	RV-26	KSS	SADM-REQ-00880		The SADM EEE shall be derated using the derating rules specified in ECSS-Q-ST-30-11C [SD8]	Should	RH	25.01.19	Elec	
R7-03			USN5-148	AV-30	KSS	SADM-REQ-00890		The SADM power transfers shall meet with the derating performance in ECSS-Q-ST-30-11C [SD8] for nominal operation	Should	RH	25.01.19	Mech Elec	
R7-04			USN5-196	AV-31	KSS	SADM-REQ-00900		The SADM power transfers shall meet the derating performance in ECSS-Q-ST-30-11C [SD8] in failure case (loss of one single wire, see definition below) Definition of failure case: If only one wire is lost, the increase in current applies only for one wire in the bundle and the margin is considered acceptable as long as the current is below single wired de-rated capacity.	Should	RH	25.01.19	Elec	
R7-05			USN5-140	AV-32	KSS	SADM-REQ-00930		The SADM driveline shall be single point failure free except for accepted mechanical parts	should	RH	25.01.19	Mech	
R7-06			USN5-226	TV-45	PG		DS-HW	The DS HWL must incorporate timeouts in the handshake protocol	Must	JS	06.03.19	Data	
R7-07			USN5-226	TV-46	PG		DS-HW	The DS HWL must have functionality for Negative Acknowledgement (NACK) in case of error in communication	Must	JS	06.03.19	Data	
R7-08			USN5-257	TV-48	PG		DS-HW	The DS HWL must incorporate error handling where it's needed.	Must	JS	17.03.19	Data	
R7-09			USN5-250	IV-18	PG		DS-HW	The final DS HWL shall be as effective as possible in the sens of proper design and having no redundant code.	Must	JS	17.03.19	Data	

Bachelor of Engineering

Project appendix

Winter semester 2019

VMD Description

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Ming Kit Wong
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document describes the Verification Management Document, and how to read it.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

Procedure for how and what to register at the *Verification management document*

Definitions by [European Cooperation of Space Standardization](#)

Verification is defined in ECSS-S-ST-00-01C as *the process which demonstrates through the provision of objective evidence that the product is designed and produced according to its specifications and the agreed deviations and waivers, and is free of defects.*

Validation is defined in ECSS-S-ST-00-01C as *process which demonstrates that the product is able to accomplish its intended use in the intended operational environment.*

Requirement is defined in ECSS-S-ST-00-01C as *documented demand to be complied with.*

Abstract

To check that a system complies with the given requirements, the group will perform specific measurements to get acceptance and proof of design. This is known as verification process.

The purpose of the [Verification management document](#) (VMD) is to function as an overview of how the group will verify the requirements from the [Requirement sheet](#). The VMD is an important part of traceability, project planning and supervising. It is therefore crucial that the VMD is followed up and filled out correctly.

This document will explain what the VMD contains and how to use it, and is intended for stakeholders and for users of the VMD.

Responsibilities

Everyone in the project team is responsible for creating and performing verification. It's the creator's responsibility to specify what standards that needs to be followed, if that is stated in the requirement or applicable for the current verification.

The one **who is performing** the verification is responsible for following the standards, if that is stated in the description. The performer is also responsible for changing the status or follow up the verification after the measurement is done.

The responsibilities of the **lead test engineer** is to ensure that all requirements will be verified during the project and that the test plan will be followed. It is also the lead test engineer's responsibility to ensure that the VMD is correctly filled and to keep the stakeholders updated.

Verification methods

Verification method	V.-ID	Initials & date	Requirement to verify
Inspection	IT-01	MKW 25.01.2019	R1-01.01

Figure 1: First part of the VMD

Verification of requirements can be done with different methods. The first thing to choose when creating a verification is to specify what kind of method is to be used. Verification methods are sorted into these five following categories:

Test

These are tests that demonstrates functions and performance of the system and is the most reliable way to verify the product. This could be functional testing or performance testing in worst case conditions or measurement of voltage and etc. in the system.

Analysis

This is verification done by finite element method (FEM), simulations, calculations or other sorts of empirical evaluation. This is important when testing is not possible.

Review

Reviewing could be to have someone to proofread document(s) or double checking drawings by team member or a supervisor who are not involved in the work of making the document/drawing.

Inspection

Visually inspect that the hardware are according to drawings or description in documents. This may be assembly of prototype or circuits, for example.

Unit testing

Unit testing is a method for testing one specific part of a code in a software.

Verification-ID

The structure of the Verification-ID is [first letter of the method + V] - [the number of verification], in example TV-01 for the first validation with testing, RV-03 for the third validation with review. To find the next number in the current category of methods, users are strongly recommended to use the filter function in Google Sheet.

Initials & date

Initials of the creator of the current verification and the date that the verification was registered. Dates shall be written in the format dd.mm.yyyy

Type of requirement to verify

The structure of a requirement-ID is built with “the type of requirement” - “the number of requirement”, in example R1-01.

This column is to state what kind of requirement is to be verified, in example R1 for Equipment breakdown, R2 for Functional requirement etc. This column separates the requirement ID to make the sheet more user friendly when the user like to sort out the verification by the type of requirement.

The color of the cell is indicating the priority of the requirement, where the color coding is yellow for **MUST**, cyan for **SHOULD** and white for CAN. This is also to indicate the importance of the verification.

Requirement to verify	Requirement	Description of verification method	Due date
R1-01.01	The SADM shall link the solar array wing mechanically to the spacecraft structure during launch and all operational phases.	Ensure that the assembly is done correctly according to drawings	ASAP

Figure 2: Second part of the VMD

Requirement

This is the description of the requirement to be verified. The text shall be exact as it is stated in the Requirement sheet. To ensure that the requirement text will be updated when changes are done on the Requirement Sheet, following shall be added in the requirement cell:

```
=IMPORTRANGE("https://docs.google.com/spreadsheets/d/1sxhgZIxREtATWt48BzuWZCH-MJr8LqtyCqZKElUhgW/edit", "[name of the sheet from Requirement sheet]![the cell with the requirement]")
```

In example:

```
=IMPORTRANGE("https://docs.google.com/spreadsheets/d/1sxhgZIxREtATWt48BzuWZCH-MJr8LqtyCqZKElUhgW/edit", "R2-Functional Requirements!H2")
```

In this case, the creator wants to get requirement from the sheet R2-Functional Requirements and from the cell H2. If there is a need to get requirements from a cell range, the syntax would be [H2:Hx] where x is the number of the last cell to be added.

Description of verification method

Test description is written by the one who creates the validation. This cell shall describe how the verification method should be implemented and any applicable test specification shall be stated here. Necessary standards shall also be mentioned here.

Due date	Responsible for verification	Status	Test report available	Done by and date
ASAP	Mechanical / MKW	Not tested	No	MKW 27.01.2019

Figure 3: The last part of the VMD

Due date

Due date could be a specific date, as soon as possible (ASAP shown in yellow), as late as possible (ALAP) or a phase in the project, like *during the design phase* or *during integration testing*. If a specific date is typed in, the cell will automatically turn yellow one week before due date. The cell will automatically turn red if the verification is not performed after due date.

Responsible for the verification

The person or subgroup who is responsible and/or assigned to, is stated here.

Status

The only available options for status are Passed, Failed or Not tested. Passed is equivalent with acceptance. If a verification fails, a new verification needs to be created for a new attempt for acceptance. See the section for Verification-ID for how to give a new identity to following attempts. @

Test report available

Some tests generates test reports or should be documented by an report. This is not required, but if there is an report available, it should be found in Google drive - Bachelor 2019 - Verification management - Test reports.

Test report number

The name of the test report is to be stated here.

Done by and date

This is the signature of the performer and the date the verification was done. Dates shall be written in the format dd.mm.yyyy

Additional information

- Stakeholders and users of the VMD are recommended to use the filter function



Bachelor of Engineering

Project appendix

Winter semester 2019

Verification Management Document

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	SatLight
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains the Verification Management Document - VMD. The requirements are extracted straight from the Requirement sheet for traceability.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

Verification method	V.-ID	Initials & date	REQ type	REQ-number	Requirement	Description of verification method	Due date	Responsible for verification	Status	Test report available	Test report	Done by and date	Comments
Inspection	IV-01	MKW 25.01.2019	R1	-01	The SADM will be composed of the following componets: - Drive line: Gear, motor - Main shaft and bearings - Housing - Signal and power transfer unit	Ensure that the assembly is done correctly according to drawings	ASAP	Mechanical / MKW	Passed	Yes	Report	Mech	See chapter of Mark3 in report
Review	RV-01	MKW 25.01.2019	R1	-01	The SADM will be composed of the following componets: - Drive line: Gear, motor - Main shaft and bearings - Housing - Signal and power transfer unit	Review that all the components are included in the drawings		Mechanical	Passed	Yes	Report	Mech	See chapter of Mark3 in report
Analysis	AV-01	MKW 26.01.2019	R2	-01	The SADM shall link the solar array wing mechanically to the spacecraft structure during launch and all operational phases	Use FEM to simulate the worst possible environment		Mechanical	Discarded				
Test	TV-01	MKW 26.01.2019	R2	-02	The SADM shall orient the solar arrays as specified herein	Use Diagnostic System for functional testing		Software	Discarded				
Test	TV-02	MKW 26.01.2019	R2	-03	The SADM shall hold the solar array wings in position when not powered	Take measurement of the position with sensors or Diagnostic System		Mechanical / Electrical	Failed	No			
Test	TV-03	MKW 26.01.2019	R2	-04	The SADM shall provide angular position information The DS shall provide angular position information of the SADM.	Take measurement of the position with sensors or Diagnostic System		Software / Electrical	Passed	No			
Inspection	TV-04	MKW 25.01.2019	R2	-05	The SADM shall transfer solar array grounding lines			Electrical	Passed				
Test	TV-05	MKW 26.01.2019	R2	-06	The DS shall imitate the spacecraft maincomputer			Software	Discarded				
Test	TV-06	MKW 26.01.2019	R2	-07	The DS shall provide information about the SADM, such as; - Input values - Output values				Discarded				
Review	RV-02	MKW 26.01.2019	R2	-08	The SADM physical reference frame shall be as defined below and depicted in following figure: - The origin shall be at the intersection of the SADM mounting plane with the S/C interface - The X-axis shall be in point - The Y-axis shall be coincident - The Z-axis shall be in the mounting plane and point through a mounting point	The designers will take that in accounting when modelling. This will be verified by review of drawings		Mechanical	Discarded				
Analysis	AV-02	MKW- 26.01.2019 RH 07.02.2019	R2	-09	The SADM shall be compliant with the solar array wing inertia as follows: - Ixx = up to 0.013 kgm ² - Iyy = up to 15 kgm ² - Izz = up to 15 kgm ² Note: The values for the solar array Moment of Inertia are provided at the SADM Interface	FEM analysis, calculations		Mechanical	Passed				Calculations, see moment of inertia
Test	TV-07	MKW 25.01.2019	R2	-10	The SADM functional performance and motorization margin shall be determined considereing the following solar array characteristics when attached to the SADM: - 1st torsional mode of the SA: 0,2 Hz with inertia of 31 kgm ² - 2nd torsional mode of the SA: 0,7 Hz with inertia of 30 kgm ² - 3rd torsional mode of the SA: 3,7 Hz with inertia of 33 kgm ² - Considered SA modal dampning 0,3% or Q = 166 - SA residual inertia: 16 kgm ² - Mass 110 kg +/- 5% - Mass: 0.40kg +/- 5%				Failed				Test results failed, see vibrations in FEM in the report
Test	TV-08	MKW 26.01.2019	R2	-11	The SADM shall be able to oscillate or rotate around the X-axis in both directions (CW or CCW), with a total angular range of minimum +/- 180 degrees				Not tested				Flex needs to be accurate
Analysis	AV-03	MKW 26.01.2019	R2	-12	The SADM shall satisfy motorization margin requirement as specified in §4.7.5.3 per ECSS-E-ST-33-01, [SD1]			Elec	Discarded				Motor needs to be space grade to satisfy this requirement
Test	TV-09	MKW 26.01.2019	R2	-13	The unpowered holding torque of the SADM shall be greater than or equal to 2 (TBC) Nm			Elec	Discarded				

Verification method	V.-ID	Initials & date	REQ type	REQ-number	Requirement	Description of verification method	Due date	Responsible for verification	Status	Test report available	Test report	Done by and date	Comments
Test	TV-10	MKW 26.01.2019	R2	-14	The SADM shall be able to deliver a minimum holding torque of 8 Nm with a motor running at a driving current of 150 mA The SADM shall be able to deliver a minimum holding torque of 0.04 Nm with a motor running at a driving current of 1 A			Elec	Not tested				
Test	TV-11	MKW 26.01.2019	R2	-15	The SADM irreversibility quasi-static torque shall be less than 70 (TBC) Nm Note: it means that if the quasi-static torque exceeds this maximum value, the SADM must rotate in order to avoid any degradation				Passed	No			
Test	TV-12	MKW 25.01.2019	R2	-16	Frequency Range on ground (Hz): - 0,1 - 1 - 1 - 10 - 10 - 200 - 200 - 1000 Max disturbance moment(Nm) - 0,001 - 0,01 - 0,1 - 0,1 All forces and torques are peak-to-peak at SADM/Spacecraft interface, taking into account all phenomena (step, acceleration, backlash)				Discarded				
Test	TV-13	MKW 26.01.2019	R2	-17	This is NA The SADM shall have enough torque margins to withstand a short circuit of motor winding without loss of functionality at 0,1 degrees/s (nominal speed) Note: using the redundant interface in case of failure in nominal winding and vice versa				Discarded				
Analysis	AV-04	MKW 26.01.2019	R2	-18	The SADM shall keep its position under the following spacecraft rotational accelerations: - Powered: +- 0,02 rad/s^2 (TBC) - Unpowered: +- 0,01 rad/s^2 (TBC) An amplification factor of 2 shall be considered				Discarded				
Test	TV-14	MKW 26.01.2019	R2	-19	The SADM shall be able to operate in normal angular speed mode between 0,0 X degrees/s to 5 degrees/s. The maximum output angular velocity capability shall be at least 3 degrees/s				Passed	No			
Test	TV-15	MKW 26.01.2019	R2	-20	The SADM angular speed at the output shaft shall have a stability performance better than: - 30 arcsec/s for frequency lower than 0,5 Hz - 0,5 arcsec/s in the frequency range from 0,5 Hz to 3 Hz - 1000 arcsec/s for frequency higher than 3 Hz				Discarded				
Analysis	AV-05	MKW 25.01.2019	R2	-21	The SADM shall provide a full step resolution at output shaft <1 degrees				Passed	No			
Test	TV-16	MKW 26.01.2019	R2	-22	The SADM shall provide an axial translation stiffness of >- 1.0 x 10^8 N/m				Discarded				
Test	TV-17	MKW 26.01.2019	R2	-23	The SADM shall provide a radial translation stiffness of >- 1.0 x 10^8 N/m with the solar array interface constrained in cross axis bending (interface plane remains parallel under loading)				Discarded				
Test	TV-18	MKW 26.01.2019	R2	-24	The SADM shall provide a cross axis bending stiffness of >- 1.0 X 10^5 Nm/rad				Discarded				
Test	TV-19	MKW 26.01.2019	R2	-25	The SADM shall provide a torsional stiffness of >_ 10^4 Nm/rad (assuming a blocked motor shaft, at 5 to 15 Nm external load on the output shaft) In the SADM backlash angular range (SADM-REQ-00170) with low loading (only overcoming frictional torque) an area with "zero stiffness" is expected and therefore not considered as a deviation to the requirement				Discarded				

Verification method	V.-ID	Initials & date	REQ type	REQ-number	Requirement	Description of verification method	Due date	Responsible for verification	Status	Test report available	Test report	Done by and date	Comments
Test	TV-20	MKW 25.01.2019	R2	-26	The SADM shall have a first Eigen-frequency > 250 Hz when its stator part is mounted on rigid interfaces and when its rotor part is connected to a dummy mass of minimum 2,5 kg with CoG placed at minimum 20 mm from the SADM I/F				Discarded				
Analysis	AV-06	MKW 26.01.2019	R3	-01	The mass of the SADM shall not exceed 2kg (Harness not included)	Find the total weight on a CAD with proper space material		Mechanical	Passed	Yes	Report		
Test	TV-21	MKW 26.01.2019	R3	-01	The mass of the SADM shall not exceed 2kg (Harness not included)	Put the prototype on a weight and see that it's not exceeding 2kg		Mechanical	Not tested				
Review	RV-03	MKW 26.01.2019	R3	-02	The SADM shall not exceed an envelope of 100 X 100 X 200 (millimeters)	Review drawings from CAD that the SADM does not exceed 100x100x200 mm		Mechanical	Passed				
Test	TV-22	MKW 26.01.2019	R3	-02	The SADM shall not exceed an envelope of 100 X 100 X 200 (millimeters)	Use a measuring tape on the engineering model to ensure that the SADM does not exceed 100x100x200 mm		Mechanical	Passed				
Analysis	AV-07	MKW 26.01.2019	R2	-10	The SADM functional performance and motorization margin shall be determined considering the following solar array characteristics when attached to the SADM: - 1st torsional mode of the SA: 0,2 Hz with inertia of 31 kgm ² - 2nd torsional mode of the SA: 0,7 Hz with inertia of 30 kgm ² - 3rd torsional mode of the SA: 3,7 Hz with inertia of 33 kgm ² - Considered SA modal dampning 0,3% or Q = 166 - SA residual inertia: 16 kgm ² - Mass 110 kg +/- 5% - Mass: 0.40kg +/- 5%				Not tested				
Review	RV-04	MKW 26.01.2019	R2	-11	The SADM shall be able to oscillate or rotate around the X-axis in both directions (CW or CCW), with a total angular range of minimum +/- 180 degrees				Not tested				
Analysis	AV-08	MKW 26.01.2019	R2	-13	The unpowered holding torque of the SADM shall be greater than or equal to 2 (TBC) Nm				Not tested				
Analysis	AV-09	MKW 26.01.2019	R2	-14	The SADM shall be able to deliver a minimum holding torque of 8 Nm with a motor running at a driving current of 150 mA The SADM shall be able to deliver a minimum holding torque of 0.04 Nm with a motor running at a driving current of 1 A				Not tested				
Analysis	AV-10	MKW 26.01.2019	R2	-15	The SADM irreversibility quasi-static torque shall be less than 70 (TBC) Nm Note: it means that if the quasi-static torque exceeds this maximum value, the SADM must rotate in order to avoid any degraation				Discarded				
Analysis	AV-11	MKW 26.01.2019	R2	-17	This is NA The SADM shall have enough torque margins to withstand a short circuit of motor winding without loss of functionality at 0,1 degrees/s (nominal speed) Note: using the redundant interface in case of failure in nominal winding and vice versa				Not tested				
Analysis	AV-12	MKW 26.01.2019	R2	-19	The SADM shall be able to operate in normal angular speed mode between 0,0 X degrees/s to 5 degrees/s. The maximum output angular velocity capability shall be at least 3 degrees/s				Not tested				
Analysis	AV-13	MKW 26.01.2019	R2	-22	The SADM shall provide an axial translation stiffness of >- 1.0 x 10 ⁸ N/m				Not tested				
Analysis	AV-14	MKW 26.01.2019	R2	-23	The SADM shall provide a radial translation stiffness of >- 1.0 x 10 ⁸ N/m with the solar array interface constrained in cross axis bending (interface plane remains parallel under loading)				Not tested				
Analysis	AV-15	MKW 26.01.2019	R2	-24	The SADM shall provide a cross axis bending stiffness of >- 1.0 X 10 ⁵ Nm/rad				Not tested				

Verification method	V.-ID	Initials & date	REQ type	REQ-number	Requirement	Description of verification method	Due date	Responsible for verification	Status	Test report available	Test report	Done by and date	Comments
Analysis	AV-16	MKW 26.01.2019	R2	-25	The SADM shall provide a torsional stiffness of $> 10^4$ Nm/rad (assuming a blocked motor shaft, at 5 to 15 Nm external load on the output shaft) In the SADM backlash angular range (SADM-REQ-00170) with low loading (only overcoming frictional torque) an area with "zero stiffness" is expected and therefore not considered as a deviation to the requirement				Not tested				
Analysis	AV-17	MKW 25.01.2019	R2	-26	The SADM shall have a first Eigen-frequency > 250 Hz when its stator part is mounted on rigid interfaces and when its rotor part is connected to a dummy mass of minimum 2,5 kg with CoG placed at minimum 20 mm from the SADM I/F				Not tested				
Review	RV-05	MKW 28.01.2019	R4	-01	The SADM mechanical interfaces to S/C and Solar Array shall be according to the SADM supplier				Passed				
Review	RV-06	MKW 28.01.2019	R4	-02	The zero reference position shall be clearly marked on the SADM rotor and stator and included on the Interface Control Drawing				Discarded				
Review	RV-07	MKW 28.01.2019	R4	-03	The length of the rotor harness shall be 500 mm flying leads				Passed				
Review	RV-08	MKW 28.01.2019	R4	-04	There shall be integrated connectors on the stator side, connector size and type are to be determined by SADM supplier				Passed	Yes			See Dsub Wall in SADM Design
Test	TV-23	MKW 28.01.2019	R4	-05	The maximum total current transfer in the power lines shall be 20A forward and 20A return; - 10A each direction each solar array				Passed				
Review	RV-09	MKW 28.01.2019	R4	-06	Electrical grounding between SADM shaft and housing shall be redundant				Failed				
Review	RV-10	MKW 28.01.2019	R4	-07	Solar Array grounding lines shall be insulated from SADM grounding.				Passed				
Review	RV-11	MKW 28.01.2019	R4	-08	The stepper motor electrical shall have the following nominal electrical interface: - 2 phase, bipolar communication with 1.8 degrees full step angle - Phase resistance: 18.5 Ohm +/- 10% - Phase inductance: 37mH +/- 20% - Power: < 5W The stepper motor electrical shall have the following nominal electrical interface: - 2 phase, bipolar communication with 1.8 degrees full step angle - Phase resistance: 3.5 ohm/phase - Phase inductance: 1.2 mH/phase				Passed				
Review	RV-12	MKW 28.01.2019	R4	-09	The motor interface shall be according to standard NEMA dimensions.				Failed				
Test	TV-24	MKW 28.01.2019	R5	-11	The total heat flow to Spacecraft (S/C) including - The radiative heat flow to the S/C internal environment - The conductive heat flow to the S/C payload panel - The conductive heat flow to the S/C harness connector, shall be in the following range (positive values are from the SADM to the S/C): - Hot case: $-5W < HF < 90W$ (TBC) - Cold case: $-5W < HF < 90W$ (TBC)				Discarded				
			R6	-12									
			R7	-13									
			R8	-14									
			R9	-15									
			R10	-16									
Inspection	IV-02	MKW 28.01.2019	R4	-01	The SADM mechanical interfaces to S/C and Solar Array shall be according to the SADM supplier				Passed				

Verification method	V.-ID	Initials & date	REQ type	REQ-number	Requirement	Description of verification method	Due date	Responsible for verification	Status	Test report available	Test report	Done by and date	Comments
Inspection	IV-03	MKW 28.01.2019	R4	-02	The zero reference position shall be clearly marked on the SADM rotor and stator and included on the Interface Control Drawing				Failed				
Inspection	IV-04	MKW 28.01.2019	R4	-03	The length of the rotor harness shall be 500 mm flying leads				Passed				
Inspection	IV-05	MKW 28.01.2019	R4	-04	There shall be integrated connectors on the stator side, connector size and type are to be determined by SADM supplier				Passed				
Analysis	AV-18	MKW 28.01.2019	R4	-05	The maximum total current transfer in the power lines shall be 20A forward and 20A return; - 10A each direction each solar array				Passed				
Test	AV-19	MKW 28.01.2019	R4	-11	The SADM power consumption shall not exceed 24 Watts.	This is verified by measure the current flow and voltage at the highest angular velocity.			Not tested				
Review	RV-13	MKW 28.01.2019	R5	-01	The SADM shall be designed to meet the following requirements for temperature under the following environmental conditions: - On-Ground: 23 +/- 5 degrees - Launch: 20 +/- 5 degrees - In-orbit: see SADM-REQ-00610 - Note: This requirement does not exist. - In orbit: See SADM-REQ-00600	FEM analysis			Passed				
Review	RV-14	MKW 28.01.2019	R5	-02	The SADM shall be designed to meet the following requirements for pressure under the following environmental conditions: - On-Ground: P Atmos - Launch: P Atmos to 10 ⁻⁵ mbar - In-orbit: 10 ⁻⁵ mbar				Discarded				Not prioritized. Needs vacuum chamber or more powerful CAD tool
Review	RV-15	MKW 28.01.2019	R5	-03	The SADM shall be designed to meet the following requirements for humidity under the following environmental conditions: - On-Ground: 30% < Hr < 65% - Launch: 0% < Hr < 65% - In-orbit: Vacuum				Discarded				Not prioritized. Needs clean room assembling
Review	RV-16	MKW 28.01.2019	R5	-04	The SADM structural analysis shall take into account the margins and factors described in §4.7.5.2 of ECSS-E-ST-33-01 [STD1]				Passed				SolidWorks analysis, SADM won't buckle
Review	RV-17	MKW 28.01.2019	R5	-05	The SADM and GSE design shall allow mounting of accelerometers (flat areas aligned with the coordinate system). Note: Not necessary for prototype, but for "Test Prediction Analysis" we should define where the output comes from				Discarded				
Test	TV-25	MKW 28.01.2019	R5	-06	The SADM shall be able to withstand the following quasi-static loads applied in the SADM solar array interface and with the SADM constrained by its normal screws in a rigid (infinitely stiff) spacecraft interface Maximum moment: Axial: +/- 0 N Radial: +/- 1500 N Bending: +/- 300 N Maximum radial and axial loads: Axial: +/- 2000 N Radial: +/- 1500 N Bending: +/- 260 Nm	FEM in SolidWorks		RH	Passed	Yes		06.05.19 RH	See the chapter for Finite Element Method in report

Verification method	V.-ID	Initials & date	REQ type	REQ-number	Requirement	Description of verification method	Due date	Responsible for verification	Status	Test report available	Test report	Done by and date	Comments
Analysis	AV-21	MKW 28.01.2019	R5	-12	The total heat flow to SA including - The radiative heat flow to the SA internal environment - The conductive heat flow to the SA payload panel - The conductive heat flow to the SA harness connector shall be in the following range (positive values are from SA to the SADM): - Hot case: $-5W < HF < 5W$ (TBC) - Cold case: $-5W < HF < 5W$ (TBC)				Discarded				Not prioritized
Analysis	AV-22	MKW 28.01.2019	R5	-13	The SADM should be coupled to the Solar Array yoke trough a thermal washer. The washer should provide a thermal coupling of maximum 0.1W/K (TBC). Note: Relevant if we are to preform a thermal analysis				Discarded				Not prioritized
Analysis	AV-23	MKW 28.01.2019	R5	-14	The SADM shall be compatible with thermal conductance between SADM and spacecraft conductive interface up to 6 W/K				Discarded				Outside scope, not prioritized
Test	TV-30	MKW 28.01.2019	R5	-15	Metallic parts of each electrical equipment chassis (case) shall be mutually bonded together by direct metal contact (preferred method). Bonding interfaces shall be designed to acheieve contact resistance as follow: - < 2.5 mOhm for 1 junction - < 4.3 mOhm for 2 junctions - < 5.9 mOhm for 3 junctions - < 7.5 mOhm for 4 junctions - < 9.0 mOhm for 5 junctions - < 10.6 mOhm for 6 junctions	Bonding test, 4 wire method.	22.05.19	Electrical/EF	Passed	Yes	Bonding test - TV-30, RV-18 and RV-19	EF HØ 20.05.19	
Test	TV-31	MKW 28.01.2019	R5	-16	All external metallic parts without area consideration (such as metallic labels, baseplates, straps, insulated electrical circuits etc) and intrinsically conductive parts (such as carbon) that do not perform any electrical function shall be grounded to the equipment structure by a DC resistance of less than 1kOhm. Floating metallic parts are strictly prohibited without any area consideration	No floating metallic parts inside SADM.			Passed	No	NA		
Review	RV-18	MKW 28.01.2019	R5	-17	The bonding test shall be performed with the SADM switched off. The measurements shall be done with the four-wire method and 1A and with both directions of polarity, ref §5.3.10 [SD3].				Passed	Yes			
Review	RV-19	MKW 28.01.2019	R5	-18	The bonding test shall be performed without any harness connection.				Passed	Yes			
Review	RV-20	MKW 28.01.2019	R5	-19	The SADM shall ensure by design the required shielding according to EEE parts sensitivity (TIDL & TNIDL)				Discarded				
Review	RV-21	MKW 28.01.2019	R5	-20	For materials, the impact and effect of electron,proton and gamma radiation shall be taken into account in the material degradation assessment	Research, and written documentation about the effects of radiation and other hazards in LEO		Mechanical / RH	Passed	Yes	In report	RH 18.03.2019	
Review	RV-22	MKW 28.01.2019	R5	-21	Assuming the SADM being an internal equipment, external materials shall withstand a TID level of 10 Mrad.								
Analysis	AV-24	MKW 28.01.2019	R5	-06	The SADM shall be able to withstand the following quasi-static loads applied in the SADM solar array interface and with the SADM constrained by its normal screws in a rigid (infinitely stiff) spacecraft interface Maximum moment: Axial: +- 0 N Radial: +- 1500 N Bending: +- 300 N Maximum radial and axial loads: Axial: +- 2000 N Radial: +- 1500 N Bending: +- 260 Nm				Passed				

Verification method	V.-ID	Initials & date	REQ type	REQ-number	Requirement	Description of verification method	Due date	Responsible for verification	Status	Test report available	Test report	Done by and date	Comments			
Analysis	AV-25	MKW 28.01.2019	R5	-07	The SADM shall be able to withstand the following sinusoidal vibration qualification levels for all three orthogonal axis. The sweep rate is specified by ECSS-E-ST-10-03C [SD2] All axis: Frequency (Hz): Acceleration (g): - 5 - Max shaker capacity (max 24g) - 22 - Max shaker capacity (max 24g) - 22 - 24 - 140 - 24				Discarded							
Analysis	AV-26	MKW 28.01.2019	R5	-08	The SADM shall be able to withstand the following random vibration environment in a single axis in any direction (X and Z-axis and out of plane Y-axis). The duration is specified by ECSS-E-ST-10-03C [SD2]. Notching is allowed in order to protect SADM sensitive components and can be incorporated with the customer's approval. Mechanical analysis shall set the basis for maximum allowed gRMS for sensitive components. Out of Plane: FREQ(Hz): ASD(g^2/Hz): Db/OCT: gRMS: - 20 - 0,1008 - * - 19,3 - 100 - 0,5000 - 3,00 - 19,3 - 400 - 0,5000 - 0,00 - 19,3 - 2000 - 0,0345 - (-5.00) - 19,3 In Plane FREQ(Hz): ASD(g^2/Hz): Db/OCT: gRMS: - 20 - 0,0422 - * - 12,5 - 100 - 0,2100 - 3,00 - 12,5 - 400 - 0,2100 - 0,00 - 12,5 - 2000 - 0,01450 - (-5.00) - 12,5							Failed				
Analysis	AV-27	MKW 28.01.2019	R5	-10	The SADM shall be compatible with qualification temperatures specified below: Temperature range: Non-operating Operating min max min max 1. S/C conductive interface -40 105 -20 70 (SADM TRP) 2. S/C radiative interface -20 70 -20 75 3. S/C connectors -40 105 -20 95 4. SA conductive interface -40 125 -40 95 5. SA connectors -40 105 -35 95 All values are in degrees per celcius											
Review	RV-23	MKW 28.01.2019	R6	-01	The materials should to be selected in the line with ECSS-Q-ST-70C ([SD5]), ECSS-Q-ST-70-71C ([SD6]) and ECSS-E-ST-32-08C ([SD7]) to ensure vacuum compability.			RH	Passed	Yes	Report		See chapter for Materials in report			
Review	RV-24	MKW 28.01.2019	R6	-02	Materials should be selected in conformance with Table 5-1 in ECSS-Q-ST-70C ([SD5]) for bimetallic compability.				Discarded							
Review	RV-25	MKW 28.01.2019	R6	-03	The SADM needs to include preferred venting path to allow for the typical depressurization profiles.				Discarded							
Analysis	AV-28	MKW 28.01.2019	R6	-04	The impact of radiation shall be taken into consideration when selecting materials. Selected materials shall be able to withstand the radion levels specified for the orbit and lifetime of the SADM (ref. radiation requirements in §4.5.5 and lifetime requirement SADM-REQ-00970 in §4.8).				Passed				Aluminum generally has high resistance against radiation			

Verification method	V.-ID	Initials & date	REQ type	REQ-number	Requirement	Description of verification method	Due date	Responsible for verification	Status	Test report available	Test report	Done by and date	Comments
Analysis	AV-29	MKW 28.01.2019	R7	-01	The SADM shall be compliant with a reliability requirement of 0,9 for its operational lifetime. This reliability figure is applicable to EEE parts and mechanical rotational parts				Discarded				
Review	RV-26	MKW 28.01.2019	R7	-02	The SADM EEE shall be derated using the derating rules specified in ECSS-Q-ST-30-11C [SD8]				Passed	Yes			See Design Description
Analysis	AV-30	MKW 28.01.2019	R7	-03	The SADM power transfers shall meet with the derating performance in ECSS-Q-ST-30-11C [SD8] for nominal operation				Passed	Yes			See Design Description
Analysis	AV-31	MKW 28.01.2019	R7	-04	The SADM power transfers shall meet the derating performance in ECSS-Q-ST-30-11C [SD8] in failure case (loss of one single wire, see definition below) Definition of failure case: If only one wire is lost, the increase in current applies only for one wire in the bundle and the margin is considered acceptable as long as the current is below single wired de-rated capacity.				Passed				
Analysis	AV-32	MKW 28.01.2019	R7	-05	The SADM driveline shall be single point failure free except for accepted mechanical parts				Discarded				NA
Test	TV-45	JS 07.03.19	R7	-06	The DS HWL must incorporate timeouts in the handshake protocol	Functional test: Force timeout and see that it works properly		Hardware / JS	Passed	Yes	VR-HWL-A2	Passed for Alpha 2) JS 07.03.2019	
Test	TV-46	JS 07.03.19	R7	-07	The DS HWL must have functionality for Negative Acknowledgement (NACK) in case of error in communication	Functional test: Force timeout and send faulty instructions to verify that a nack is sent		Hardware / JS	Passed	Yes	VR-HWL-A2	Passed for Alpha 2) JS 07.03.2019	
Inspection	IV-06	TM 06.02.2019	R2	-27	The DS Software Layer shall display diagnostics data in real time	Ensure that data is displayed on screen		Software / TM	Passed	Yes	VR-SWL-A1	TM 14.02.2019	
Analysis	AV-34	TM 06.02.2019	R2	-27	The DS Software Layer shall display diagnostics data in real time	Compare timestamp from when data is sent to when data is displayed		Software / TM	Discarded	Yes	VR-SWL-A1		
		TM 06.02.2019	R2	-28	The DS Software Layer shall have a modular design for easy addition of diagnostic components	?		Software / TM					
Unit test	UTV-1	TM 06.02.2019	R2	-29	The DS Software Layer shall save diagnostic data to a database	Unit Test: Assert database query success		Software / TM	Discarded	Yes	VR-SWL-A3	TM 12.05.2019	
Unit test	UTV-2	TM 06.02.2019	R2	-29	The DS Software Layer shall save diagnostic data to a database	Unit Test: Assert value saved equal to value input		Software / TM	Discarded	Yes	VR-SWL-A3	TM 12.05.2019	
Unit test	UTV-3	TM 06.02.2019	R4	-12	The DS Software Layer should have standarized interface for input data The DS Software Layer should use JSON for input data	Unit Test: Assert input format match	17.02.2019	Software / TM	Discarded	Yes	VR-SWL-A1	TM 14.02.2019	
Unit test	UTV-5	TM 06.02.2019	R4	-12	The DS Software Layer should have standarized interface for input data The DS Software Layer should use JSON for input data	Assert datatype and value as expected for: double	17.02.2019	Software / TM	Passed	Yes	VR-SWL-A1	TM 14.02.2019	
Unit test	UTV-4	TM 06.02.2019	R4	-13	The DS Software Layer should have standarized interface for output data The DS Software Layer should use JSON for output data	Unit Test: assert output format match		Software / TM	Discarded	Yes	VR-SWL-A1	TM 14.02.2019	
Unit test	UTV-7	TM 13.02.2019	R4	-13	The DS Software Layer should have standarized interface for output data The DS Software Layer should use JSON for output data	Assert datatype and value as expected for: double	17.02.2019	Software / TM	Discarded	Yes	VR-SWL-A1	TM 14.02.2019	
Inspection	IV-07	TM 06.02.2019	R4	-14	DS Software layer GUI shall support changing underlying COM port settings	Ensure that com settings are correct		Software / TM	Passed	Yes	VR-SWL-A2	TM 04.03.2019	
Test	TV-37	TM 06.02.2019	R4	-15	DS Software layer GUI shall support selection of diagnostic data output view	Functional test: View should display correctly after changed to desired output type		Software / TM	Passed	Yes	VR-SWL-A3	TM 12.05.2019	
Test	TV-38	TM 06.02.2019	R4	-16	DS Software layer GUI shall support displaying historical data	Functional test: Output should display historical data according to selected time settings		Software / TM	Passed	Yes	VR-SWL-A3	TM 12.05.2019	
Test	TV-39	JS 06.02.19	R2	- 30	The DS Hardware layer must be able to control the SADM for testing purposes.	Functional test: Run as many instructions as see fit, and confirm that the desired result is achieved.		Hardware / JS	Passed	Yes	VR-HWL-A2	Passed for Alpha 2) JS 07.03.2019	
Test	TV-40	JS 06.02.19	R2	- 31	The DS Hardware layer must be able to transmit sensor data to the Software layer for comparison and visualization purposes.	Functional test: Transmit as much sensor data as see fit, and confirm that the receiver is able to properly read the data.		Hardware / JS	Passed	Yes	VR-HWL-A1	Passed for Alpha 1) JS 07.03.2019	

Verification method	V.-ID	Initials & date	REQ type	REQ-number	Requirement	Description of verification method	Due date	Responsible for verification	Status	Test report available	Test report	Done by and date	Comments
Inspection	IV-08	JS 06.02.19	R4	- 17	DS Hardware layer shall have a standardized expandable Input handler, buffering input data according to input type. Rev. A) DS Hardware layer shall have a general input handler responsible for every kind of input.	Confirm that the code replicates the class diagrams and follow related standards.		Hardware / JS	Discarded	Yes	VR-HWL-A1	Passed for Alpha 1) JS 07.03.2019 Discarded for Alpha 2) JS 07.03.2019	See Alpha 1 Class Diagram
Inspection	IV-09	JS 06.02.19	R4	- 18	DS Hardware layer shall have a standardized expandable output handler translating the input data passed from a specific controller, and output the translated information to the device to be controlled. Rev. A) DS Hardware layer shall have a general output handler supporting the different protocols/formats needed in order to communicate with the different connected devices.	Confirm that the code replicates the class diagrams and follow related standards.		Hardware / JS	Discarded	Yes	VR-HWL-A1	Passed for Alpha 1) JS 07.03.2019 Discarded for Alpha 2) JS 07.03.2019	See Alpha 1 Class Diagram
Inspection	IV-10	JS 06.02.19	R4	- 19	There shall be an interface between the input handler and the output handler. Preferably one for each device to be controlled. Rev. A) There shall be an interface responsible for fetching from the input handler and passing to the output handler.	Confirm that the code replicates the class diagrams and follow related standards.		Hardware / JS	Discarded	Yes	VR-HWL-A1	Passed for Alpha 1) JS 07.03.2019 Discarded for Alpha 2) JS 07.03.2019	See Alpha 1 Class Diagram
Test	TV-41	JS-06.02-19 Rev. A) JS-21.02-19 Rev. B) JS 07.03.19	R4	- 17.01	The input handler of the DS Hardware layer must be compatible with the software layer of the diagnostic system. Rev. A) The input handler must support JSON input received from the software layer. Rev. B) The HWL must support JSON input received from the software layer.	Functional test: Send as many instructions as see fit from the software layer, and confirm that the data received is properly handled.		Hardware / JS	Passed	Yes	VR-HWL-A2	Passed for Alpha 2) JS 07.03.2019	
Test	TV-42	JS-06.02-19 Rev. A) JS-21.02-19 Rev. B) JS 07.03.19	R4	- 18.01	The output handler of the DS Hardware layer shall support the different protocols needed in order to communicate with the different connected devices. Rev. A) The output handler shall incorporate serial printing of JSON formatted data. Rev. B) The HWL shall incorporate serial printing of JSON formatted data.	Confirm that the code replicates the class diagrams and related standards. Addition in Rev. A) Check that the software layer receives the data as expected.		Hardware / JS	Passed	Yes	VR-HWL-A1	Passed for Alpha 1) JS 07.03.2019	
Inspection	IV-12	JS 06.02.19	R5	-22	The Diagnostics Systems shall not interfere with the-SADM.	Confirm that the DS and SADM is indeed two separate systems.		Hardware / JS	Discarded				
Inspection	IV-13	TM 07.02.19	R2	-32	The DS Software layer shall be able to calibrate position and speed of the SADM drive shaft	Ensure that the SADM correctly responds to calibration input		Software / TM	Passed	Yes	VR-SWL-A2	TM 07.03.2019	
Inspection	IV-14	TM 07.02.19	R4	-20	The DS Software Layer should provide an interface for adjusting calibration settings such as position and speed of the SADM motor	Ensure that interface records and delivers the right configuration information		Software / TM	Passed	Yes	VR-SWL-A2		
Unit test	UTV-24	TM 07.02.2019	R2	-33	The DS Software layer should be able to remember previous COM settings and try to establish a connection to HW layer with these	Unit Test: Assert writing to config file success		Software / TM	Passed	Yes	VR-SWL-B1	TM 08.04.2019	
Unit test	UTV-6	TM 07.02.2019	R2	-33	The DS Software layer should be able to remember previous COM settings and try to establish a connection to HW layer with these	Unit Test: Assert written config data equal to input		Software / TM	Passed	Yes	VR-SWL-B1	TM 08.04.2019	
Inspection	IV-15	TM 07.02.19	R2	-33	The DS Software layer should be able to remember previous COM settings and try to establish a connection to HW layer with these	Verify that correct config data is written to config file, and no other config data is overwritten		Software / TM	Passed	Yes	VR-SWL-B1	TM 08.04.2019	Achieved with functional testing instead of inspection
Unit test	UTV-8	TM 13.02.2019	R4	-12	The DS Software Layer should have standarized interface for input data The DS Software Layer should use JSON for input data	Assert datatype and value as expected for string	17.02.2019	Software / TM	Passed	Yes	VR-SWL-A2 VR-SWL-A1	TM 14.02.2019 TM 18.02.2019	Performed for version Alpha1 on 14.02. Performed for version Alpha2 on 18.02
Unit test	UTV-9	TM 13.02.2019	R4	-12	The DS Software Layer should have standarized interface for input data The DS Software Layer should use JSON for input data	Assert datatype and value as expected for float	17.02.2019	Software / TM	Passed	Yes	VR-SWL-A2 VR-SWL-A1	TM 14.02.2019 TM 18.02.2019	Performed for version Alpha1 on 14.02. Performed for version Alpha2 on 18.02
Unit test	UTV-10	TM 13.02.2019	R4	-12	The DS Software Layer should have standarized interface for input data The DS Software Layer should use JSON for input data	Assert datatype and value as expected for int	17.02.2019	Software / TM	Passed	Yes	VR-SWL-A2 VR-SWL-A1	TM 14.02.2019 TM 18.02.2019	Performed for version Alpha1 on 14.02. Performed for version Alpha2 on 18.02
Unit test	UTV-11	TM 13.02.2019	R4	-12	The DS Software Layer should have standarized interface for input data The DS Software Layer should use JSON for input data	Assert datatype and value as expected for long	17.02.2019	Software / TM	Passed	Yes	VR-SWL-A2 VR-SWL-A1	TM 14.02.2019 TM 18.02.2019	Performed for version Alpha1 on 14.02. Performed for version Alpha2 on 18.02

Verification method	V.-ID	Initials & date	REQ type	REQ-number	Requirement	Description of verification method	Due date	Responsible for verification	Status	Test report available	Test report	Done by and date	Comments
Unit test	UTV-12	TM 13.02.2019	R4	-12	The DS Software Layer should have standarized interface for input data The DS Software Layer should use JSON for input data	Assert datatype and value as expected for JArray	17.02.2019	Software / TM	Passed	Yes	VR-SWL-A2 VR-SWL-A1	TM 14.02.2019 TM 18.02.2019	Performed for version Alpha1 on 14.02. Performed for version Alpha2 on 18.02
Unit test	UTV-13	TM 13.02.2019	R4	-13	The DS Software Layer should have standarized interface for output data The DS Software Layer should use JSON for output data	Assert datatype and value as expected for: double	17.02.2019	Software / TM	Passed	Yes	VR-SWL-A2 VR-SWL-A1	TM 14.02.2019 TM 18.02.2019	Performed for version Alpha1 on 14.02. Performed for version Alpha2 on 18.02
Unit test	UTV-14	TM 13.02.2019	R4	-13	The DS Software Layer should have standarized interface for output data The DS Software Layer should use JSON for output data	Assert datatype and value as expected for string	17.02.2019	Software / TM	Passed	Yes	VR-SWL-A2 VR-SWL-A1	TM 14.02.2019 TM 18.02.2019	Performed for version Alpha1 on 14.02. Performed for version Alpha2 on 18.02
Unit test	UTV-15	TM 13.02.2019	R4	-13	The DS Software Layer should have standarized interface for output data The DS Software Layer should use JSON for output data	Assert datatype and value as expected for float	17.02.2019	Software / TM	Passed	Yes	VR-SWL-A2 VR-SWL-A1	TM 14.02.2019 TM 18.02.2019	Performed for version Alpha1 on 14.02. Performed for version Alpha2 on 18.02
Unit test	UTV-16	TM 13.02.2019	R4	-13	The DS Software Layer should have standarized interface for output data The DS Software Layer should use JSON for output data	Assert datatype and value as expected for int	17.02.2019	Software / TM	Passed	Yes	VR-SWL-A2 VR-SWL-A1	TM 14.02.2019 TM 18.02.2019	Performed for version Alpha1 on 14.02. Performed for version Alpha2 on 18.02
Unit test	UTV-17	TM 13.02.2019	R4	-13	The DS Software Layer should have standarized interface for output data The DS Software Layer should use JSON for output data	Assert datatype and value as expected for long	17.02.2019	Software / TM	Passed	Yes	VR-SWL-A2 VR-SWL-A1	TM 14.02.2019 TM 18.02.2019	Performed for version Alpha1 on 14.02. Performed for version Alpha2 on 18.02
Unit test	UTV-18	TM 13.02.2019	R4	-13	The DS Software Layer should have standarized interface for output data The DS Software Layer should use JSON for output data	Assert datatype and value as expected for generic list	17.02.2019	Software / TM	Discarded	Yes	VR-SWL-A2 VR-SWL-A1	TM 14.02.2019 TM 18.02.2019	Performed for version Alpha1 on 14.02. Performed for version Alpha2 on 18.02
Unit test	UTV-19	TM 13.02.2019	R4	-13	The DS Software Layer should have standarized interface for output data The DS Software Layer should use JSON for output data	Assert datatype and value as expected for key value pair with generic value	17.02.2019	Software / TM	Discarded	Yes	VR-SWL-A2 VR-SWL-A1	TM 14.02.2019 TM 18.02.2019	Performed for version Alpha1 on 14.02. Performed for version Alpha2 on 18.02
Unit test	UTV-20	TM 13.02.2019	R4	-13	The DS Software Layer should have standarized interface for output data The DS Software Layer should use JSON for output data	Assert datatype and value as expected for Hashtable	17.02.2019	Software / TM	Passed	Yes	VR-SWL-A2 VR-SWL-A1	TM 14.02.2019 TM 18.02.2019	Performed for version Alpha1 on 14.02. Performed for version Alpha2 on 18.02
Unit test	UTV-21	TM 13.02.2019	R4	-13	The DS Software Layer should have standarized interface for output data The DS Software Layer should use JSON for output data	Assert datatype and value as expected for JObject	17.02.2019	Software / TM	Passed	Yes	VR-SWL-A2 VR-SWL-A1	TM 14.02.2019 TM 18.02.2019	Performed for version Alpha1 on 14.02. Performed for version Alpha2 on 18.02
Unit test	UTV-22	TM 13.02.2019	R4	-12	The DS Software Layer should have standarized interface for input data The DS Software Layer should use JSON for input data	Assert correct values received when multiple subscribers subscribe on same stream to different keys	17.02.2019	Software / TM	Passed	Yes	VR-SWL-A2 VR-SWL-A1	TM 14.02.2019 TM 18.02.2019	Performed for version Alpha1 on 14.02. Performed for version Alpha2 on 18.02
Unit test	UTV-23	TM 13.02.2019	R4	-12	The DS Software Layer should have standarized interface for input data The DS Software Layer should use JSON for input data	Assert correct values received when multiple subscribers subscribe on same stream to same key	17.02.2019	Software / TM	Passed	Yes	VR-SWL-A2 VR-SWL-A1	TM 14.02.2019 TM 18.02.2019	Performed for version Alpha1 on 14.02. Performed for version Alpha2 on 18.02
Unit test	UTV-25	TM 13.02.2019	R4	-12	The DS Software Layer should have standarized interface for input data The DS Software Layer should use JSON for input data	Assert datatype and value as expected for JObject	17.02.2019	Software / TM	Passed	Yes	VR-SWL-A2 VR-SWL-A1	TM 14.02.2019 TM 18.02.2019	Performed for version Alpha1 on 14.02. Performed for version Alpha2 on 18.02
Inspection	IV-16	TM 04.03.2019	R2	-28	The DS Software Layer shall have a modular design for easy addition of diagnostic components	Code audit		Software / TM	Passed	Yes	VR-SWL-A2	TM 04.03.2019	
Inspection	IV-17	TM 04.03.2019	R4	-25	Shall follow a unified communication protocol	Code audit		Software / TM	Passed	Yes	VR-SWL-A2	TM 07.03.2019	
Test	TV-43	TM 04.03.2019	R4	-25	Shall follow a unified communication protocol	Functional test: System behaves as expected and follows defined protocol	04.03.2019	Software / TM / JS	Passed	Yes	VR-HWL-A2 VR-SWL-A2	TM 07.03.2019 JS 07.03.2019	
Test	TV-44	JS 06.03.2019	R2	-36	The DS Hardware layer shall be able to interpret and execute instructions received from the software layer.	Functional test: Run as many instructions as see fit, and confirm that the desired result is achieved.		Hardware / JS	Passed	Yes	VR-HWL-A2	Passed for Alpha 2) JS 07.03.2019	
Test	TV-47	JS 15.03.2019	R4	-26	The HWL should be able to enable or disable different sensors according to received instructions.	Functional test: Run as many instructions as see fit, and confirm that the desired result is achieved.		Hardware / JS	Passed	Yes	VR-HWL-B1	JS 15.04.2019	
Test	TV-48	JS 17.03.2019	R7	-08	The DS HWL must incorporate error handling where it's needed.	Functional test: Provoke errors to check if it is properly handled.		Hardware / JS	Passed	Yes	VR-HWL-A3	JS 15.04.2019	

Verification method	V.-ID	Initials & date	REQ type	REQ-number	Requirement	Description of verification method	Due date	Responsible for verification	Status	Test report available	Test report	Done by and date	Comments
Inspection	IV-18	JS 17.03.2019	R7	-09	The final DS HWL shall be as effective as possible in the sens of proper design and having no redundant code.	Inspect the class diagram to check for redundant code and changes that could be made to improve performance. If it looks good, the verification can be considered passed.		Hardware / JS	Passed	Yes	VR-HWL-B2	JS 06.05.2019	
Test	TV-49	TM 22.03.2019	R2	-29	The DS Software Layer shall save diagnostic data to a database	Functional test: Data is saved to database and is readable as values for data plot		Software / TM	Passed	Yes	VR-SWL-A3	TM 12.05.2019	
Analysis	TV-50	TM 06.02.2019	R2	-27	The DS Software Layer shall display diagnostics data in real time	Inspect live data values from live data table in the user interface	08.04.2019	Software / TM	Passed	Yes	VR-SWL-B1	TM 08.04.2019	
Unit test	UTV-26	TM 15.04.2019	R2	-34	DS SW should be able to configure desired maximum and minimum limits for operational sensor values to display status indicator on live input values during diagnostics run. Values outside accepted values should indicate deviation	Verify functionality for ObservableNumericValue and ObservableNumericValueCollection	15.04.2019	Software / TM	Passed	Yes	VR-SWL-B1-2	TM 15.04.2019	
Test	TV-51.A	TM 15.04.2019	R2	-34.01	DS SW should provide functionality to save diagnostics settings to a template	Functional test: Entry appears in UI after saving. Data is displayed in parameter control UI after loading. Correct values for minimum and maximum values appear after loading.	15.04.2019	Software / TM	Passed	Yes	VR-SWL-B1-2	TM 15.04.2019	
Test	TV-51.B	TM 15.04.2020	R3	-34.02	DS SW should be able to load and apply settings from a diagnostics settings template	Functional test: Entry appears in UI after saving. Data is displayed in parameter control UI after loading. Correct values for minimum and maximum values appear after loading.	15.04.2019	Software / TM	Passed	Yes	VR-SWL-B1-2	TM 15.04.2019	
Test	TV-52	JS 24.04.2019	R4	-27	The HWL should provide a list of available instructions to the SWL when prompted.	Functional test: Request a list of available instructions and check if what's sent reflects the expectations.		Hardware / JS	Passed	Yes	VR-HWL-B2	JS 06.05.2019	
Test	TV-53	TM 06.05.2019	R2	-35	Should be able to run acceptance test on applied test parameters and generate test report	Functional test: A complete test run should run based on configured instructions and give feedback based on configured test parameters		Software / TM	Passed	Yes	VR-SWL-B2	TM 12.05.2019	
Test	TV-54	TM 06.05.2019	R2	-35.01	Should be able to define acceptance test parameters	Functional Test: Parameters can be provided trough a UI and the parameters can be used to observe whether or not a value is inside their defined bounds		Software / TM	Passed	Yes	VR-SWL-B2	TM 12.05.2019	
Test	TV-55	TM 06.05.2019	R2	-35.02	Should provide functionality for saving acceptance test parameters to an acceptance test template	Functional test: The parameters can be saved to the database, and are loaded in the state that they was saved		Software / TM	Passed	No			
Test	TV-57	TM 06.05.2019	R2	-37	Should provide capabilities for communicate trough ethernet network			Software / TM	Passed	No			
Inspection	IV-58	EF 10.05.19	R2	-38	The sensor for angular position must fulfill : - Electrical angle of 360 degrees - Mechanical angle of <360 degrees - Suitable for 24VDC or 5VDC voltage level - Provide a output signal suitable for Arduino Mega	Inspection and fuctional test. Read output.		Software/Elec/JS+EF	Not tested	No			
Inspection	IV-59		R1	-02	the DS will be composed of the following components: - HMI Software for transmitting test instructions - Cabinet: DC power supply, motor and sensor controller (Microcontroller / FPGA) - Stepper motor drive - Angular position sensor - Fuses				Passed	No			

Bachelor of Engineering

Project appendix

Winter semester 2019

Risk Tables

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Håvar Østrem
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains a document descriptions and two risk tables. One for project risks and one for technical risks.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

Document Description

Both tables in this document have the same structure. The first three rows, explain the content of each column, with the exception of the two first cells in the second column. These explains the content in the colored rows (orange and red). The colored rows explains their respected columns. The uppermost red row explains the areas that may be affected of the risks. These are project Scope, project velocity, project cost, project quality and project credibility. The second orange row contains weights for the importance of each area of affection.

The first column contains the risk ID number. The second column contains the actual risks. The odd numbered columns from third to eleventh, under the area of affection contains the weights for how much each risk affect any of the five areas. The even numbered columns from the fourth to the twelfth contain the product of the weights. Namely the importance of each area and how much one risk affects that particular area. Each of these are scaled form one to ten, although the weights in the area of affection only go up to seven.

The thirteenth column (Total impact), are the sum of the products for each risk. The fourteenth column (normalized impact) scales down the total impact sum from one to six to correspond with the risk matrix. the fifteenth column (possible causes), explains what causes may result in the impact occurring. The sixteenth column contains the probability of one impact occurring, scaled from one to four to correspond with the risk matrix. The seventeenth column (total risk) contains the product of impact and probability. The eighteenth and last column contains suggestions of mitigation actions, to prevent the risk from occurring or minimize the impact for each risk.

	Area of affection	Project scope		Project velocity		Project cost		Product quality		Project credibility		Total Impact	Impact normalized	Possible causes	Probability (1-4)	Risk Product	Mitigation Action
Weight (1-10)		7		6		1		3		5							
Risk ID	RISKS																
RP-	Project Risks																
RP-01	One member quits	5	35	8	48	0	0	5	15	5	25	123	6	Illness, Lack of motivation, social conflicts	2	12	Encourage each other, show appreciation for the work that is being done
RP-02	Several members quits	10	70	10	60	0	0	10	30	8	40	200	6	Social conflicts	1	6	Encourage each other, show appreciation for the work that is being done
RP-03	One member does less work than a minimum of what is expected	3	21	5	30	0	0	3	9	4	20	80	3	Illness, Lack of motivation, social conflicts	3	9	Transparreny is high valued in the project, communicatin is key
RP-04	Several members does less work than a minimum of what is expected	5	35	8	48	0	0	6	18	6	30	131	6	Illness, Lack of motivation, social conflicts	2	12	Meetings and retrospective
RP-05	Requirements misinterpreted by the team	5	35	3	18	5	5	5	15	3	15	88	4	Miscommunication, lack of knowledge	2	8	Research, Communications and meetings within the team and with supervisors
RP-06	Requirements interpreted differently within the team	2	14	3	18	2	2	4	12	3	15	61	3	Miscommunication	2	6	Research, Communications and meetings within the team and with supervisors
RP-07	A few less important requirements are not met	2	14	2	12	0	0	3	9	2	10	45	2	Neglectance, too little time, bad assumptions	3	6	Less important requirements may have to be accepted. If they are not acceptable they are important
RP-08	Several requirements are not met	6	42	4	24	0	0	7	21	5	25	112	6	Neglectance, too little time, bad assumptions	2	12	Prioritize Must requirements
RP-09	Some very important requirements are not met	5	35	2	12	0	0	8	24	4	20	91	4	Neglectance, too little time, bad assumptions	3	12	If possible, use simplified solution
RP-10	Lack of documentation	3	21	2	12	0	0	5	15	10	50	98	5	Neglectance, too little time	4	20	Document as early as possilbe
RP-11	Loss of documents	3	21	4	24	0	0	4	12	9	45	102	5	Computer crashing, sabotage, neglectance	2	10	Documentation is archived i Google Drive, and in LaTeX documents in addition to each of the team members' computer
RP-12	A few errors in documents	1	7	2	12	0	0	3	9	5	25	53	2	Neglectance, too little time, miscalculations, language problems	3	6	The team has set aside a week for correction/proof reading of the documentation
RP-13	A lot of errors in documents	2	14	5	30	0	0	5	15	10	50	109	6	Neglectance, too little time, miscalculations, language problems	2	12	The team has set aside a week for correction/proof reading of the documentation
RP-14	Lack of references	0	0	0	0	0	0	2	6	10	50	56	2	Neglectance, too little time, lack of citation knowlegde	3	6	The team has set aside two days for correction of - and reference control
RP-15	Wrong choices in terms of making a satisfying product for the stakeholders	2	14	1	6	2	2	5	15	7	35	72	3	Misintrepretation of the project tasks, and wishes from the stakeholders	1	3	Short sprints, supervisor meetings for updates and feedback

	Area of affection	Project scope		Project velocity		Project cost		Product quality		Project credibility		Total Impact	Impact normalized	Possible causes	Probability (1-4)	Risk Product	Mitigation Action
Risk ID	RISKS																
RT-	Technical Risks																
RT-01	Flex radiates too much heat	0	0	0	0	0	0	8	24	5	25	49	2	High current, miscalculated derating	3	6	Hard to test and verify
RT-02	Wires radiates too much heat	0	0	0	0	0	0	8	24	5	25	49	2	High current, miscalculated derating	3	6	Hard to test and verify
RT-03	Electrical connections fail due to heat	0	0	0	0	0	0	8	24	6	30	54	2	High current, miscalculated derating, bad soldering	2	4	Assure good soldering
RT-04	Flex is too stiff	2	14	0	0	2	2	9	27	4	20	63	3	Too high percentage of copper, miscalculated TS geometry	3	9	Adjust TS geometry
RT-05	Flex does not fit Twist Capsule	0	0	4	24	2	2	9	27	8	40	93	4	Miscommunication between electro and mech	3	12	Tight collaboration between electro and mech
RT-06	Motor do not provide sufficient power	2	14	0	0	2	2	8	24	4	20	60	2	Too much resistance/torque, wrong ball bearings	2	4	Buy as powerful a motor as possilbe, use good bearings
RT-07	Motor is susceptible to emc noise	1	7	0	0	2	2	6	18	3	15	42	2	Bad shielding	3	6	Hard to test, use sufficient shilding
RT-08	Drivetrain does not move freely	0	0	1	6	1	1	8	24	6	30	61	3	Too much viscosity in the bearing lubricant	3	9	Research lubricant for bearings and choose the right bearing for the design
RT-09	Slack in drivetrain gears	0	0	1	6	1	1	8	24	6	30	61	3	Wrong measurements of the distance between the gears, or the measurements of the gears are wrong	3	9	Research gear technology, vendors and make sure that the holes in the motor holder are measured correctly
RT-10	Slack in ball bearings	0	0	1	6	2	2	8	24	2	10	42	2	Tolerances are wrong	1	2	Make sure tolerances are considered
RT-11	Flattened balls in ball bearings	0	0	1	6	2	2	7	21	1	5	34	1	The bearings are experiencing too much forces and gets deformed	1	1	Choose the right dimension in the bearings
RT-12	SADM housing trapps heat and pressure	2	14	2	12	0	0	8	24	2	10	60	2	No venting paths	3	6	Hard to test, drill holes, add heatsinks
RT-13	Bearings are too weak	3	21	2	12	0	0	7	21	2	10	64	3	Underdimensioned	2	6	Check acceleration of launch vehicle and design/choose bearings on this basis
RT-14	Flex does not arrive in time	5	35	0	0	0	0	8	24	6	30	89	4	Long ordering time	4	16	Research several possible vendors
RT-15	Mechanical parts does not arrive in time		0		0		0		0		0	0	1	Complex parts and late ordering time	2	2	Set a deadline for design. Early ordering of designed parts
RT-16	Late delivery of external components affects the testing	2	14	5	30	0	0	2	6	7	35	85	4	No parts, no testing	4	16	Make sure the HWL facilitates easy addition of additional external devices upon arrival.
RT-17	Software is too complicated to use for end-user	5	35	5	30	0	0	8	24	8	40	129	6	Bad UI design	2	12	Create a detailed user manual to help users use the software as it is intended to be used
RT-18	Software crashes during use in production	8	56	3	18	0	0	10	30	10	50	154	6	Null refence exceptions, Index out of bounds. thread safety	1	6	Do extensive testing and create unit tests to ensure code operability
RT-19	Hardware layer crashes during operation	5	35	3	18	0	0	10	30	10	50	133	6	Memory leaks. Access violation. Faulty error handling.	1	6	Ensure proper error handling is implemented.



Bachelor of Engineering

Project appendix

Winter semester 2019

SWOT Analysis

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Håvar Østrem
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains a SWOT analysis and a description of this.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

Description of Project SWOT Analysis

SWOT analysis is a project management tool for marketing, risk identification and general team self reflection. SWOT is an acronym of which the four letters stands for strengths, weaknesses, opportunities and threats. Strengths and weaknesses refers to the internal positive and negative qualities of the project group. Opportunities and threats refers to external positive and negative factors that is affecting the project or might affect the project in the future.

Our reason for using SWOT analysis is first and foremost for risk identification, as marketing isn't relevant for the project (at least not directly). Risk is generally negative, thus the most important categories of the SWOT analysis are weaknesses and threats. The idea is to accumulate information that we already know or realize early in the project and then rewrite the content as risks.

The positive categories, namely strengths and opportunities are mostly unrelated to risk. However, the content in these may help us to understand our advantages and create strategies that enhance the likelihood for success in the project.



Bachelor of Engineering

Project appendix

Winter semester 2019

Meeting requests

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	SatLight
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains email correspondence about meeting requests with internal and external supervisors.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 21, 2019	Signature
---------------------------------------	-----------

Contents

1	Jan 11	2
2	Jan 15	3
3	Jan 17	4
4	Jan 21	5
5	Jan 31 first presentation	6
6	Jan 31	8
7	Feb 05	10
8	Feb 07	12
9	Feb 13	14
10	Feb 14	15
11	Feb 21	17
12	Feb 28	19
13	Mar 11	20
14	Mar 18	21
15	Mar 26	22
16	Mar 21	24
17	Apr 01	25
18	Apr 08	26
19	Apr 04	27
20	Apr 11	29
21	Apr 17 and Apr 25	31
22	Apr 26	35
23	Apr 29	36

5/20/2019

Gmail - Veiledermøte ekstern veileder



Thomas Mundal <thmundal@gmail.com>

Veiledermøte ekstern veileder

1 message

Erica.Fegri@kongsberg.com <Erica.Fegri@kongsberg.com> Fri, Jan 11, 2019 at 11:27 AM
To: Eirik.Demmo.Normann@kongsberg.com, ritahogstad@gmail.com, mingkit.wong@gmail.com,
havar.ostrem@gmail.com, thmundal@gmail.com, skjelsbek@gmail.com

Ukentlig møte med ekstern veileder Eirik.

 **invite.ics**
3K



20.5.2019

E-post – Rita Hogstad – Outlook

Fwd: First presentation - Bachelor group 5 - 2019

Rita Hogstad <ritahogstad@gmail.com>

ti. 15.01.2019 11.45

Til: aage.soerensen@kongsberg.com <aage.soerensen@kongsberg.com>; eirik.demmo.normann@kongsberg.com <eirik.demmo.normann@kongsberg.com>

Hei dere!

Var ikke helt sikker på om dere fikk denne mailen i og med at dere ikke bruker Gmail, så jeg videresender igjen, slik at vi er sikker.

Med vennlig hilsen Rita Hogstad

----- Forwarded message -----

From: **Rita Hogstad** <ritahogstad@gmail.com>

Date: Tue, Jan 15, 2019 at 11:34 AM

Subject: First presentation - Bachelor group 5 - 2019

To: <havar.ostrem@gmail.com>, <skjelsbek@gmail.com>, <ritahogstad@gmail.com>, <thmundal@gmail.com>, <erica.fegri@kongsberg.com>, <josemmf@gmail.com>, <aage.soerensen@kongsberg.com>, <eirik.demmo.normann@kongsberg.com>, <mingkit.wong@gmail.com>, <moholth@gmail.com>

Hello everyone,

We wish to welcome you to our first presentation in our bachelor project.

We have set the date to the 4th of February at 09.00 to 11.00, in class room 5119 and meeting room "Åkern" at Krona.

For those of you who don't know where the rooms are, we will meet you by the main entrance at the first floor at Krona 09.00.

It is very important that external censor (Aage) and internal censor (Jose) is present.

Documentation will be sent to you two days in advance of the presentation.

Please let us know if you have any questions.

Best regards,

Rita Hogstad, on behalf of bachelor group 5

First presentation - Bachelor 2019

Når man. 4. feb 2019 9am – 11am Sentraleuropeisk tid - Oslo

Hvem • Rita Hogstad– oppretter

20.5.2019


E-post – Rita Hogstad – Outlook

Meeting 18.01-2019 - Bachelor group 5

Rita Hogstad

to. 17.01.2019 16.05

Til: josemmf.usn@gmail.com <josemmf.usn@gmail.com>

 1 vedlegg (167 kB)

agenda-documents-18.01.19.pdf;

Hi Jose!

Attached in this mail lies some information about our process model, our roles in the project and a temporary Gantt chart.

So feel free to look at that if you want.

The agenda for the meeting tomorrow:

- Inform Jose about the model
- Feedback on our project model
- Feedback on our project plan, and Gantt chart
- Feedback on our agenda for the first presentation

The meeting will be held at our group room (2263) at Krona, 10.00 to 11.00.

Best regards,

Rita Hogstad, on behalf of our group.

5/20/2019

Gmail - Agenda for møte 23.01.2019



Thomas Mundal <thmundal@gmail.com>

Agenda for møte 23.01.2019

1 message

Thomas Mundal <thmundal@gmail.com>

Mon, Jan 21, 2019 at 3:48 PM

To: eirik.demmo.normann@kongsberg.com, Erica Fegri <ericafegri@gmail.com>, Håvar Østrem <havar.ostrem@gmail.com>, Rita Hogstad <ita95@hotmail.no>, Ming Kit Wong <mingkit.wong@gmail.com>, Jon Skjelsbek <skjelsbek@gmail.com>

Vi kaller inn til møte igjen, på Kongsberg Innovation Center, samme rom som sist. Kalenderinvitasjon kommer i egen epost. Under finner du punker vi ønsker å legge på agendaen, og vedlagt ligger det noen dokumenter fra oss som inneholder informasjon vi ønsker å legge frem.

Et tilleggsønske er om du har en flex som vi kan benytte til prototypen?

1. Avklare om KSS har noen spesifikke tekniske eller funksjonelle krav til prototypen. Hvilket miljø er rimelig at prototypen utvikles/testes for? (Ground/Launch/Orbit)
2. VCD
3. Avklare status på kontrakt og budsjett.
4. Eksempler på riskanalyse
5. Jon: Ønsker å vite hvor Eirik stiller seg til utvikling av FPGA/SADE. Det er ønskelig for oss å utvikle en slik enhet selv, siden vi ønsker å tilrettelegge for testing. Hvis det skal utvikles en slik enhet er det interessant å vite noe om hva som kommer fra "Main Computer", eller da evt. om dette ikke er relevant.
6. Fremlegge revidert utkast av oppgavetekst
7. Legge frem plan for håndtering av krav (Erica) - Hvordan ønsker Eirik å ha innsikt i dette?
8. Fremlegge prosjektplan

--

Med Vennlig Hilsen

Thomas Mundal
thmundal@gmail.com
TLF: 92808309

 **meeting-documents.pdf**
534K

5/20/2019

Gmail - Filer og informasjon om første presentasjon - Bachelorgruppe 5 / SatLight



Thomas Mundal <thmundal@gmail.com>

Filer og informasjon om første presentasjon - Bachelorgruppe 5 / SatLight

1 message

Rita Hogstad <ita95@hotmail.no>

Thu, Jan 31, 2019 at 10:24 AM

To: "aage.soerensen@kongsberg.com" <aage.soerensen@kongsberg.com>, "skjelsbek@gmail.com" <skjelsbek@gmail.com>, "havar.ostrem@gmail.com" <havar.ostrem@gmail.com>, Erica Fegri <ericafegri@gmail.com>, Thomas Mundal <thmundal@gmail.com>, "mingkit.wong@gmail.com" <mingkit.wong@gmail.com>

Hei Aage! 😊

Velkommen til første presentasjon på mandag 4. Februar.

Vedlagt ligger alle filene vi føler er relevant og interessant for deg.

Ta gjerne en kikk i filen som heter "[Description] Folder content" for å se hva mappen inneholder.

Vi møter deg gjerne ved hovedinngangen (første etasje på Krona) 08.55.

Formøtet til presentasjonen begynner klokken 09.00 i femte etasje på Krona.

Vedrørende parkering: Det er 3 timers parkering på kirketorget, eventuelt langs veien ved Krona. I tillegg er det mulig å parkere på Skauløkka P-hus.

For ytterligere informasjon ta gjerne kontakt med meg på denne mail-adressen eller på tlf: 95 47 05 40

Vi ses!

Med vennlig hilsen

Rita Hogstad, på vegne av Bachelorgruppe 5 - SatLight

5/20/2019

Gmail - Filer og informasjon om første presentasjon - Bachelorgruppe 5 / SatLight

 **Documents - First Presentation.zip**
3231K

5/20/2019

Gmail - 1st Presentation files, and agenda - Bachelor



Thomas Mundal <thmundal@gmail.com>

1st Presentation files, and agenda - Bachelor

1 message

Rita Hogstad <ita95@hotmail.no>

Thu, Jan 31, 2019 at 11:43 AM

To: "josemmf.usn@gmail.com" <josemmf.usn@gmail.com>

Cc: "mingkit.wong@gmail.com" <mingkit.wong@gmail.com>, Erica Fegri <ericafegri@gmail.com>, Thomas Mundal <thmundal@gmail.com>, "skjelsbek@gmail.com" <skjelsbek@gmail.com>, "havar.ostrem@gmail.com" <havar.ostrem@gmail.com>

Hi Jose! 😊

Here are two links that we think you might be interested in:

Link to relevant files before the presentation:<https://goo.gl/Cr6Edr>

(All the files are attached in a zip folder in the mail)

Link to our presentation slides:<https://docs.google.com/presentation/d/1tO7GZrpVx8hQ3jwlyyYWCo4I1z1xpDcq1pI0mgYjsDA/edit?usp=sharing>**Agenda for the meeting tomorrow:**

- Presentation slides
- Verification Management Document (VMD)
- Files in docs

5/20/2019

Gmail - 1st Presentation files, and agenda - Bachelor

Best regards,
Rita Hogstad, on behalf of team SatLight (Bachelorgroup 5)

 **Files - First Presentation - Jose.zip**
14109K



5/20/2019

Gmail - Fwd: Veiledermøte onsdag 06.02.19



Thomas Mundal <thmundal@gmail.com>

Fwd: Veiledermøte onsdag 06.02.19

1 message

Erica Fegri <ericafegri@gmail.com>

Tue, Feb 5, 2019 at 3:53 PM

To: mingkit.wong@gmail.com, thmundal@gmail.com, ritahogstad@gmail.com, skjelsbek@gmail.com, havar.ostrem@gmail.com

Med vennlig hilsen

Erica Fegri
Tlf: 95768624

Videresendt melding:

Fra: <Eirik.Demmo.Normann@kongsberg.com>**Dato:** 5. februar 2019 kl. 15:36:04 CET**Til:** <ericafegri@gmail.com>**Emne:** RE: Veiledermøte onsdag 06.02.19

Heisann!

Den er god! Hørte Aage var fornøyd med presentasjonen. For øvrig han dere kan skylde å for mitt fravær og. Jeg ble sendt til Oslo for å skryte og menge meg med romfartsiffen..

Lurer på om det er denne vi er ute etter:

ECSS-E-ST-50-13C Interface and communication protocol for MIL-STD-1553B data bus onboard spacecraft

Jeg har litt PC-trøbbel om dagen og får ikke sendt vedlegg fra desktoopen min og er for lat til å ta frem den laptopen nå, så dere for gi en lyd om dere ikke finner den på nett så skal jeg få sendt PDF fra serveren vår.

Suveniren er mottatt og settes veldig pris på ☺

--Eirik

From: Erica Fegri <ericafegri@gmail.com>**Sent:** 4. februar 2019 11:26**To:** Normann, Eirik Demmo <Eirik.Demmo.Normann@kongsberg.com>**Subject:** Veiledermøte onsdag 06.02.19

Hei Eirik,



5/20/2019

Gmail - Fwd: Veiledermøte onsdag 06.02.19

Vedrørende møtet på onsdag tenker vi å fristille deg fra oppmøte. Vi har jobbet med presentasjonen de siste dagene og har ikke noe nytt å legge på bordet til førstkomende onsdag. Dersom du ikke har noe du ønsker å ta opp med oss denne uken, tenker vi at vi kan benytte onsdag neste uke til å vise frem prototypen vår.

Om du har funnet ut hvilken MIL-standard vi snakket om forrige uke, er det flott om du kan sende oss navnet på denne.

Selv om vi gjerne skulle sett ditt blide åsyn i dag, har vi laget en liten suvenir til deg som er sendt med Aage. 😊

Vedlagt link til presentasjonen i dag;

<https://docs.google.com/presentation/d/1tO7GZrpVx8hQ3jwlyyYwCo4l1z1xpDcq1pl0mgYjsDA/edit?usp=sharing>

Med vennlig hilsen

Erica Fegri

På vegne av bachelorgruppa

CONFIDENTIALITY

This e-mail and any attachment contain KONGSBERG information which may be proprietary, confidential or subject to export regulations, and is only meant for the intended recipient(s). Any disclosure, copying, distribution or use is prohibited, if not otherwise explicitly agreed with KONGSBERG. If received in error, please delete it immediately from your system and notify the sender properly.

5/20/2019

Gmail - Meeting friday 08.02-2019



Thomas Mundal <thmundal@gmail.com>

Meeting friday 08.02-2019

3 messages

Håvar Østrem <havar.ostrem@gmail.com> Thu, Feb 7, 2019 at 1:27 PM
To: josemmf.usn@gmail.com, Erica Fegri <ericafegri@gmail.com>, Rita Hogstad <ita95@hotmail.no>, Thomas Mundal <thmundal@gmail.com>, Jon Skjelsbek <skjelsbek@gmail.com>, Ming Kit Wong <mingkit.wong@gmail.com>

Hi José

Here is the agenda for the meeting tomorrow:

- Requirements definition for Diagnostics System software layer
- Requirements definition for Diagnostics System hardware layer
- Availability for 2nd and 3rd presentation

Attached is the Diagnostics System requirements.

Best regards,
Håvar Østrem, on behalf of Satlight

2 attachments **Diagnostics_System-Hardware_Layer-Requirements.pdf**
17K **DiagnosticsSystem-SoftwareLayer-070219-1108.pdf**
24K

Ming Kit Wong <mingkit.wong@gmail.com> Fri, Feb 8, 2019 at 10:37 AM
To: Håvar Østrem <havar.ostrem@gmail.com>
Cc: josemmf.usn@gmail.com, Erica Fegri <ericafegri@gmail.com>, Rita Hogstad <ita95@hotmail.no>, Thomas Mundal <thmundal@gmail.com>, Jon Skjelsbek <skjelsbek@gmail.com>


Hello,

5/20/2019

Gmail - Meeting friday 08.02-2019

Here is the meeting notes from today's meeting. No major decisions taken.

Best regards,
Ming Kit
[Quoted text hidden]

 **2019-02-08 Meeting notes.pdf**
8K

Jose Ferreira <Jose.Ferreira@usn.no> Fri, Feb 8, 2019 at 11:13 AM
To: Ming Kit Wong <mingkit.wong@gmail.com>
Cc: Håvar Østrem <havar.ostrem@gmail.com>, "josemmf.usn@gmail.com" <josemmf.usn@gmail.com>, Erica Fegri <ericafegri@gmail.com>, Rita Hogstad <ita95@hotmail.no>, Thomas Mundal <thmundal@gmail.com>, Jon Skjelsbek <skjelsbek@gmail.com>

Hil :)
Just a comment concerning my availability on March 15th – in the morning I am available without restrictions, in the afternoon it will depend on the time. Although my flight leaves Gardermoen at 6:55 pm, I sill have to go from Kongsberg to Gardermoen, and be there one hour before the flight...

Kind regards, Jose.

José Manuel Martins Ferreira
Professor
Universitetet i Sørøst-Norge
Fakultet for teknologi, naturvitenskap og maritime fag
Campus Kongsberg

[Quoted text hidden]
[Quoted text hidden]
<2019-02-08 Meeting notes.pdf>

5/20/2019

Gmail - Omvisning Mek. Lab.



Thomas Mundal <thmundal@gmail.com>

Omvisning Mek. Lab.

1 message

Eirik.Demmo.Normann@kongsberg.com
<Eirik.Demmo.Normann@kongsberg.com>Wed, Feb 13, 2019 at 2:55
PM

To: Runar.Lund@kongsberg.com, thmundal@gmail.com, ericafegri@gmail.com, havar.ostrem@gmail.com, ita95@hotmail.no, skjelsbek@gmail.com, mingkit.wong@gmail.com, Geir.Andresen@kongsberg.com, Daniel.Engebretsen@kongsberg.com

Hei!

Da får vi lov til å ta en tur innom Mek. Lab. og kikke på fasilitetene deres onsdag klokken 10 ☺

Prøver å ordne så vi kan ta en kikk på renrom og testfasiliteter i etterkant.

Mvh

Eirik

 **invite.ics**
3K

5/20/2019

Gmail - Agenda for weekly meeting



Thomas Mundal <thmundal@gmail.com>

Agenda for weekly meeting

4 messages

Ming Kit Wong <mingkit.wong@gmail.com>

Thu, Feb 14, 2019 at 8:20 PM

To: josemmf.usn@gmail.com

Cc: Jon Skjelsbek <skjelsbek@gmail.com>, Thomas Mundal <thmundal@gmail.com>, Håvar Østrem <havar.ostrem@gmail.com>, Erica Fegri <ericafegri@gmail.com>, Rita Hogstad <ita95@hotmail.no>

Hello, Jose

This week, we want to show you our timeline and talk about the updates done in the requirements document for SW layer.

 [USA-DiagnosticsSystem-SoftwareLayer-140219-0847...](#)Best regards,
Ming Kit

Jose Ferreira <Jose.Ferreira@usn.no>

Thu, Feb 14, 2019 at 8:32 PM

To: Ming Kit Wong <mingkit.wong@gmail.com>

Cc: "josemmf.usn@gmail.com" <josemmf.usn@gmail.com>, Jon Skjelsbek <skjelsbek@gmail.com>, Thomas Mundal <thmundal@gmail.com>, Håvar Østrem <havar.ostrem@gmail.com>, Erica Fegri <ericafegri@gmail.com>, Rita Hogstad <ita95@hotmail.no>

Hi Ming!

Thanks! Would it be OK to start our meeting at 09:30?

(I would like to leave at 10:55, and that would give us some more time...)

Kind regards, Jose.

José Manuel Martins Ferreira
Professor

5/20/2019

Gmail - Agenda for weekly meeting

Universitetet i Sørøst-Norge
Fakultet for teknologi, naturvitenskap og maritime fag
Campus Kongsberg

[Quoted text hidden]

Ming Kit Wong <mingkit.wong@gmail.com> Thu, Feb 14, 2019 at 8:48 PM
To: Jose Ferreira <Jose.Ferreira@usn.no>
Cc: josemmf.usn@gmail.com, Jon Skjelsbek <skjelsbek@gmail.com>, Thomas Mundal <thmundal@gmail.com>, Håvar Østrem <havar.ostrem@gmail.com>, Erica Fegri <ericafegri@gmail.com>, Rita Hogstad <ita95@hotmail.no>

09:30 works for the team. See you tomorrow :)

[Quoted text hidden]

Jose Ferreira <Jose.Ferreira@usn.no> Thu, Feb 14, 2019 at 8:49 PM
To: Ming Kit Wong <mingkit.wong@gmail.com>
Cc: Jon Skjelsbek <skjelsbek@gmail.com>, Thomas Mundal <thmundal@gmail.com>, Håvar Østrem <havar.ostrem@gmail.com>, Erica Fegri <ericafegri@gmail.com>, Rita Hogstad <ita95@hotmail.no>

Great! :)

José Manuel Martins Ferreira
Professor
Universitetet i Sørøst-Norge
Fakultet for teknologi, naturvitenskap og maritime fag
Campus Kongsberg

[Quoted text hidden]

5/20/2019

Gmail - Fwd: Meeting canceled 22.02.19



Thomas Mundal <thmundal@gmail.com>

Fwd: Meeting canceled 22.02.19

1 message

Jon Skjelsbek <skjelsbek@gmail.com>

Thu, Feb 21, 2019 at 7:12 PM

To: Thomas Mundal <thmundal@gmail.com>, Håvar Østrem <havar.ostrem@gmail.com>, Ming Kit Wong <mingkit.wong@gmail.com>, Rita Hogstad <ita95@hotmail.no>, Erica Fegri <ericafegri@gmail.com>

----- Forwarded message -----

From: **Jose Ferreira** <Jose.Ferreira@usn.no>

Date: Thu, 21 Feb 2019, 18:59

Subject: Re: Meeting canceled 22.02.19

To: Jon Skjelsbek <skjelsbek@gmail.com>

Hi Jon!

No problem at all!

Would it be OK that we move our meeting next week from 10am to 9am?

Kind regards, Jose.

José Manuel Martins Ferreira
Professor
Universitetet i Sørøst-Norge
Fakultet for teknologi, naturvitenskap og maritime fag
Campus Kongsberg

On 21 Feb 2019, at 14:16, Jon Skjelsbek <skjelsbek@gmail.com> wrote:

5/20/2019

Gmail - Fwd: Meeting canceled 22.02.19

Hello José!

After some consideration, we've concluded that as of now we have almost nothing to disclose at the meeting scheduled tomorrow (22.02.19). This in combination with illness within the group is the main reasons why we decided to cancel the meeting, unless you have objections.

Kind regards,
Jon Skjelsbæk, on behalf of SatLight.



5/20/2019

Gmail - Agenda meeting 29.02.19



Thomas Mundal <thmundal@gmail.com>

Agenda meeting 29.02.19

1 message

Jon Skjelsbek <skjelsbek@gmail.com>

Thu, Feb 28, 2019 at 3:41 PM

To: Jose Ferreira <Jose.Ferreira@usn.no>

Cc: Thomas Mundal <thmundal@gmail.com>, Håvar Østrem <havar.ostrem@gmail.com>, Ming Kit Wong <mingkit.wong@gmail.com>, Erica Fegri <ericafegri@gmail.com>, Rita Hogstad <ita95@hotmail.no>

Hi José,

We would like to dedicate most of the upcoming meeting to inform you of our progress and future plan in general, but there are a few things we would like discuss apart from that.

In the attached document you'll find a description of our communication protocol, as well as one for the Software Layer Alpha2 version. We would like to discuss the designed of the communication protocol, as well as the class- and sequence diagrams.

Kind regards,
Jon, on behalf of the group.

 **Attachments.pdf**
289K

20.5.2019

E-post – Rita Hogstad – Outlook

Sv: Your presentation today

Rita Hogstad

ma. 11.03.2019 10.25

Til: Jose Ferreira <Jose.Ferreira@usn.no>

Helga is the smallest of the cinemas 😊 The one to the right

Fra: Jose Ferreira <Jose.Ferreira@usn.no>**Sendt:** mandag 11. mars 2019 10.20**Til:** Rita Hogstad**Emne:** Re: Your presentation today

Yes, of course! Is Helga one of the meeting rooms in the 5th floor?

José Manuel Martins Ferreira

Professor

Universitetet i Sørøst-Norge

Fakultet for teknologi, naturvitenskap og maritime fag

Campus Kongsberg

On 11 Mar 2019, at 10:19, Rita Hogstad <ita95@hotmail.no> wrote:

Hi Jose 😊

I called Tina in the reseption, and we figured that we should try to present at Helga.
Meet you there 11.00 or 11.10?

- Rita

Fra: Jose Ferreira <Jose.Ferreira@usn.no>**Sendt:** mandag 11. mars 2019 08.52**Til:** thmundal@gmail.com; ita95@hotmail.no; mingkit.wong@gmail.com; havar.ostrem@gmail.com; ericafegri@gmail.com; skjelsbek@gmail.com**Emne:** Your presentation today

Hi Guys!

I will be in a meeting that starts at 9 am, but will be free by 11 am.
I will check my email to know what room I shall head to.

Regards, Jose.

José Manuel Martins Ferreira

Professor

Universitetet i Sørøst-Norge

Fakultet for teknologi, naturvitenskap og maritime fag

Campus Kongsberg



5/20/2019

Gmail - Agenda onsdag 20.03.2019



Thomas Mundal <thmundal@gmail.com>

Agenda onsdag 20.03.2019

1 message

Thomas Mundal <thmundal@gmail.com>

Mon, Mar 18, 2019 at 1:04 PM

To: eirik.demmo.normann@kongsberg.com, Ming Kit Wong <mingkit.wong@gmail.com>, Erica Fegri <ericafegri@gmail.com>, Håvar Østrem <havar.ostrem@gmail.com>, Rita Hogstad <ita95@hotmail.no>, Jon Skjelsbek <skjelsbek@gmail.com>

Hei alle sammen

Vi møtes igjen på KIC til onsdag. På agendaen står det å snakke om referanser (ref tilbakemelding på presentasjon).

Vi sees!

--

Med Vennlig Hilsen

Thomas Mundal
thmundal@gmail.com
TLF: 92808309



20.5.2019

E-post – Rita Hogstad – Outlook

Re: Meeting on Friday this week

Rita Hogstad

ti. 26.03.2019 13.11

Til: Jose Ferreira <jose.ferreira@usn.no>**Kopi:** Erica Fegri <ericafegri@gmail.com>; Ming Kit Wong <mingkit.wong@gmail.com>; Thomas Mundal <thmundal@gmail.com>; havar.ostrem@gmail.com <havar.ostrem@gmail.com>; skjelsbek@gmail.com <skjelsbek@gmail.com>

Thank you 🙏🌸

From: Jose Ferreira <Jose.Ferreira@usn.no>**Sent:** Tuesday, March 26, 2019 12:41:45 PM**To:** Rita Hogstad**Cc:** Erica Fegri; Ming Kit Wong; Thomas Mundal; havar.ostrem@gmail.com; skjelsbek@gmail.com**Subject:** Re: Meeting on Friday this week

Hi Rita!

That's fine with me. I wish you good luck for your exams!

Kind regards, Jose.

José Manuel Martins Ferreira
Professor
Universitetet i Sørøst-Norge
Fakultet for teknologi, naturvitenskap og maritime fag
Campus Kongsberg

On 26 Mar 2019, at 11:34, Rita Hogstad <ita95@hotmail.no> wrote:

Hello Jose 😊

Due to our exams coming up next week, we have decided not to arrange supervisor meetings.
Therefore we suggest that the next meeting will be 12th of april. Hope that this is okay for you!

Best regards,
Rita, and SatLight

Fra: Jose Ferreira <Jose.Ferreira@usn.no>**Sendt:** tirsdag 26. mars 2019 11.10**Til:** Thomas Mundal; Rita Hogstad; Ming Kit Wong; Håvar Østrem; Erica Fegri; Jon Skjelsbek**Emne:** Meeting on Friday this week

Dear All,

Because I need to participate in a meeting that will start at 9 am, it is possible that I am not ready at 10 am for our usual meeting. Is it OK with you that we start at 10:30 instead of 10 am?



20.5.2019

E-post – Rita Hogstad – Outlook

Thanks and kind regards, Jose.

José Manuel Martins Ferreira
Professor
Universitetet i Sørøst-Norge
Fakultet for teknologi, naturvitenskap og maritime fag
Campus Kongsberg



5/20/2019

Gmail - Meeting

Cc: Thomas Mundal <thmundal@gmail.com>, Rita Hogstad <ita95@hotmail.no>, Ming Kit Wong <mingkit.wong@gmail.com>, Håvar Østrem <havar.ostrem@gmail.com>, Erica Fegri <ericafegri@gmail.com>

Hi José,

We're really sorry we didn't let you know earlier, but we've decided that as of today we're going spend most of our time individually preparing for the upcoming exam. This implies that we won't necessarily be working together with the bachelor project from 9am to 4pm as usual, and thus we would like to cancel tomorrow's meeting.

Best regards,
Jon, on behalf of the group

[Quoted text hidden]

Jose Ferreira <Jose.Ferreira@usn.no>

Thu, Mar 21, 2019 at 7:02 PM

To: Jon Skjelsbek <skjelsbek@gmail.com>

Cc: Thomas Mundal <thmundal@gmail.com>, Rita Hogstad <ita95@hotmail.no>, Ming Kit Wong <mingkit.wong@gmail.com>, Håvar Østrem <havar.ostrem@gmail.com>, Erica Fegri <ericafegri@gmail.com>

That's OK, I actually wondered that might be the case, since you usually send an agenda beforehand. But there's no problem at all, it doesn't cause me any problem.

Kind regards, Jose.

José Manuel Martins Ferreira
Professor
Universitetet i Sørøst-Norge
Fakultet for teknologi, naturvitenskap og maritime fag
Campus Kongsberg

[Quoted text hidden]

Jon Skjelsbek <skjelsbek@gmail.com>

Thu, Mar 21, 2019 at 7:55 PM

To: Jose Ferreira <Jose.Ferreira@usn.no>

Cc: Thomas Mundal <thmundal@gmail.com>, Rita Hogstad <ita95@hotmail.no>, Ming Kit Wong <mingkit.wong@gmail.com>, Håvar Østrem <havar.ostrem@gmail.com>, Erica Fegri <ericafegri@gmail.com>



5/20/2019

Gmail - Avlyst: Veiledermøte ekstern veileder



Thomas Mundal <thmundal@gmail.com>

Avlyst: Veiledermøte ekstern veileder

1 message

Erica.Fegri@kongsberg.com <Erica.Fegri@kongsberg.com> Mon, Apr 1, 2019 at 11:15 AM
To: Eirik.Demmo.Normann@kongsberg.com, ritahogstad@gmail.com, mingkit.wong@gmail.com,
havar.ostrem@gmail.com, thmundal@gmail.com, skjelsbek@gmail.com
Cc: ita95@hotmail.no, Carsten.Haavardtun@kongsberg.com, Lars.Helge.Surdal@kongsberg.com

Ukentlig møte med ekstern veileder Eirik.

 **invite.ics**
3K



5/20/2019

Gmail - Veiledermøte 10. April



Thomas Mundal <thmundal@gmail.com>

Veiledermøte 10. April

1 message

Eirik.Demmo.Normann@kongsberg.com

Mon, Apr 8, 2019 at 4:58

<Eirik.Demmo.Normann@kongsberg.com>

PM

To: ita95@hotmail.no, mingkit.wong@gmail.com, ericafegri@gmail.com, skjelsbek@gmail.com, havar.ostrem@gmail.com, thmundal@gmail.com

Hei!

Jeg har et møte i Oslo på onsdagen så jeg rekker dessverre ikke denne, Aage er også bortreist. Jeg kan høre med Lars Helge og Carsten (som var med en gang her tidligere) om en eller begge kan ta seg turen?

Mvh

Eirik

CONFIDENTIALITY

This e-mail and any attachment contain KONGSBERG information which may be proprietary, confidential or subject to export regulations, and is only meant for the intended recipient(s). Any disclosure, copying, distribution or use is prohibited, if not otherwise explicitly agreed with KONGSBERG. If received in error, please delete it immediately from your system and notify the sender properly.

5/20/2019

Gmail - Meeting



Thomas Mundal <thmundal@gmail.com>

Meeting

3 messages

Jose Ferreira <Jose.Ferreira@usn.no>

Thu, Apr 4, 2019 at 8:32 PM

To: Thomas Mundal <thmundal@gmail.com>, Rita Hogstad <ita95@hotmail.no>, Ming Kit Wong <mingkit.wong@gmail.com>, Håvar Østrem <havar.ostrem@gmail.com>, Erica Fegri <ericafegri@gmail.com>, Jon Skjelsbek <skjelsbek@gmail.com>

Ladies and Gentlemen! :)
How are you doing? Are we going to talk tomorrow? How's the work going?

Kind regards, Jose.

José Manuel Martins Ferreira
Professor
Universitetet i Sørøst-Norge
Fakultet for teknologi, naturvitenskap og maritime fag
Campus Kongsberg

Rita Hogstad <ita95@hotmail.no>

Thu, Apr 4, 2019 at 9:03 PM

To: Jose Ferreira <Jose.Ferreira@usn.no>, Thomas Mundal <thmundal@gmail.com>, Ming Kit Wong <mingkit.wong@gmail.com>, Håvar Østrem <havar.ostrem@gmail.com>, Erica Fegri <ericafegri@gmail.com>, Jon Skjelsbek <skjelsbek@gmail.com>

Hello Jose 😊

Electro and mechanical are having their exams tomorrow, so there will be no meeting.

Next meeting will be next friday!

Best regards,
Rita and SatLight

Skaff deg [Outlook for Android](#)

From: Jose Ferreira <Jose.Ferreira@usn.no>**Sent:** Thursday, April 4, 2019 8:32:04 PM**To:** Thomas Mundal; Rita Hogstad; Ming Kit Wong; Håvar Østrem; Erica Fegri; Jon Skjelsbek**Subject:** Meeting

[Quoted text hidden]

Jose Ferreira <Jose.Ferreira@usn.no>

Thu, Apr 4, 2019 at 9:07 PM

To: Rita Hogstad <ita95@hotmail.no>

Cc: Thomas Mundal <thmundal@gmail.com>, Ming Kit Wong <mingkit.wong@gmail.com>, Håvar Østrem <havar.ostrem@gmail.com>, Erica Fegri <ericafegri@gmail.com>, Jon Skjelsbek <skjelsbek@gmail.com>

OK (Hangout it will be)
I hope the work is going well?

José Manuel Martins Ferreira
Professor

5/20/2019

Gmail - Meeting

Universitetet i Sørøst-Norge
Fakultet for teknologi, naturvitenskap og maritime fag
Campus Kongsberg

[Quoted text hidden]



20.5.2019

E-post – Rita Hogstad – Outlook

Sv: Meeting

Rita Hogstad

to. 11.04.2019 12.52

Til: Jose Ferreira <Jose.Ferreira@usn.no>

 3 vedlegg (408 kB)

April.PNG; June.PNG; May.PNG;

Hello Jose 😊

Sorry for the late answer!

Our work is going well, although we have experienced some delays in the design process. We are working hard to get the design finished so that we can order the remaining components that we need. I made a calendar with some milestones if you want to look at that, the pictures are attached in this mail. 😊

Are you in Portugal now? And do you have the opportunity to have a meeting with us at 10 am tomorrow?

Best regards,
Rita Hogstad and SatLight

Fra: Jose Ferreira <Jose.Ferreira@usn.no>**Sendt:** torsdag 4. april 2019 21.07**Til:** Rita Hogstad**Kopi:** Thomas Mundal; Ming Kit Wong; Håvar Østrem; Erica Fegri; Jon Skjelsbek**Emne:** Re: Meeting

OK (Hangout it will be)
I hope the work is going well?

José Manuel Martins Ferreira
Professor
Universitetet i Sørøst-Norge
Fakultet for teknologi, naturvitenskap og maritime fag
Campus Kongsberg

On 4 Apr 2019, at 21:03, Rita Hogstad <ita95@hotmail.no> wrote:

Hello Jose 😊

Electro and mechanical are having their exams tomorrow, so there will be no meeting.



20.5.2019

E-post – Rita Hogstad – Outlook

Next meeting will be next friday!

Best regards,
Rita and SatLight

Skaff deg [Outlook for Android](#)

From: Jose Ferreira <Jose.Ferreira@usn.no>

Sent: Thursday, April 4, 2019 8:32:04 PM

To: Thomas Mundal; Rita Hogstad; Ming Kit Wong; Håvar Østrem; Erica Fegri; Jon Skjelsbek

Subject: Meeting

Ladies and Gentlemen! :)

How are you doing? Are we going to talk tomorrow? How's the work going?

Kind regards, Jose.

José Manuel Martins Ferreira
Professor
Universitetet i Sørøst-Norge
Fakultet for teknologi, naturvitenskap og maritime fag
Campus Kongsberg



5/20/2019

Gmail - Skype meeting 19. April 2019



Thomas Mundal <thmundal@gmail.com>

Skype meeting 19. April 2019

7 messages

Håvar Østrem <havar.ostrem@gmail.com>

Wed, Apr 17, 2019 at 6:29 PM

To: Jose.Ferreira@usn.no

Cc: Erica Fegri <ericafegri@gmail.com>, Rita Hogstad <ita95@hotmail.no>, Thomas Mundal <thmundal@gmail.com>, Jon Skjelsbek <skjelsbek@gmail.com>, Ming Kit Wong <mingkit.wong@gmail.com>

Hello Jose.

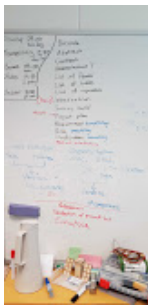
We would like to discuss our Report outline with you tomorrow at 10:00.

Please see the attachments for more content. This is what we have discussed so far.

Best regards

Håvar Østrem and SatLight

Virusfri. www.avg.com

2 attachments**report_outline_whiteboard_draft.jpg**
302K**Report Outline.pdf**
25K

5/20/2019

Gmail - Skype meeting 19. April 2019

Jose Ferreira <Jose.Ferreira@usn.no> Wed, Apr 17, 2019 at 6:32 PM
To: Håvar Østrem <havar.ostrem@gmail.com>
Cc: Erica Fegri <ericafegri@gmail.com>, Rita Hogstad <ita95@hotmail.no>, Thomas Mundal <thmundal@gmail.com>, Jon Skjelsbek <skjelsbek@gmail.com>, Ming Kit Wong <mingkit.wong@gmail.com>

OK! :)
Will you please create the Hangout event and send me the link?

Kind regards, Jose.

José Manuel Martins Ferreira
Professor
Universitetet i Sørøst-Norge
Fakultet for teknologi, naturvitenskap og maritime fag
Campus Kongsberg
[Quoted text hidden]

<report_outline_whiteboard_draft.jpg>


<Report Outline.pdf>

Ming Kit Wong <mingkit.wong@gmail.com> Thu, Apr 18, 2019 at 1:30 PM
To: Jose Ferreira <Jose.Ferreira@usn.no>
Cc: Håvar Østrem <havar.ostrem@gmail.com>, Erica Fegri <ericafegri@gmail.com>, Rita Hogstad <ita95@hotmail.no>, Thomas Mundal <thmundal@gmail.com>, Jon Skjelsbek <skjelsbek@gmail.com>

Here is the meeting note from todays meeting. You can find a suggestion of the report outline that we agreed on and a few other guidelines.

I see that "challenges" is missing.

Best regards,
Ming Kit
[Quoted text hidden]

 **USA-2019-04-18Meetingnotes-180419-1127.pdf**
6K

5/20/2019

Gmail - Skype meeting 19. April 2019

Jose Ferreira <Jose.Ferreira@usn.no> Thu, Apr 18, 2019 at 1:34 PM
To: Ming Kit Wong <mingkit.wong@gmail.com>
Cc: Håvar Østrem <havar.ostrem@gmail.com>, Erica Fegri <ericafegri@gmail.com>, Rita Hogstad <ita95@hotmail.no>, Thomas Mundal <thmundal@gmail.com>, Jon Skjelsbek <skjelsbek@gmail.com>

Thanks, Ming! :)

José Manuel Martins Ferreira
Professor
Universitetet i Sørøst-Norge
Fakultet for teknologi, naturvitenskap og maritime fag
Campus Kongsberg

[Quoted text hidden]
[Quoted text hidden]
<USA-2019-04-18Meetingnotes-180419-1127.pdf>

Jose Ferreira <Jose.Ferreira@usn.no> Thu, Apr 25, 2019 at 10:05 PM
To: Ming Kit Wong <mingkit.wong@gmail.com>
Cc: Håvar Østrem <havar.ostrem@gmail.com>, Erica Fegri <ericafegri@gmail.com>, Rita Hogstad <ita95@hotmail.no>, Thomas Mundal <thmundal@gmail.com>, Jon Skjelsbek <skjelsbek@gmail.com>

I had forgotten to ask! Shall we talk tomorrow at 10am?

José Manuel Martins Ferreira
Professor
Universitetet i Sørøst-Norge
Fakultet for teknologi, naturvitenskap og maritime fag
Campus Kongsberg
[Quoted text hidden]



5/20/2019

Gmail - Skype meeting 19. April 2019

Ming Kit Wong <mingkit.wong@gmail.com>

Thu, Apr 25, 2019 at 10:12 PM

To: Jose Ferreira <Jose.Ferreira@usn.no>

Cc: Håvar Østrem <havar.ostrem@gmail.com>, Erica Fegri <ericafegri@gmail.com>, Rita Hogstad <ita95@hotmail.no>, Thomas Mundal <thmundal@gmail.com>, Jon Skjelsbek <skjelsbek@gmail.com>

My fault!

Yes, 10 am if that suits you. The agenda are a general update from us, plus Thomas showing SW layer.

You will get a hang out link before the meeting :)

[Quoted text hidden]

Jose Ferreira <Jose.Ferreira@usn.no>

Thu, Apr 25, 2019 at 10:14 PM

To: Ming Kit Wong <mingkit.wong@gmail.com>

Cc: Håvar Østrem <havar.ostrem@gmail.com>, Erica Fegri <ericafegri@gmail.com>, Rita Hogstad <ita95@hotmail.no>, Thomas Mundal <thmundal@gmail.com>, Jon Skjelsbek <skjelsbek@gmail.com>

Good! Thanks :)

José Manuel Martins Ferreira
Professor
Universitetet i Sørøst-Norge
Fakultet for teknologi, naturvitenskap og maritime fag
Campus Kongsberg

[Quoted text hidden]



5/20/2019

Gmail - Møteagenda + SpaceNight



Thomas Mundal <thmundal@gmail.com>

Møteagenda + SpaceNight

1 message

Ming Kit Wong <mingkit.wong@gmail.com>

Tue, Apr 23, 2019 at 12:53 PM

To: Eirik.Demmo.Normann@kongsberg.com, lars.helge.surdal@kongsberg.com

Cc: Rita Hogstad <ita95@hotmail.no>, Jon Skjelsbek <skjelsbek@gmail.com>, Thomas Mundal <thmundal@gmail.com>, Håvar Østrem <havar.ostrem@gmail.com>, Erica Fegri <ericafegri@gmail.com>

Hei!

Har dere blitt invitert til Space Night på USN av Aage? Se vedlegg hvis ikke.

Agenda for møtet imorgen blir:

- Kort statusoppdatering fra maskin, elektro og data
- Demonstrering av soft ware layer
- Hvis tid: litt om planen for Space Night

mvh

Ming Kit

56161512_565310193878790_1530855644284846080_o.jpg
92K

5/20/2019

Gmail - Møte Onsdag 1. Mai



Thomas Mundal <thmundal@gmail.com>

Møte Onsdag 1. Mai

1 message

Jon Skjelsbek <skjelsbek@gmail.com>

Mon, Apr 29, 2019 at 10:44 AM

To: eirik.demmo.normann@kongsberg.com

Cc: Ming Kit Wong <mingkit.wong@gmail.com>, Thomas Mundal <thmundal@gmail.com>, Erica Fegri <ericafegri@gmail.com>, Håvar Østrem <havar.ostrem@gmail.com>, Rita Hogstad <ita95@hotmail.no>

Hei Eirik,

Førstkommende onsdag er 1. Mai, og i den forbindelse regner jeg med at du har fri. Pga SpaceNight på torsdag og viledermøte på fredagen, ser det ut til at vi må utsette neste møte til onsdag 8. Mai.

Mvh,

Jon, på vegne av gruppen.





Bachelor of Engineering

Project appendix

Winter semester 2019

Jira Export - Epic Reports

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Thomas Mundal and Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains every Epic Report exported from Jira. These reports provides a brief overview of User Stories concerning the given Epic.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

Contents

- A Epic Reports 2**
- A.1 DS Hardware layer 2
- A.2 DS Software layer 5
- A.3 First presentation 7
- A.4 SADM Assembly 9
- A.5 SADM Drivetrain 11
- A.6 SADM Housing 13
- A.7 SADM Motor 15
- A.8 SADM Power transfer 17
- A.9 Second presentation 19

Appendix A

Epic Reports

5/20/2019

SP board - Agile Board - USN Bachelor group 5

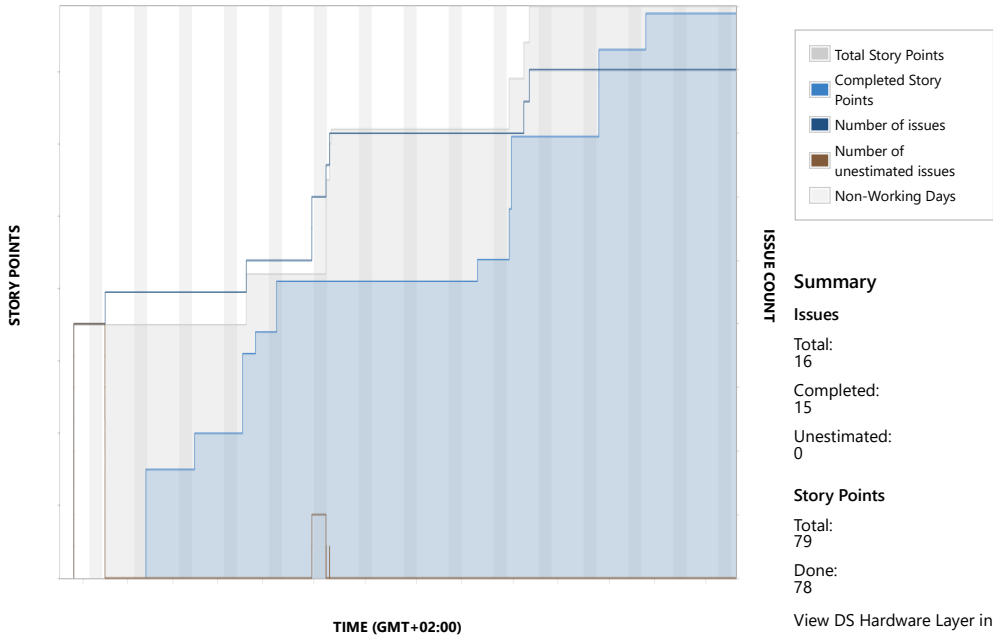


Projects / USN Bachelor group 5 / SP board / Reports

Epic Report

DS Hardware Layer

USN5-95 Hardware layer of the Diagnostics System
6 linked pages



Status Report



Completed Issues

[View in Issue Navigator](#)
















Key	Summary	Issue Type	Priority	Status	Story Points (78)
USN5-101	As an embedded developer, I need the receiving end of my hardware to speak the same language as the software layer in order to properly interpret the received instructions.	Story	Medium	DONE	5
USN5-98	As an embedded developer, I need to make sure that instructions can be received even though another instruction is currently executing.	Story	Medium	DONE	5
USN5-99	As an embedded developer, I need to adapt the instructions to the proper format for the receiving device to understand.	Story	Medium	DONE	5
USN5-232	For documentation purposes I need to finalize the documents related to Alpha 2 HWL so that they're ready for hand in.	Story	Medium	DONE	7
USN5-97	As an embedded developer, I need to provide sensible sensor data to verify that the systems behaves as expected.	Story	Medium	DONE	3
USN5-199	As an electrical engineer, I should design first prototype for the cabinet so that we can review this for costumer.	Story	Medium	DONE	5
USN5-96	As an embedded developer, I need my hardware to be able to control the SADM to be able to measure the result.	Story	Medium	DONE	3
USN5-102	As an embedded developer, I need the transmitting end of my hardware to speak languages that the receiving devices understand in order to transmit data in the correct format.	Story	Medium	DONE	3
USN5-100	As an embedded developer, I need to make sure that the correct instructions are executed at the correct device for the test results to be correct.	Story	Medium	DONE	5
USN5-257	As the HWL has to function at all times, error handling must be implemented to ensure that the system doesn't crash.	Story	Medium	DONE	3

<https://usnbachelor.atlassian.net/secure/RapidBoard.jspa?rapidView=2&projectKey=USN5&view=reporting&chart=epicReport&epic=USN5-95>

1/2



5/20/2019

SP board - Agile Board - USN Bachelor group 5

		USN5-251	As an embedded developer, I need to implement functionality supporting a subscriber system for sensor data to comply with the SatStat communication protocol.	 Story	 Medium	DONE	10
		USN5-250	As an embedded developer, I need to optimize the current version of the HWL to prepare for further development.	 Story	 Medium	DONE	7
		USN5-311	As a user of the HMI I want a list of available instructions so that I don't have to remember all of them	 Story	 Medium	DONE	5
		USN5-263	As a successor of the project, I need descriptive documents/diagrams of the HWL to further develop the system.	 Story	 Medium	DISCARDED	7
		USN5-290	Complete order of electrical equipment.	 Story	 Medium	DONE	5

Incomplete Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (1)
USN5-103	As an embedded developer, I need to make sure my hardware doesn't affect the SADM as they are two separate systems.	 Story	 Medium	OPEN	1



5/20/2019

SP board - Agile Board - USN Bachelor group 5



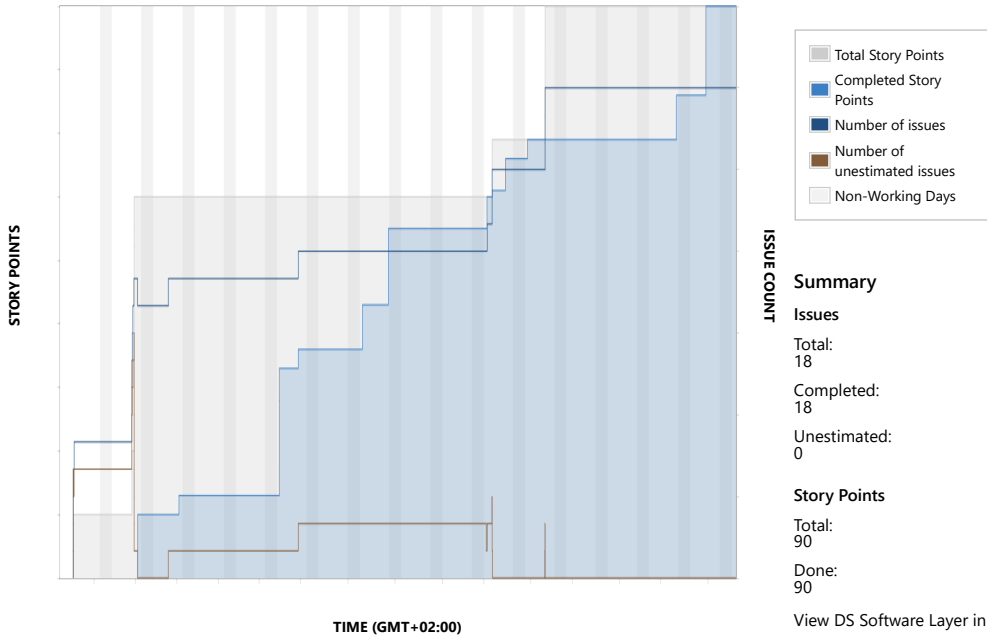
Projects / USN Bachelor group 5 / SP board / Reports

Epic Report



DS Software Layer

USN5-78 Software layer of Diagnostics System
5 linked pages



Status Report



Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (90)
USN5-50	As a test engineer, I want a software that can monitor the state of the system to verify requirements	Story	Medium	DISCARDED	10
USN5-74	As a test engineer, I need a software that can give me diagnostic data in real time	Story	High	DONE	5
USN5-108	As a software engineer, I need the diagnostic system to use a standardized way of exchanging data	Story	High	DONE	3
USN5-77	As a software engineer, I need the diagnostic system to be easily expandable to allow for other diagnostics components	Story	High	DONE	10
USN5-131	As a test engineer, I need the ability to calibrate position and speed of the SADM drive shaft so that I can set correct initial conditions for operation	Story	High	DONE	3
USN5-76	As a test engineer, I need a simple way of configuring the communication between the diagnostics system software and hardware	Story	Medium	DONE	5
USN5-205	The diagnostic system should include several functions so that it will satisfy the customer needs	Story	Medium	DISCARDED	-
USN5-75	As a test engineer, I need a software that can display historical diagnostic data	Story	Medium	DONE	7
USN5-87	As a test engineer, I need an easy and intuitive way of choosing which diagnostics data I want to look at	Story	Medium	DONE	5
USN5-146	As a test engineer, I would like to not have to adjust COM settings every time I have to connect to the HW layer so that the software is easier to use	Story	Medium	DONE	5

5/20/2019

SP board - Agile Board - USN Bachelor group 5

		USN5-247	Implement error handling and connection timeouts for SWL	 Story	 Medium	DISCARDED	-
		USN5-275	As a test engineer, I would like to see the current value of a sensor input so that I can keep an eye on the current status of the system	 Story	 Medium	DONE	1
		USN5-279	As a test engineer, I need the ability to adjust accepted values for sensor readings to be able to verify that the system is operating as intended	 Story	 Medium	DONE	5
		USN5-281	As a test engineer, I want the ability to save accepted sensor value ranges to a template, so that I can load the same settings for testing at any point in time	 Story	 Medium	DONE	3
		USN5-302	As a test engineer I want the ability to define behaviour and parameters to measure during a test run for the SADM	 Story	 Medium	DONE	7
		USN5-304	As a test engineer I want the ability to load test-run settings from a template and initiate a test run based on loaded or defined parameters	 Story	 Medium	DONE	7
		USN5-303	As a test engineer I want the ability to save a defined test-run as a template so that I can load the same settings for a future test run	 Story	 Medium	DONE	7
		USN5-118	As a software engineer, I need a standardized interface for communicating with a database so that the system can save historical data	 Story	 Medium	DONE	7



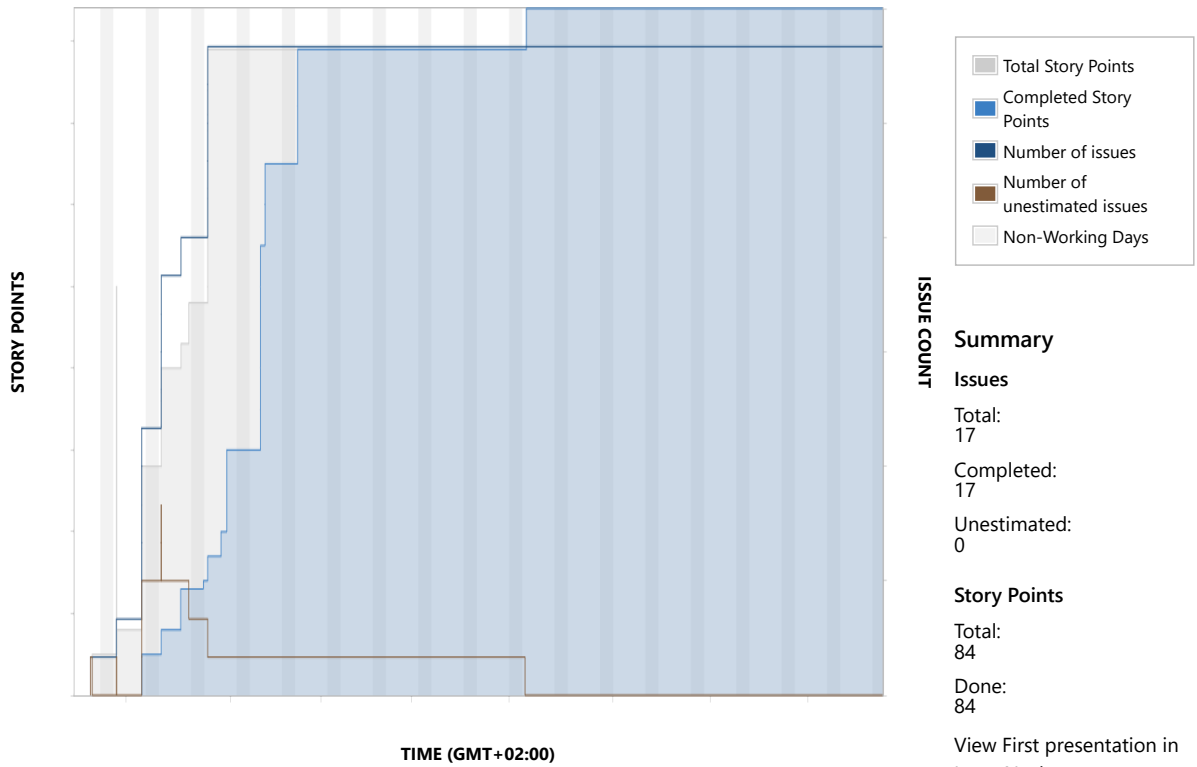
Projects / USN Bachelor group 5 / SP board / Reports

Epic Report

First presentation

USN5-13 Work towards first presentation

Linked pages



Status Report

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (84)
USN5-10	The group needs a project model in order to complete the project	Story	Highest	DONE	5
USN5-22	As a product owner i need to be familiar with my tasks so that i can satisfy the role description	Story	Highest	DONE	3
USN5-51	The group needs to form and agree upon a description of the task in a document	Story	Medium	DONE	5
USN5-55	The group needs a name	Story	Medium	DONE	3
USN5-67	Requirement analysis	Story	Highest	DONE	7
USN5-68	Define verification methods for requirements	Story	Highest	DONE	7
USN5-71	Revise the task description	Story	Medium	DONE	5
USN5-56	We need a script draft for the first presentation	Story	Highest	DONE	5
USN5-48	The group needs to deliver documents two days before the presentation because it's required from the sensors	Story	High	DONE	10
USN5-57	We need a finalized script for the first presentation	Story	High	DONE	5
USN5-80	Finish the slides for the presentation	Story	High	DONE	5

USN5-43	The process model needs to incorporate process retrospectives so that the group can uncover challenges related to the project generally	Story	↑	Medium	DONE	1
USN5-45	The group needs a first prototype so that we can show this at the first presentation	Story	↑	Medium	DONE	10
USN5-64	Do stakeholder analysis	Story	↑	Medium	DONE	5
USN5-41	Get documentation/scope from KONGSBERG Space	Story	↑	High	DONE	-
USN5-79	Make a logo	Story	↑	Medium	DONE	3
USN5-49	As the one responsible for the presentation tool, the assigned needs to be familiar with Prezi or other presentation tools.	Story	↓	Low	DONE	5



5/20/2019

SP board - Agile Board - USN Bachelor group 5

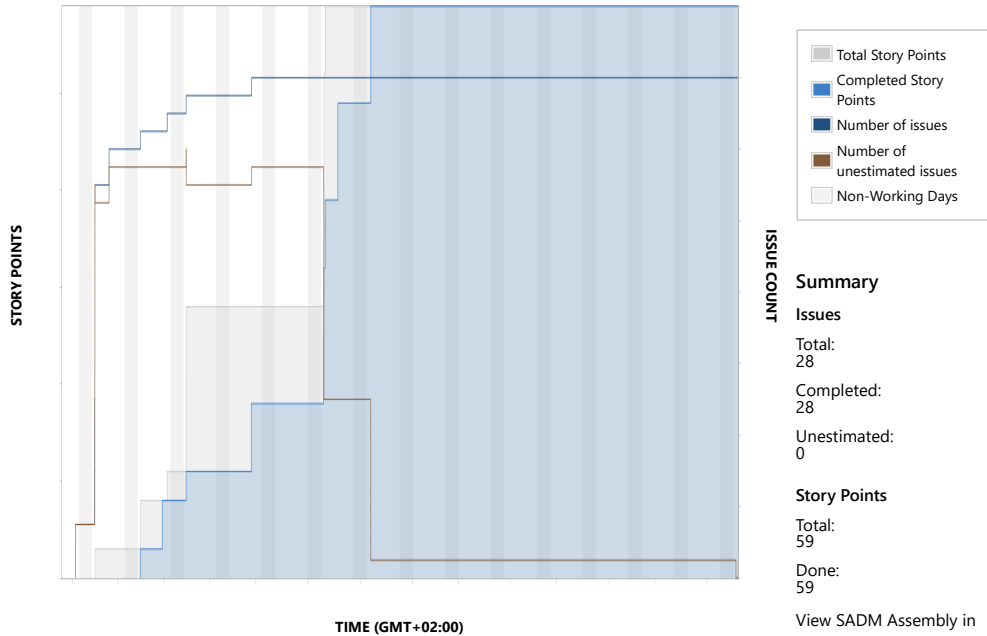


Projects / USN Bachelor group 5 / SP board / Reports

Epic Report

SADM Assembly

USN5-152 User stories and requirements concerning the whole SADM
Linked pages



Status Report



Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (59)
USN5-193	As a mechanical engineer, I need to start looking at solutions of design that are suitable for us	Story	Medium	DONE	3
USN5-185	As a mechanical engineer, I must take into account impacts and effects of environmental radiation etc. in a reasonable way to prepare the SADM prototype for the environment in space.	Story	Medium	DONE	-
USN5-157	The SADM should be able to connect accordingly to the satellite and the solar arrays	Story	Medium	DISCARDED	-
USN5-172	As a mechanical and electrical engineer, I need to design the SADM to withstand vibrations to ensure functionality trough it's lifecycle.	Story	Medium	DISCARDED	-
USN5-166	The SADM and related equipment shall be designed for operation at given temprature range, so that the environmental temperature does not harm the functionality.	Story	Medium	DISCARDED	-
USN5-167	As an electrical and mechanical engineer, I have to design the SADM so that functionality and operation will not be harmed due to humidity during it's life cycle.	Story	Medium	DISCARDED	-
USN5-168	As mechanical and electrical engineer, I shuld design the SADM to operate under given values for pressure to ensure functionality during it's entire life cycle.	Story	Medium	DISCARDED	-
USN5-169	As a mechanical engineer, I need to be familiar with \$4.7.5.2 in ECSS-E-ST-33-01 so that we can we can define the structural analysis for the SADM.	Story	Medium	DISCARDED	-
USN5-173	As a mechanical engineer, I need to design the SADM to withstand random vibrations so that the SADM is suited for environmental impacts.	Story	Medium	DISCARDED	-

<https://usnbachelor.atlassian.net/secure/RapidBoard.jspa?rapidView=2&projectKey=USN5&view=reporting&chart=epicReport&epic=USN5-152>

1/2

5/20/2019

SP board - Agile Board - USN Bachelor group 5

	USN5-174	As a mechanical engineer, I need to consider how the SADM should be able to withstand shock.	Story	Medium	DISCARDED	-
	USN5-175	As a mechanical and electrical engineer, I should design the SADM connectors and interfaces to be operative in given temperature range.	Story	Medium	DISCARDED	-
	USN5-176	As a mechanical engineer, I should design the SADM mechanically considering the heat flow from the SADM, so that the amount of heat flow will not have a negative impact to the rest of the spacecraft and it's functionality.	Story	Medium	DISCARDED	-
	USN5-177	As a mechanical and electrical engineer, I should design the SADM mechanically considering the heat flow from the SADM to the solar arrays, so that the amount of heat flow will not have a negative impact to the solar arrays.	Story	Medium	DISCARDED	-
	USN5-178	As a mechanical engineer, I should decide if we are going to do a thermal analysis and define the necessary equipment to this analysis.	Story	Medium	DISCARDED	-
	USN5-180	As a electrical and mechanical engineer, I need to ensure contact between each part so that all mechanical parts of the SADM has he same electrical potential.	Story	Medium	DISCARDED	-
	USN5-182	As a test engineer and electrical engineer, I must make sure the bonding test is preformed the specified way to ensure the verification is preformed correctly.	Story	Medium	DISCARDED	-
	USN5-184	As a mechanical engineer, I must design the required shielding for the SADM (EEE)	Story	Medium	DISCARDED	-
	USN5-187	As a mechanical engineer, I have to be familiar with standards for spacegrade compability and decide how and if the SADM will ensure vacuum compability.	Story	Medium	DISCARDED	-
	USN5-188	As a mechanical engineer, I should ensure bimetallic compability to avoid corrosion and other inconveniences.	Story	Medium	DISCARDED	-
	USN5-189	As mechanical and electrical engineers, we shall take radiation into consideration when selecting materials for the SADM to be able to withstand the radiation levels for orbit and lifetime.	Story	Medium	DISCARDED	10
	USN5-194	As electrical and mechanical engineers, we have to design the SADM to meet requirements for reliability to ensure functionality trough the SADMs entire lifetime.	Story	Medium	DISCARDED	-
	USN5-212	As a mech. engineer, I need to talk with our external supervisor to clarify some requirements for design	Story	Medium	DONE	5
	USN5-204	The SADM should include all the necessary components to function	Story	High	DONE	7
	USN5-222	As designers, we need to decide the final design of our product so we can start choosing materials and setting right dimensions	Story	High	DONE	10
	USN5-246	The SADM's weight should not exceed 2kg so that the nano satellite will not be affected by the SADM weight	Story	Medium	DONE	7
	USN5-153	The SADM's weight should not exceed 2kg so that the nano satellite will not be affected by the SADM weight - For Mark2	Story	Medium	DONE	7
	USN5-154	The volume of the SADM should not exceed 100 x 100 x 200 millimeters so that it will fit inside of an 16u satellite	Story	Medium	DONE	7
	USN5-221	As a mechanical engineer, I need to start doing research on materials so I can be familiar with the possibility of use for coming prototypes	Story	Medium	DONE	3

5/20/2019

SP board - Agile Board - USN Bachelor group 5



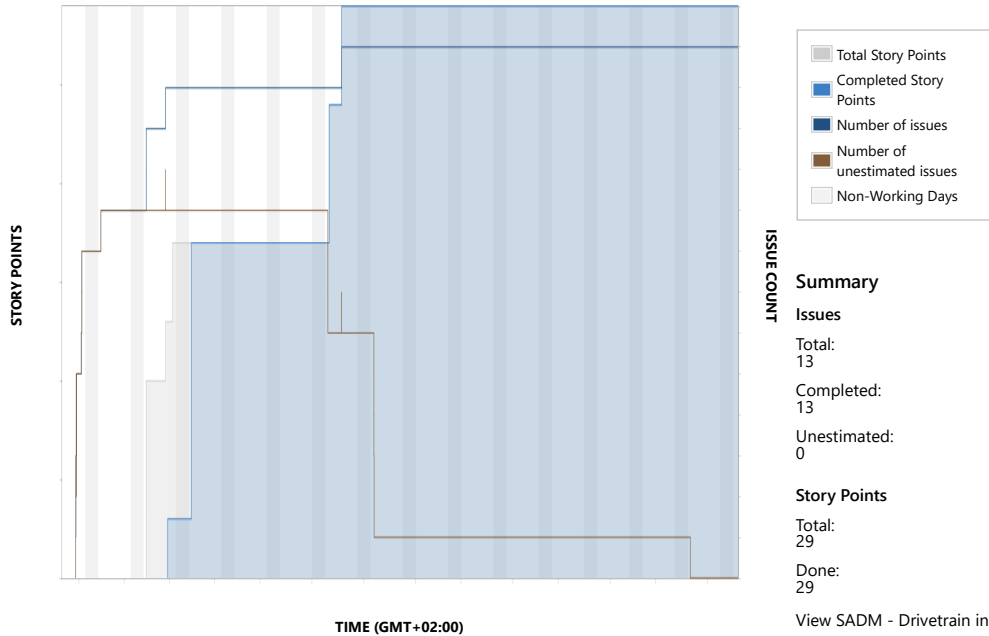
Projects / USN Bachelor group 5 / SP board / Reports

Epic Report

SADM - Drivetrain

USN5-111 SADM - Drivetrain

1 linked page



Status Report



Completed Issues

[View in Issue Navigator](#)














Key	Summary	Issue Type	Priority	Status	Story Points (29)
USN5-211	As a mechanical engineer, I need to calculate the total moment of inertia that affects our product to make a proper design	Story	Medium	DONE	3
USN5-119	As a mechanical engineer I have to design the shaft so that solar arrays can be attached to the spacecraft.	Story	Medium	DISCARDED	-
USN5-124	The design of interface between solar array and SADM and motor must be adapted to the weight and size of solar arrays so that it functions as intended.	Story	Medium	DISCARDED	-
USN5-134	The drivetrain shall keep its position under spacecraft rotational accelerations so that the solar arrays holds it's positions.	Story	Medium	DISCARDED	-
USN5-135	The drive train should have a life time of 5 years in an orbit of 600km so that it will satisfy the customers	Story	Medium	DISCARDED	-
USN5-140	The driveline should be single point failure free so that it will be more independent and the whole SADM won't stop working if one component stops working	Story	Medium	DISCARDED	-
USN5-151	The power output, the angular velocity and velocity resolution from the motor, as well as the gearing ratio, gear stiffness, and shaft stiffness should all be of adequate capacity for the SADM to rotate the shaft connected to the solar arrays with adjust...	Story	Medium	DISCARDED	-
USN5-155	The total mass of the SADM, excluding wiring, must be two kilograms or less. This means the drivetrain can only be a certain percentage of this mass and other massive components must be taken into consideration when designing the drivetrain.	Story	Medium	DONE	-
USN5-171	As a mechanical engineer, I need to design the interface between SADM and solar arrays so that it can withstand given values for forces.	Story	Medium	DISCARDED	-

<https://usnbachelor.atlassian.net/secure/RapidBoard.jspa?rapidView=2&projectKey=USN5&view=reporting&chart=epicReport&epic=USN5-111>

1/2

5/20/2019

SP board - Agile Board - USN Bachelor group 5

		USN5-213 As a mech. engineer, I need to cooperate with an electrical engineer to consider different designs to decide a suitable i/f between the flex and the shaft	 Story	 Medium	DONE	7
		USN5-156 The outer spatial dimensions of The Housing of the SADM must be exactly one hundred by one hundred by two hundred millimeters. The inner spatial dimensions depends on the thickness of the housing walls. The drivetrain and other components inside the SADM...	 Story	 Medium	DONE	7
		USN5-220 As a mechanical engineer I need to consider materials to use on the shaft so that it will be compatible in space environment	 Story	 Medium	DONE	7
		USN5-269 As the test manager, I need to check that the VMD is up to date	 Story	 Medium	DONE	5

 9







5/20/2019

SP board - Agile Board - USN Bachelor group 5

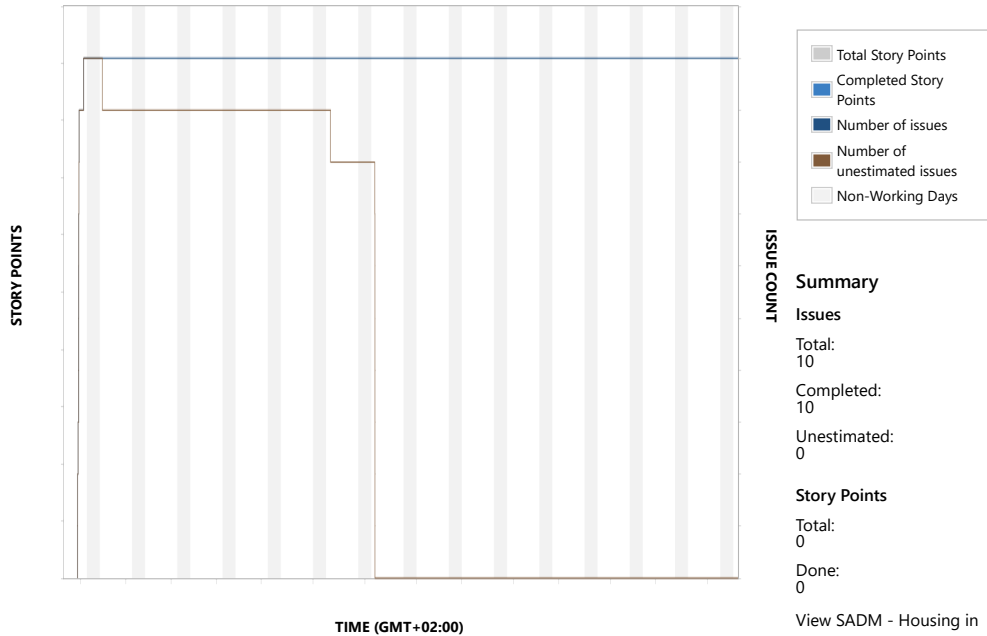


Projects / USN Bachelor group 5 / SP board / Reports

Epic Report

SADM - Housing

USN5-113 SADM - Housing
1 linked page



Status Report

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (0)
USN5-123	The SADM physical reference should be illustrated in a sketch so that it's orientation is easy to understand.	Story	Medium	DONE	-
USN5-130	The SADM shall be able to oscillate or rotate 180 degrees around the X- axis in both directions so that it always can point the solar arrays against the sun	Story	Medium	DISCARDED	-
USN5-138	The housing should have a life time of 5 years in an orbit of 660km so that it will satisfy the customers	Story	Medium	DISCARDED	-
USN5-142	The SADM shall provide an axial translation stiffness of $>- 1.0 \times 10^8$ N/m	Story	Medium	DISCARDED	-
USN5-143	The SADM shall provide a radial translation stiffness of $>- 1.0 \times 10^8$ N/m with the solar array interface constrained in cross axis bending (interface plane remains parallel under loading)	Story	Medium	DISCARDED	-
USN5-144	The SADM shall provide a cross axis bending stiffness of $>- 1.0 \times 10^5$ Nm/rad	Story	Medium	DISCARDED	-
USN5-145	The SADM shall provide a torsional stiffness of $>- 10^4$ Nm/rad (assuming a blocked motor shaft, at 5 to 15 Nm external load on the output shaft)	Story	Medium	DISCARDED	-
USN5-147	Our product is to be mounted on a 8u-dummy, with the center of gravity placed at minimum 20 mm from the SADM interface, during vibration testing. The SADM shall have a first Eigen-frequency greater than 250 Hz to fulfill this requirement.	Story	Medium	DISCARDED	-
USN5-149	The housing needs to include venting paths so that there will be low pressure in the SADM	Story	Medium	DISCARDED	-
USN5-150	The materials of the housing should meet the requirement of	Story	Medium	DISCARDED	-

<https://usnbachelor.atlassian.net/secure/RapidBoard.jspa?rapidView=2&projectKey=USN5&view=reporting&chart=epicReport&epic=USN5-113>

1/2

5/20/2019

SP board - Agile Board - USN Bachelor group 5

Total Ionising Dose (TID) level of 10 Mrad (Milliradians)



5/20/2019

SP board - Agile Board - USN Bachelor group 5



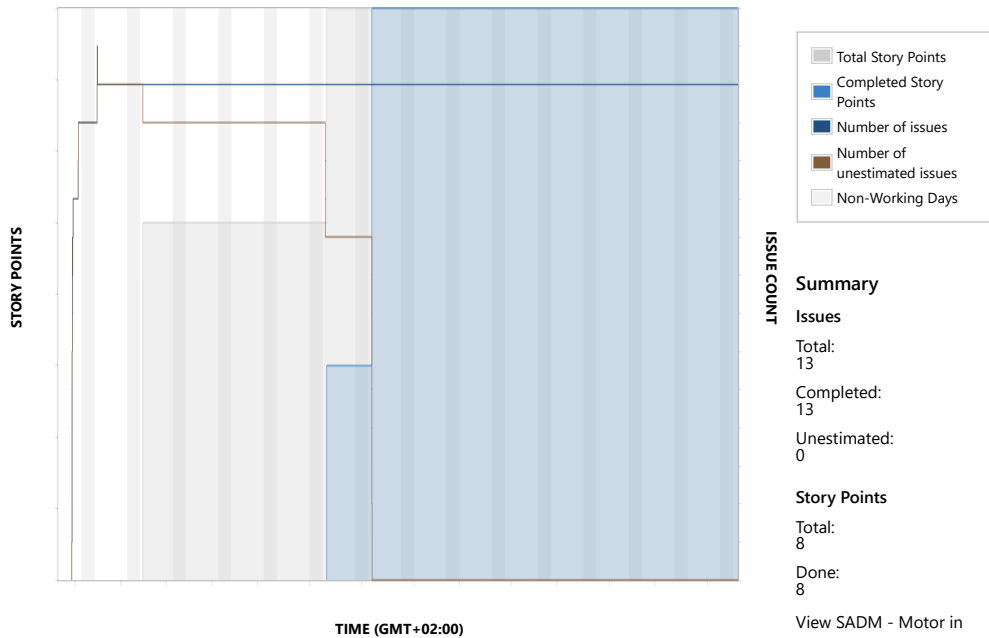
Projects / USN Bachelor group 5 / SP board / Reports

Epic Report

SADM - Motor

USN5-112 SADM - Motor

1 linked page



Status Report













Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (8)
USN5-120	Our product needs to hold the solar array wings in position when not powered	Story	Medium	DISCARDED	-
USN5-125	Our product needs to perform and function with the given values for a solar array.	Story	Medium	DISCARDED	-
USN5-126	The European standard for the space industry must be followed when selecting and designing the placement of the motor. Other requirements from the specified standard must be followed as well.	Story	Medium	DISCARDED	-
USN5-127	The motor must be able to hold the solar array in the same position with a load of 2Nm or greater, when it is not unpowered.	Story	Medium	DISCARDED	-
USN5-128	The motor must be able to hold the solar array in the same position with a load of 8 Nm, when the motor is running with a current of 150 mA.	Story	Medium	DISCARDED	-
USN5-129	The mechanism must allow movement when load exceeds 70 Nm	Story	Medium	DISCARDED	-
USN5-133	The shaft is rotating with a speed that must have a stability greater than the values given in the requirements.	Story	Medium	DISCARDED	-
USN5-136	The motor shall keep its position under spacecraft rotational accelerations so that the solar arrays holds it's positions.	Story	Medium	DISCARDED	-
USN5-137	The motor should have a life time of 5 years in an orbit of 660km so that it will satisfy the customers	Story	Medium	DISCARDED	-
USN5-141	The SADM shall have enough torque margins to withstand a short circuit of motor winding without loss of functionality so that operation continues uninterrupted.	Story	Medium	DISCARDED	-

5/20/2019

SP board - Agile Board - USN Bachelor group 5

		USN5-162	The motors interface should be designed/chosen by standard sizes(NEMA) so that it will be easier to connect to other standard products	 Story	 Medium	DISCARDED	-
		USN5-161	The motor should have specific nominal electrical interfaces so that it will satisfy the SADM power needs	 Story	 High	DISCARDED	5
		USN5-158	The motor of the SADM should be marked with a zero reference point so that it will be easier to mate the assembly	 Story	 Medium	DONE	3

-  9
- 
- 
- 

5/20/2019

SP board - Agile Board - USN Bachelor group 5

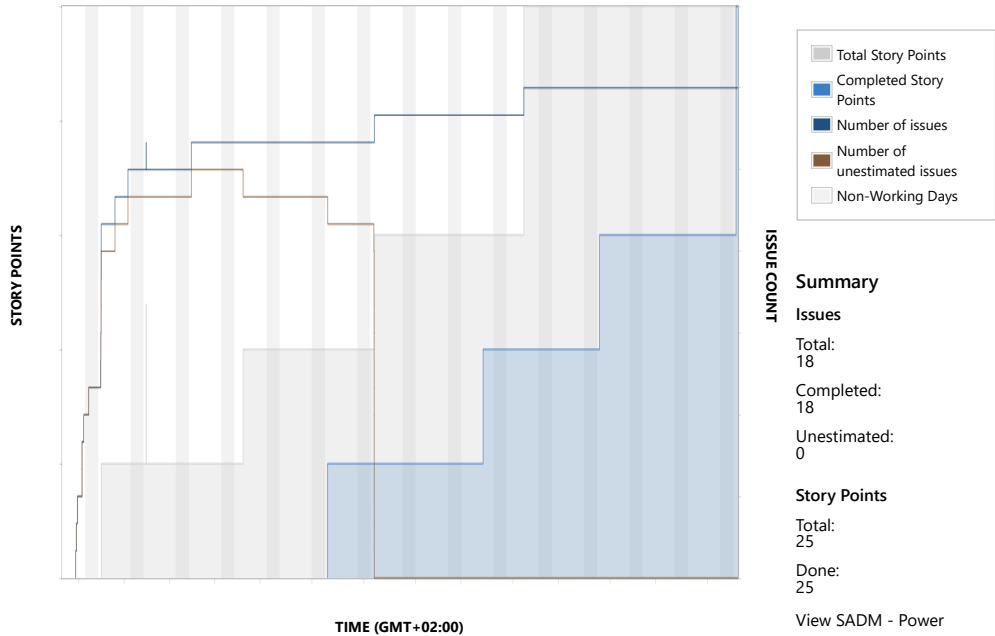


Projects / USN Bachelor group 5 / SP board / Reports

Epic Report

SADM - Power transfer

USN5-114 SADM - Power transfer
1 linked page



Status Report


























Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (25)
USN5-122	The SADM shall transfer solar array grounding lines to ensure grounding	Story	Medium	DISCARDED	-
USN5-139	The power transfer system should have a life time of 5 years in an orbit of 660km so that it will satisfy the customers	Story	Medium	DISCARDED	-
USN5-148	As an electrical engineer, I need to make sure the required derating performances of the power tranfers will be adequate	Story	Medium	DISCARDED	-
USN5-159	As an electrical engineer, I shall make sure the rotor harness have 500mm flying leads so that it will be easy to connect the leads to future connectors.	Story	Medium	DISCARDED	-
USN5-160	As an electrical engineer, I need to make sure the grounding lines from the solar arrays are seperated from the SADM grounding.	Story	Medium	DISCARDED	-
USN5-163	As an electrical engineer, I need to make sure there are connectors on the stator side to be able to tranfer power and signals from/to the SADM and rest of the spacecraft.	Story	Medium	DISCARDED	-
USN5-164	As an electrical engineer, I have to ensure sufficient grounding to maintain optimal operation of the SADM/spacecraft	Story	Medium	DISCARDED	-
USN5-165	The SADM power consumption shall not exceed a certain level, so that the spacecraft's total power consumption does not exceed the total capacity.	Story	Medium	DISCARDED	-
USN5-181	As a electrical engineer, I should make sure the potential between metallic parts and intrinsically conductive parts that do not perform any electrical funtion is grounded according to specifications.	Story	Medium	DISCARDED	-

5/20/2019

SP board - Agile Board - USN Bachelor group 5

		USN5-183	As a test engineer, I must preform the bonding test without any harness connection to ensure no equipment will take damage and that the test is preformed correctly.	 Story	 Medium	DISCARDED	-
		USN5-195	As an electrical engineer, I should derate the electrical components so that the will function optimal and not take any damage due to overload, heat and other.	 Story	 Medium	DISCARDED	-
		USN5-196	As an electrical engineer, I should make sure the power transfers meets acceptable derating performance, so that the power transfer can still be operative in failure cause.	 Story	 Medium	DISCARDED	-
		USN5-198	As an electrical engineer, I must be familiar with flex-suppliers and characteristics needed for the flex so that we can decide what spesifications are needed.	 Story	 High	DONE	5
		USN5-209	The SADM power transfer system should be derated so that if a failure happens it won't affect the other components in the assembly	 Story	 Medium	DISCARDED	-
		USN5-210	As an electrical engineer, I need to design the cables, flex and equipment to have adequate conductivity.	 Story	 Medium	DISCARDED	-
		USN5-121	Information about the solar array position shall be available so that the rest of the system (DS) can use this information.	 Story	 Medium	DONE	5
		USN5-298	Design connection interface between flex and wires.	 Story	 Medium	DONE	10
		USN5-271	As an electrical engineer, I must make the first physical prototype of the connection between flex and wires.	 Story	 High	DONE	5



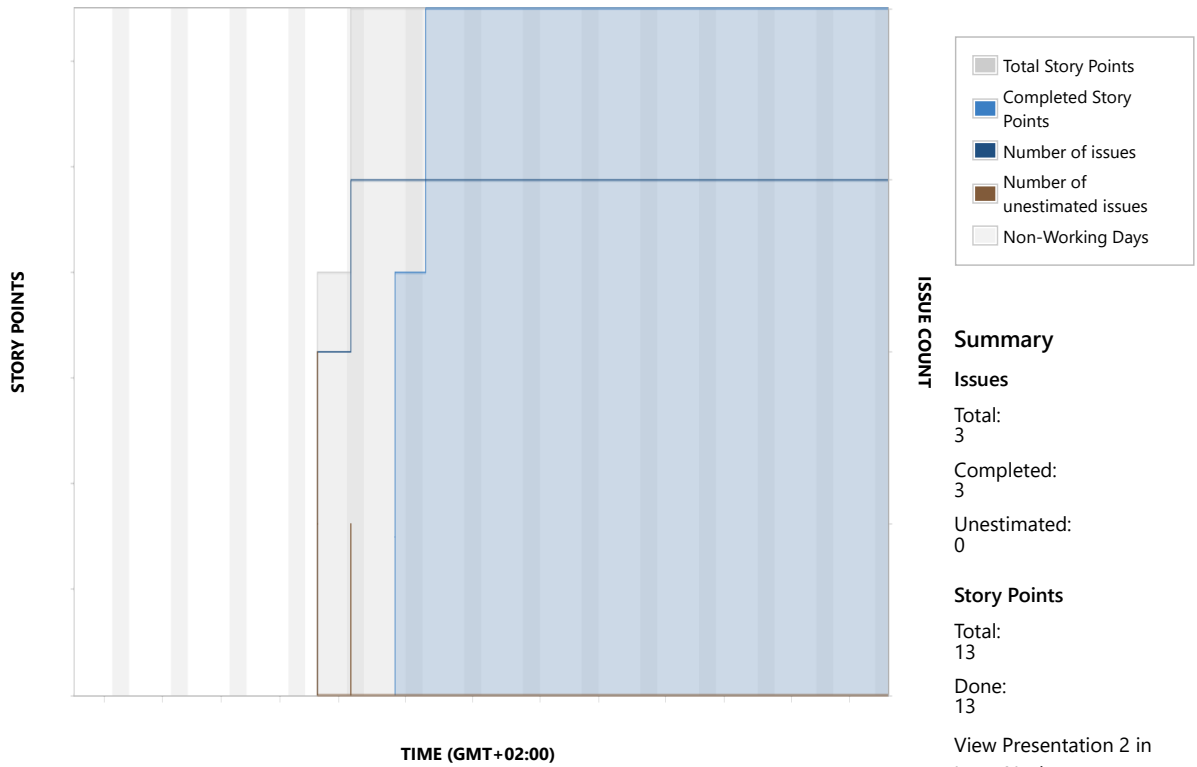
Projects / USN Bachelor group 5 / SP board / Reports

Epic Report

Presentation 2

USN5-92 Second presentation, middle of March

Linked pages



Status Report

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (13)
USN5-237	As a group we need to prepare a script for presentation 2	Story	Medium	DONE	5
USN5-238	As a group we need to prepare slides for presentation 2	Story	Medium	DONE	3
USN5-248	As mechanical engineers, we need to write documentation for our work	Story	Medium	DONE	5

Bachelor of Engineering

Project appendix

Winter semester 2019

Jira Export - Version Reports

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Thomas Mundal and Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains every Version Report exported from Jira. These reports provides a brief overview of User Stories concerning the given Version.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

Contents

- A Version Reports** **2**
- A.1 DS Alpha 1 2
- A.2 DS Alpha 2 5
- A.3 DS Alpha 3 7
- A.4 DS Beta 1 8
- A.5 DS Beta 2 10
- A.6 Mark 1 11
- A.7 Mark 2 13
- A.8 Mark 3 17

Appendix A

Version Reports

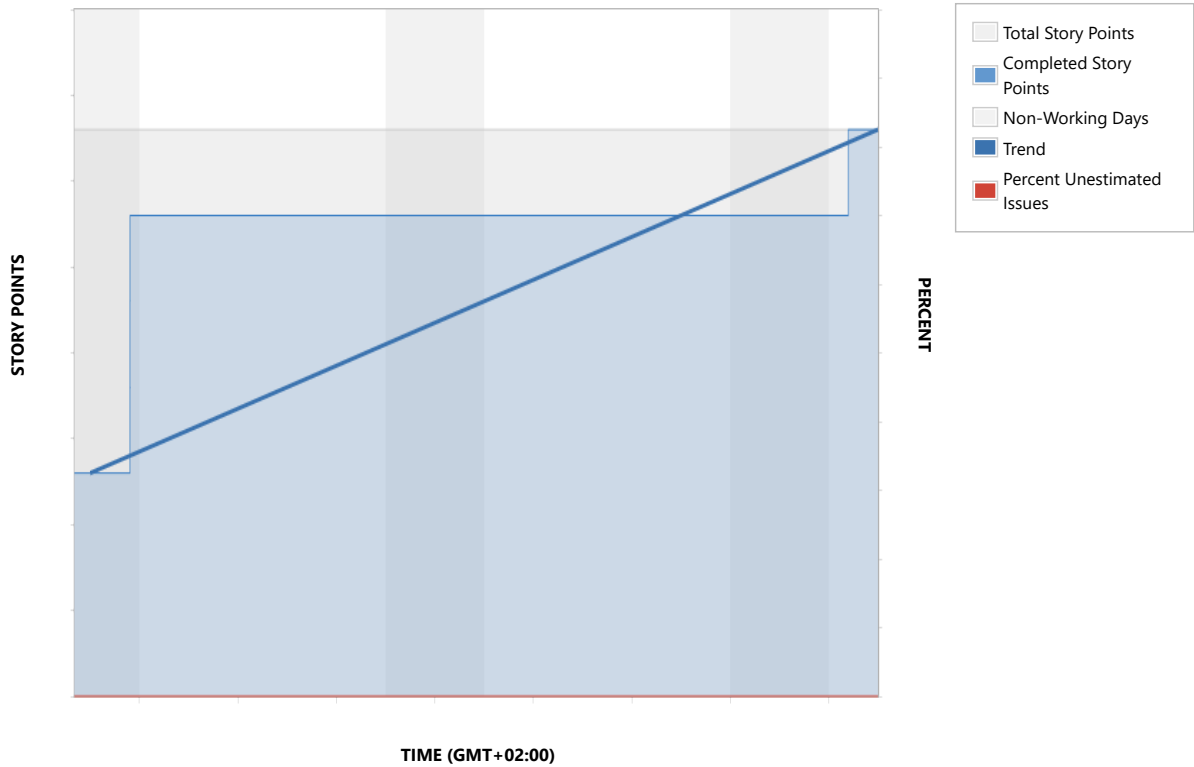
Projects / USN Bachelor group 5 / SP board / Reports

Version Report

USN5: SatStat alpha 1

Start Date: 17/Feb/19
Released on 04/Mar/19

[View SatStat alpha 1 in Issue Navigator](#)








Status Report

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Epic	Story Points (33)
USN5-50	As a test engineer, I want a software that can monitor the state of the system to verify requirements	Story	Medium	DISCARDED	DS Software Layer	10
USN5-52	As a software engineer, I need a piece of hardware that can give me sensor data from the SADM	Story	Medium	DISCARDED		-
USN5-53	As a software engineer, I need a piece of hardware that can give me state information about the SADE	Story	Medium	DISCARDED		-
USN5-74	As a test engineer, I need a software that can give me diagnostic data in real time	Story	High	DONE	DS Software Layer	5
USN5-108	As a software engineer, I need the diagnostic system to use a standardized way of exchanging data	Story	High	DONE	DS Software Layer	3
USN5-101	As an embedded developer, I need the receiving end of my hardware to speak the same language as the software layer	Story	Medium	DONE	DS Hardware Layer	5



	in order to properly interpret the received instructions.					
USN5-98	As an embedded developer, I need to make sure that instructions can be received even though another instruction is currently executing.	 Story	 Medium	DONE	DS Hardware Layer	5
USN5-99	As an embedded developer, I need to adapt the instructions to the proper format for the receiving device to understand.	 Story	 Medium	DONE	DS Hardware Layer	5

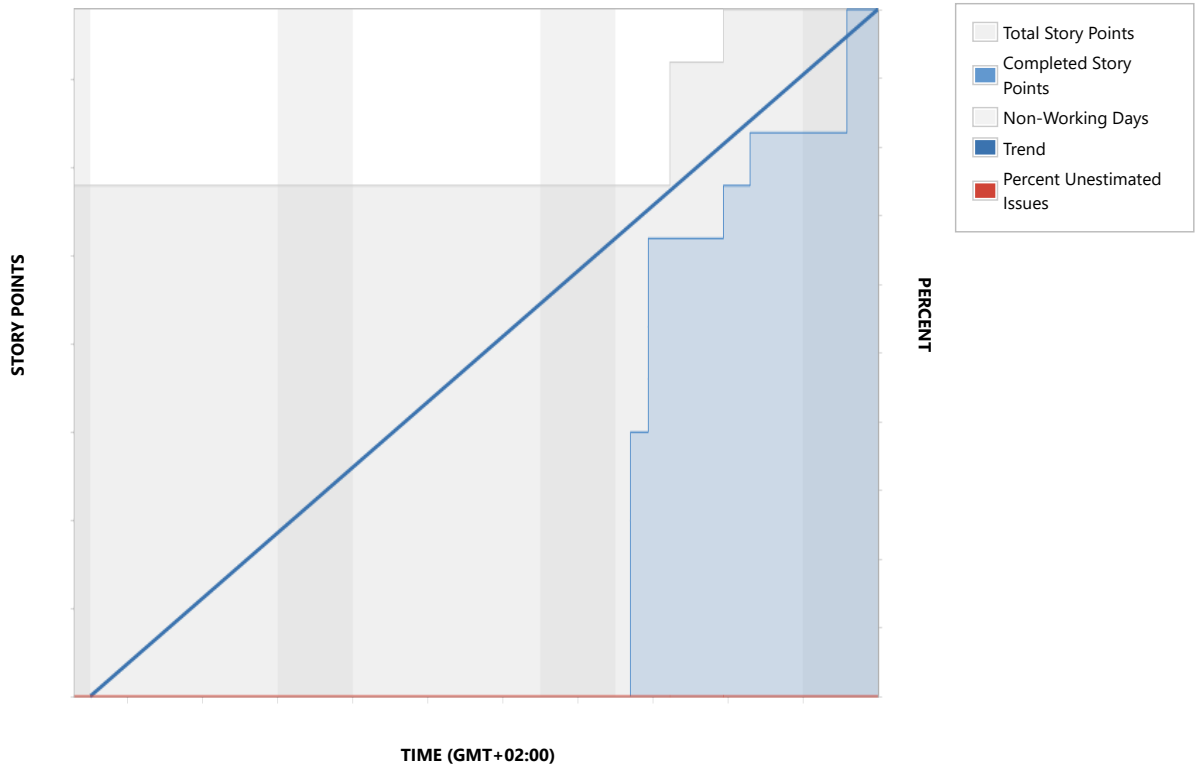
Projects / USN Bachelor group 5 / SP board / Reports

Version Report

USN5: SatStat Alpha 2

Start Date: 18/Feb/19
Released on 10/Mar/19

[View SatStat Alpha 2 in Issue Navigator](#)












Status Report

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Epic	Story Points (39)
USN5-77	As a software engineer, I need the diagnostic system to be easily expandable to allow for other diagnostics components	Story	High	DONE	DS Software Layer	10
USN5-232	For documentation purposes I need to finalize the documents related to Alpha 2 HWL so that they're ready for hand in.	Story	Medium	DONE	DS Hardware Layer	7
USN5-131	As a test engineer, I need the ability to calibrate position and speed of the SADM drive shaft so that I can set correct initial conditions for operation	Story	High	DONE	DS Software Layer	3
USN5-76	As a test engineer, I need a simple way of configuring the communication between the diagnostics system software and hardware	Story	Medium	DONE	DS Software Layer	5
USN5-97	As an embedded developer, I need to provide sensible sensor data to verify that the systems behaves as expected.	Story	Medium	DONE	DS Hardware Layer	3
USN5-96	As an embedded developer, I need my hardware to be able to control the SADM to	Story	Medium	DONE	DS Hardware Layer	3

be able to measure the result.

   		USN5-102 As an embedded developer, I need the transmitting end of my hardware to speak languages that the receiving devices understand in order to transmit data in the correct format.	 Story  Medium DONE DS Hardware Layer	3
		USN5-100 As an embedded developer, I need to make sure that the correct instructions are executed at the correct device for the test results to be correct.	 Story  Medium DONE DS Hardware Layer	5



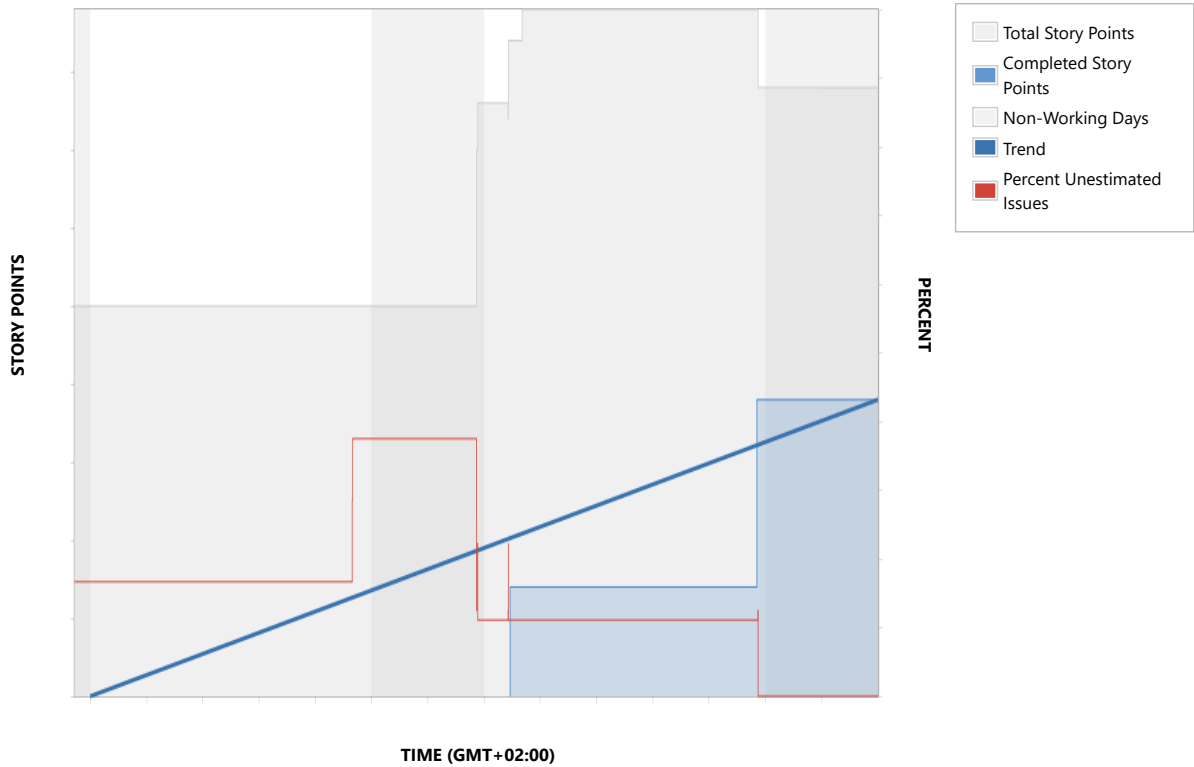
Projects / USN Bachelor group 5 / SP board / Reports

Version Report

USN5: SatStat Alpha 3

Start Date: 11/Mar/19
Released on 24/Mar/19

[View SatStat Alpha 3 in Issue Navigator](#)



Status Report

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Epic	Story Points (22)
USN5-75	As a test engineer, I need a software that can display historical diagnostic data	Story	Medium	DONE	DS Software Layer	7
USN5-87	As a test engineer, I need an easy and intuitive way of choosing which diagnostics data I want to look at	Story	Medium	DONE	DS Software Layer	5
USN5-257	As the HWL has to function at all times, error handling must be implemented to ensure that the system doesn't crash.	Story	Medium	DONE	DS Hardware Layer	3
USN5-118	As a software engineer, I need a standardized interface for communicating with a database so that the system can save historical data	Story	Medium	DONE	DS Software Layer	7

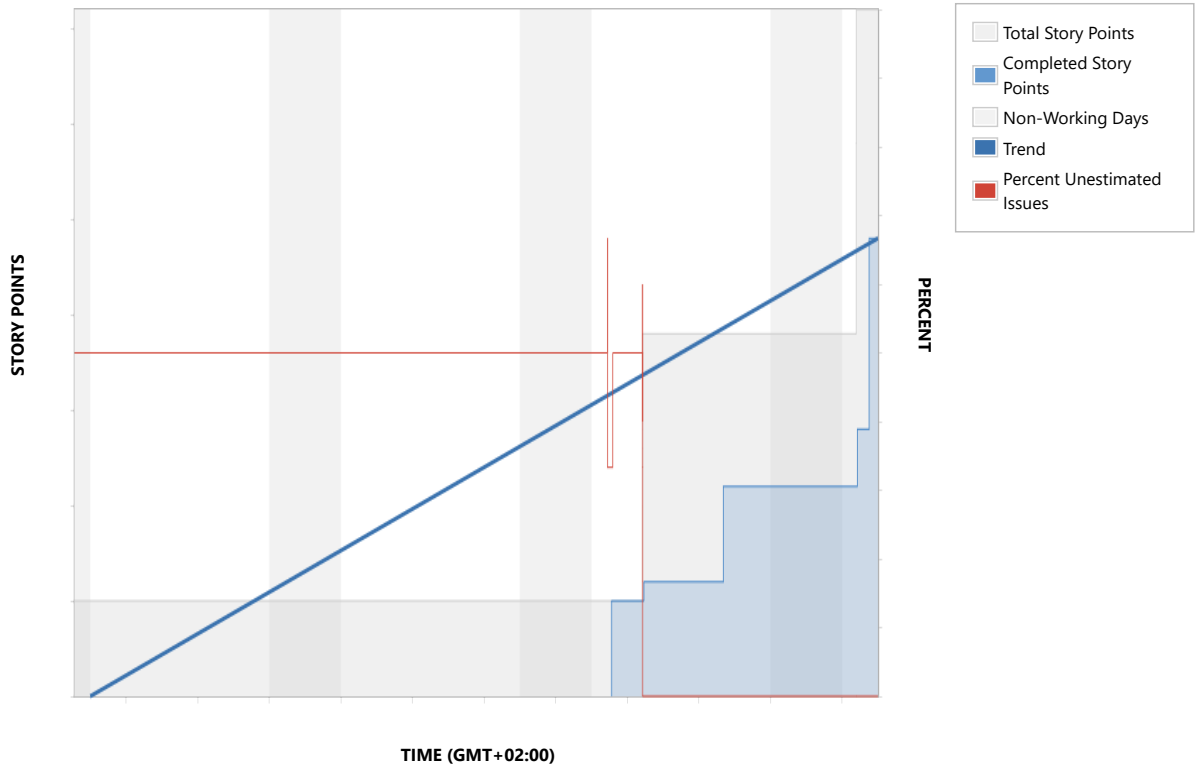
Projects / USN Bachelor group 5 / SP board / Reports

Version Report

USN5: SatStat Beta 1

Start Date: 25/Mar/19
Released on 15/Apr/19

[View SatStat Beta 1 in Issue Navigator](#)



Status Report

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Epic	Story Points (24)
USN5-251	As an embedded developer, I need to implement functionality supporting a subscriber system for sensor data to comply with the SatStat communication protocol.	Story	Medium	DONE	DS Hardware Layer	10
USN5-146	As a test engineer, I would like to not have to adjust COM settings every time I have to connect to the HW layer so that the software is easier to use	Story	Medium	DONE	DS Software Layer	5
USN5-247	Implement error handling and connection timeouts for SWL	Story	Medium	DISCARDED	DS Software Layer	-
USN5-275	As a test engineer, I would like to see the current value of a sensor input so that I can keep an eye on the current status of the system	Story	Medium	DONE	DS Software Layer	1
USN5-279	As a test engineer, I need the ability to adjust accepted values for sensor readings to be able to verify that the system is operating as intended	Story	Medium	DONE	DS Software Layer	5



USN5-281 As a test engineer, I want the ability to save accepted sensor value ranges to a template, so that I can load the same settings for testing at any point in time

Story Medium DONE

DS Software Layer

3



5/15/2019

SP board - Agile Board - USN Bachelor group 5



Projects / USN Bachelor group 5 / SP board / Reports

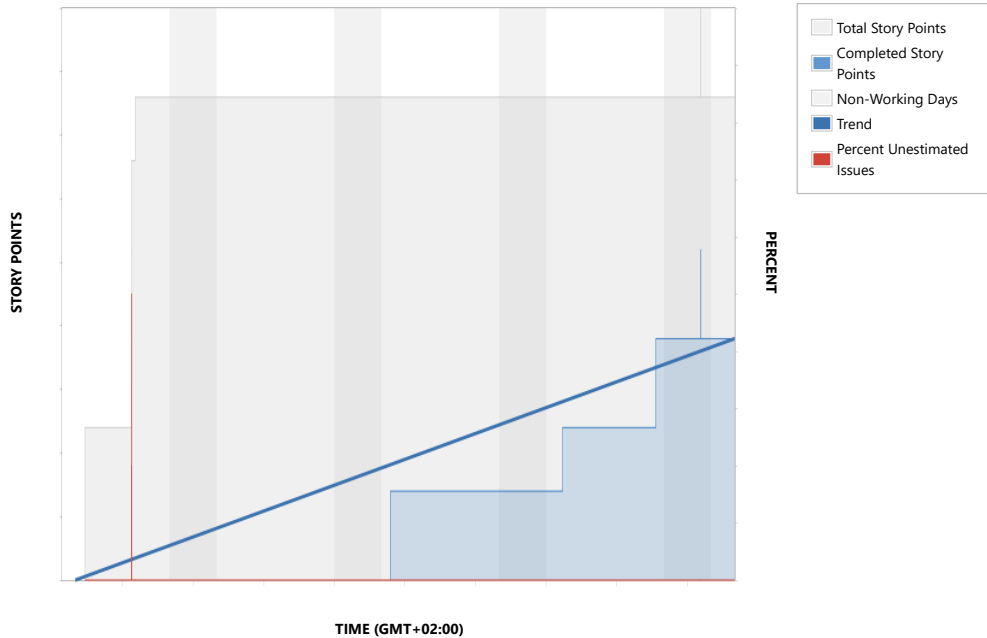
Version Report

...

USN5: SatStat Beta2

Start Date: 16/Apr/19
Released on 13/May/19

[View SatStat Beta2 in Issue Navigator](#)



Predicted completion date: 28/May/19
Optimistic completion date: 27/May/19
Pessimistic completion date: 29/May/19

Status Report

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Epic	Story Points (33)
USN5-250	As an embedded developer, I need to optimize the current version of the HWL to prepare for further development.	Story	Medium	DONE	DS Hardware Layer	7
USN5-311	As a user of the HMI I want a list of available instructions so that I don't have to remember all of them	Story	Medium	DONE	DS Hardware Layer	5
USN5-302	As a test engineer I want the ability to define behaviour and parameters to measure during a test run for the SADM	Story	Medium	DONE	DS Software Layer	7
USN5-304	As a test engineer I want the ability to load test-run settings from a template and initiate a test run based on loaded or defined parameters	Story	Medium	DONE	DS Software Layer	7
USN5-303	As a test engineer I want the ability to save a defined test-run as a template so that I can load the same settings for a future test run	Story	Medium	DONE	DS Software Layer	7

5/13/2019

SP board - Agile Board - USN Bachelor group 5



Projects / USN Bachelor group 5 / SP board / Reports

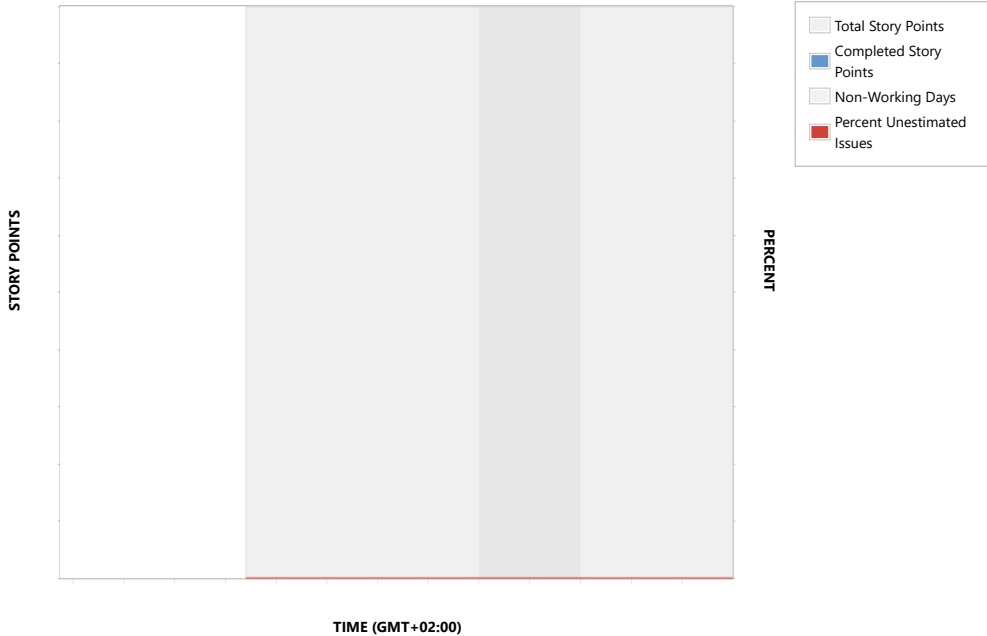
Version Report

...

USN5: Mark1

Start Date: 18/Jan/19
Released on 30/Jan/19

[View Mark1 in Issue Navigator](#)



Predicted completion date: 24/May/19
Optimistic completion date: 23/May/19
Pessimistic completion date: 27/May/19

Status Report

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Epic	Story Points (34)
USN5-45	The group needs a first prototype so that we can show this at the first presentation	Story	Medium	DONE	First presentation	10
USN5-193	As a mechanical engineer, I need to start looking at solutions of design that are suitable for us	Story	Medium	DONE	SADM Assembly	3
USN5-155	The total mass of the SADM, excluding wiring, must be two kilograms or less. This means the drivetrain can only be a certain percentage of this mass and other massive components must be taken into consideration when designing the drivetrain.	Story	Medium	DONE	SADM - Drivetrain	-
USN5-246	The SADMs weight should not exceed 2kg so that the nano satellite will not be affected by the SADM weight	Story	Medium	DONE	SADM Assembly	7
USN5-154	The volume of the SADM should not exceed 100 x 100 x 200 millimeters so that it will fit inside of an 16u satellite	Story	Medium	DONE	SADM Assembly	7
USN5-156	The outer spatial dimensions of The Housing of the SADM must be exactly one hundred by one hundred by two hundred millimeters. The inner spatial dimensions depends on the thickness of the housing walls. The drivetrain and other components inside the SADM...	Story	Medium	DONE	SADM - Drivetrain	7

5/13/2019

SP board - Agile Board - USN Bachelor group 5



5/13/2019

SP board - Agile Board - USN Bachelor group 5



Projects / USN Bachelor group 5 / SP board / Reports

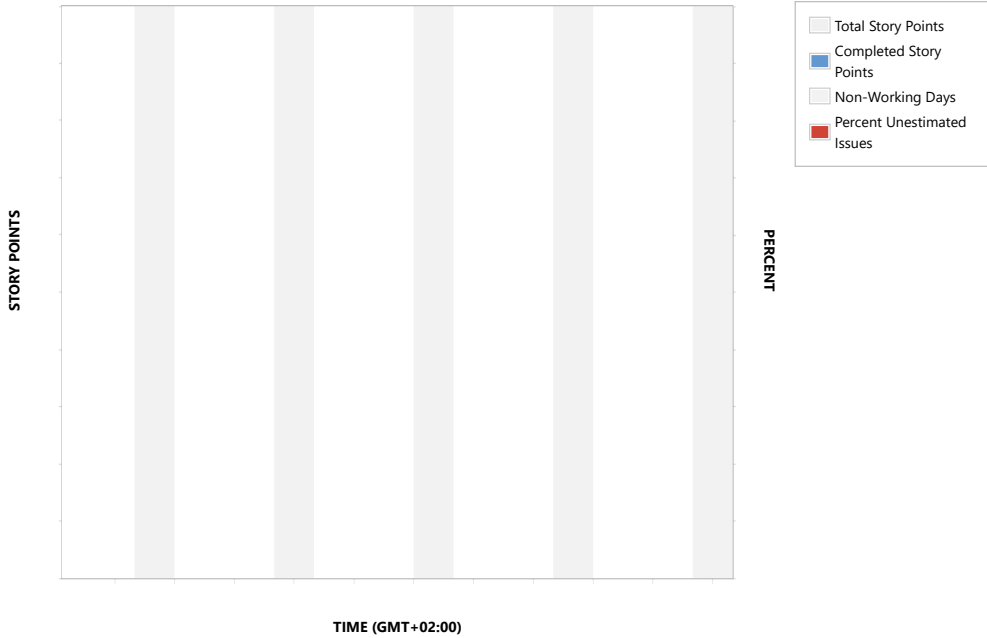
Version Report

...

USN5: Mark2

Start Date: 06/Feb/19
Released on 10/Mar/19

[View Mark2 in Issue Navigator](#)



Predicted completion date: 24/May/19
Optimistic completion date: 23/May/19
Pessimistic completion date: 27/May/19

Status Report

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Epic	Story Points (94)
USN5-123	The SADM physical reference should be illustrated in a sketch so that it's orientation is easy to understand.	Story	Medium	DONE	SADM - Housing	-
USN5-193	As a mechanical engineer, I need to start looking at solutions of design that are suitable for us	Story	Medium	DONE	SADM Assembly	3
USN5-211	As a mechanical engineer, I need to calculate the total moment of inertia that affects our product to make a proper design	Story	Medium	DONE	SADM - Drivetrain	3
USN5-119	As a mechanical engineer I have to design the shaft so that solar arrays can be attached to the spacecraft.	Story	Medium	DISCARDED	SADM - Drivetrain	-
USN5-120	Our product needs to hold the solar array wings in position when not powered	Story	Medium	DISCARDED	SADM - Motor	-
USN5-122	The SADM shall transfer solar array grounding lines to ensure grounding	Story	Medium	DISCARDED	SADM - Power transfer	-
USN5-124	The design of interface between solar array and SADM and motor must be adapted to the weight and	Story	Medium	DISCARDED	SADM - Drivetrain	-

5/13/2019

SP board - Agile Board - USN Bachelor group 5

	size of solar arrays so that it functions as intended.						
USN5-125	Our product needs to perform and function with the given values for a solar array.	Story	↑	Medium	DISCARDED	SADM - Motor	-
USN5-126	The European standard for the space industry must be followed when selecting and designing the placement of the motor. Other requirements from the specified standard must be followed as well.	Story	↑	Medium	DISCARDED	SADM - Motor	-
USN5-129	The mechanism must allow movement when load exceeds 70 Nm	Story	↑	Medium	DISCARDED	SADM - Motor	-
USN5-130	The SADM shall be able to oscillate or rotate 180 degrees around the X-axis in both directions so that it always can point the solar arrays against the sun	Story	↑	Medium	DISCARDED	SADM - Housing	-
USN5-140	The driveline should be single point failure free so that it will be more independent and the whole SADM won't stop working if one component stops working	Story	↑	Medium	DISCARDED	SADM - Drivetrain	-
USN5-147	Our product is to be mounted on a 8u-dummy, with the center of gravity placed at minimum 20 mm from the SADM interface, during vibration testing. The SADM shall have a first Eigen-frequency greater than 250 Hz to fulfill this requirement.	Story	↑	Medium	DISCARDED	SADM - Housing	-
USN5-151	The power output, the angular velocity and velocity resolution from the motor, as well as the gearing ratio, gear stiffness, and shaft stiffness should all be of adequate capacity for the SADM to rotate the shaft connected to the solar arrays with adjust...	Story	↑	Medium	DISCARDED	SADM - Drivetrain	-
USN5-155	The total mass of the SADM, excluding wiring, must be two kilograms or less. This means the drivetrain can only be a certain percentage of this mass and other massive components must be taken into consideration when designing the drivetrain.	Story	↑	Medium	DONE	SADM - Drivetrain	-
USN5-159	As an electrical engineer, I shall make sure the rotor harness have 500mm flying leads so that it will be easy to connect the leads to future connectors.	Story	↑	Medium	DISCARDED	SADM - Power transfer	-
USN5-165	The SADM power consumption shall not exceed a certain level, so that the spacecraft's total power consumption does not exceed the total capacity.	Story	↑	Medium	DISCARDED	SADM - Power transfer	-
USN5-168	As mechanical and electrical engineer, I shuld design the SADM to operate under given values for pressure to ensure functionality during it's entire life cycle.	Story	↑	Medium	DISCARDED	SADM Assembly	-
USN5-188	As a mechanical engineer, I should ensure bimetallic compability to avoid corrosion and other inconveniences.	Story	↑	Medium	DISCARDED	SADM Assembly	-
USN5-189	As mechanical and electrical engineers, we shall take radiation	Story	↑	Medium	DISCARDED	SADM Assembly	10

<https://usnbachelor.atlassian.net/secure/RapidBoard.jspa?rapidView=2&projectKey=USN5&view=reporting&chart=versionReport&version=10003>

2/4











5/13/2019

SP board - Agile Board - USN Bachelor group 5

	into consideration when selecting materials for the SADM to be able to withstand the radiation levels for orbit and lifetime.					
USN5-194	As electrical and mechanical engineers, we have to design the SADM to meet requirements for reliability to ensure functionality through the SADM's entire lifetime.	Story	Medium	DISCARDED	SADM Assembly	-
USN5-195	As an electrical engineer, I should derate the electrical components so that they will function optimally and not take any damage due to overload, heat and other.	Story	Medium	DISCARDED	SADM - Power transfer	-
USN5-196	As an electrical engineer, I should make sure the power transfer meets acceptable derating performance, so that the power transfer can still be operative in failure cause.	Story	Medium	DISCARDED	SADM - Power transfer	-
USN5-198	As an electrical engineer, I must be familiar with flex-suppliers and characteristics needed for the flex so that we can decide what specifications are needed.	Story	High	DONE	SADM - Power transfer	5
USN5-199	As an electrical engineer, I should design first prototype for the cabinet so that we can review this for customer.	Story	Medium	DONE	DS Hardware Layer	5
USN5-209	The SADM power transfer system should be derated so that if a failure happens it won't affect the other components in the assembly	Story	Medium	DISCARDED	SADM - Power transfer	-
USN5-210	As an electrical engineer, I need to design the cables, flex and equipment to have adequate conductivity.	Story	Medium	DISCARDED	SADM - Power transfer	-
USN5-212	As a mech. engineer, I need to talk with our external supervisor to clarify some requirements for design	Story	Medium	DONE	SADM Assembly	5
USN5-213	As a mech. engineer, I need to cooperate with an electrical engineer to consider different designs to decide a suitable i/f between the flex and the shaft	Story	Medium	DONE	SADM - Drivetrain	7
USN5-161	The motor should have specific nominal electrical interfaces so that it will satisfy the SADM power needs	Story	High	DISCARDED	SADM - Motor	5
USN5-214	As an electrical engineer, I must design a prototype PCB rigid-flex	Story	Medium	DISCARDED		3
USN5-204	The SADM should include all the necessary components to function	Story	High	DONE	SADM Assembly	7
USN5-268	As an electrical engineer, I must design a flexible PCB	Story	Medium	DONE		7
USN5-246	The SADM's weight should not exceed 2kg so that the nano satellite will not be affected by the SADM weight	Story	Medium	DONE	SADM Assembly	7
USN5-154	The volume of the SADM should not exceed 100 x 100 x 200 millimeters so that it will fit inside of an 16u satellite	Story	Medium	DONE	SADM Assembly	7
USN5-158	The motor of the SADM should be marked with a zero reference point so that it will be easier to mate the assembly	Story	Medium	DONE	SADM - Motor	3

5/13/2019

SP board - Agile Board - USN Bachelor group 5

   	<p>USN5-156 The outer spatial dimensions of The Housing of the SADM must be exactly one hundred by one hundred by two hundred millimeters. The inner spatial dimensions depends on the thickness of the housing walls. The drivetrain and other components inside the SADM...</p>	 Story  Medium DONE SADM - Drivetrain	7
	<p>USN5-221 As a mechanical engineer, I need to start doing research on materials so I can be familiar with the possibility of use for coming prototypes</p>	 Story  Medium DONE SADM Assembly	3
	<p>USN5-220 As a mechanical engineer I need to consider materials to use on the shaft so that it will be compatible in space environment</p>	 Story  Medium DONE SADM - Drivetrain	7

 5




5/20/2019

SP board - Agile Board - USN Bachelor group 5



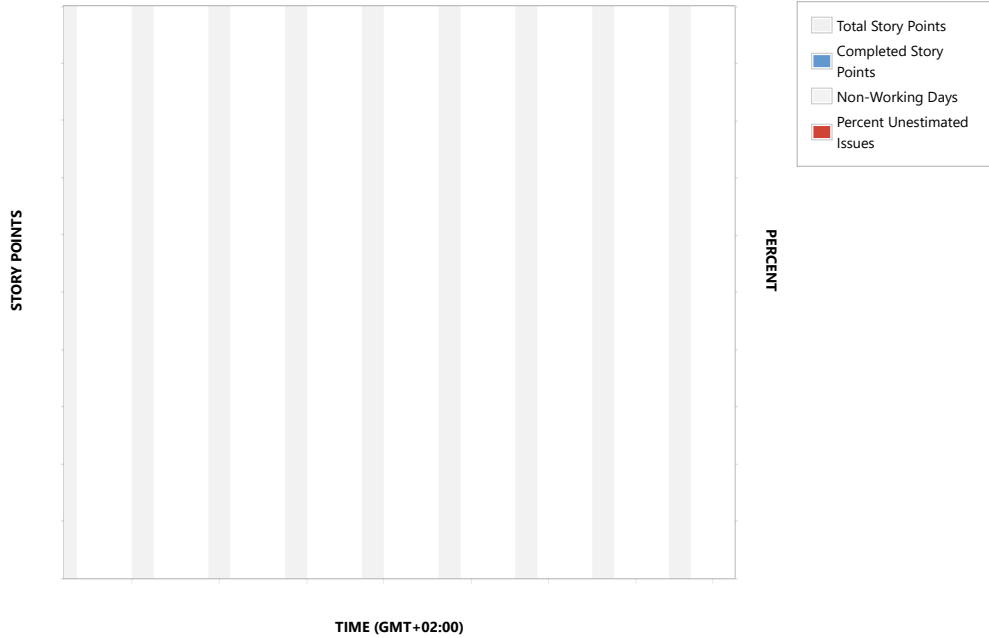
Projects / USN Bachelor group 5 / SP board / Reports

Version Report

USN5: Mark3

Start Date: 11/Mar/19
Released on 09/May/19

[View Mark3 in Issue Navigator](#)



Status Report



Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Epic	Story Points (251)
USN5-123	The SADM physical reference should be illustrated in a sketch so that it's orientation is easy to understand.	Story	Medium	DONE	SADM - Housing	-
USN5-185	As a mechanical engineer, I must take into account impacts and effects of environmental radiation etc. in a reasonable way to prepare the SADM prototype for the environment in space.	Story	Medium	DONE	SADM Assembly	-
USN5-289	As an electrical engineer, I must draw version 2 of the electrical drawings so that production of the cabinet will be correct.	Story	Medium	DONE	Final presentation	5
USN5-119	As a mechanical engineer I have to design the shaft so that solar arrays can be attached to the spacecraft.	Story	Medium	DISCARDED	SADM - Drivetrain	-
USN5-120	Our product needs to hold the solar array wings in position when not powered	Story	Medium	DISCARDED	SADM - Motor	-
USN5-122	The SADM shall transfer solar array grounding lines to ensure grounding	Story	Medium	DISCARDED	SADM - Power transfer	-
USN5-124	The design of interface between solar array and SADM and motor must be adapted to the weight and	Story	Medium	DISCARDED	SADM - Drivetrain	-

5/20/2019

SP board - Agile Board - USN Bachelor group 5

ID	Description	Icon	Type	Priority	Status	Component	Assignee
	size of solar arrays so that it functions as intended.						
USN5-125	Our product needs to perform and function with the given values for a solar array.	🟢	Story	↑ Medium	DISCARDED	SADM - Motor	-
USN5-126	The European standard for the space industry must be followed when selecting and designing the placement of the motor. Other requirements from the specified standard must be followed as well.	🟢	Story	↑ Medium	DISCARDED	SADM - Motor	-
USN5-127	The motor must be able to hold the solar array in the same position with a load of 2Nm or greater, when it is not unpowered.	🟢	Story	↑ Medium	DISCARDED	SADM - Motor	-
USN5-128	The motor must be able to hold the solar array in the same position with a load of 8 Nm, when the motor is running with a current of 150 mA.	🟢	Story	↑ Medium	DISCARDED	SADM - Motor	-
USN5-129	The mechanism must allow movement when load exceeds 70 Nm	🟢	Story	↑ Medium	DISCARDED	SADM - Motor	-
USN5-130	The SADM shall be able to oscillate or rotate 180 degrees around the X- axis in both directions so that it always can point the solar arrays against the sun	🟢	Story	↑ Medium	DISCARDED	SADM - Housing	-
USN5-134	The drivetrain shall keep its position under spacecraft rotational accelerations so that the solar arrays holds it's positions.	🟢	Story	↑ Medium	DISCARDED	SADM - Drivetrain	-
USN5-135	The drive train should have a life time of 5 years in an orbit of 600km so that it will satisfy the customers	🟢	Story	↑ Medium	DISCARDED	SADM - Drivetrain	-
USN5-136	The motor shall keep its position under spacecraft rotational accelerations so that the solar arrays holds it's positions.	🟢	Story	↑ Medium	DISCARDED	SADM - Motor	-
USN5-137	The motor should have a life time of 5 years in an orbit of 660km so that it will satisfy the customers	🟢	Story	↑ Medium	DISCARDED	SADM - Motor	-
USN5-138	The housing should have a life time of 5 years in an orbit of 660km so that it will satisfy the customers	🟢	Story	↑ Medium	DISCARDED	SADM - Housing	-
USN5-139	The power transfer system should have a life time of 5 years in an orbit of 660km so that it will satisfy the customers	🟢	Story	↑ Medium	DISCARDED	SADM - Power transfer	-
USN5-140	The driveline should be single point failure free so that it will be more independent and the whole SADM won't stop working if one component stops working	🟢	Story	↑ Medium	DISCARDED	SADM - Drivetrain	-
USN5-141	The SADM shall have enough torque margins to withstand a short circuit of motor winding without loss of functionality so that operation continues uninterrupted.	🟢	Story	↑ Medium	DISCARDED	SADM - Motor	-
USN5-142	The SADM shall provide an axial translation stiffness of >- 1.0 x 10^8 N/m	🟢	Story	↑ Medium	DISCARDED	SADM - Housing	-
USN5-143	The SADM shall provide a radial translation stiffness of >- 1.0 x 10^8 N/m with the solar array	🟢	Story	↑ Medium	DISCARDED	SADM - Housing	-

<https://usnbachelor.atlassian.net/secure/RapidBoard.jspa?rapidView=2&projectKey=USN5&view=reporting&chart=versionReport&version=10008>

2/7

5/20/2019

SP board - Agile Board - USN Bachelor group 5

		interface constrained in cross axis bending (interface plane remains parallel under loading)					
	USN5-144	The SADM shall provide a cross axis bending stiffness of $> - 1.0 \times 10^5$ Nm/rad	Story	Medium	DISCARDED	SADM - Housing	-
	USN5-145	The SADM shall provide a torsional stiffness of $> - 10^4$ Nm/rad (assuming a blocked motor shaft, at 5 to 15 Nm external load on the output shaft)	Story	Medium	DISCARDED	SADM - Housing	-
	USN5-147	Our product is to be mounted on a 8u-dummy, with the center of gravity placed at minimum 20 mm from the SADM interface, during vibration testing. The SADM shall have a first Eigen-frequency greater than 250 Hz to fulfill this requirement.	Story	Medium	DISCARDED	SADM - Housing	-
	USN5-148	As an electrical engineer, I need to make sure the required derating performances of the power transfers will be adequate	Story	Medium	DISCARDED	SADM - Power transfer	-
	USN5-149	The housing needs to include venting paths so that there will be low pressure in the SADM	Story	Medium	DISCARDED	SADM - Housing	-
	USN5-150	The materials of the housing should meet the requirement of Total Ionising Dose (TID) level of 10 Mrad (Milliradians)	Story	Medium	DISCARDED	SADM - Housing	-
	USN5-151	The power output, the angular velocity and velocity resolution from the motor, as well as the gearing ratio, gear stiffness, and shaft stiffness should all be of adequate capacity for the SADM to rotate the shaft connected to the solar arrays with adjust...	Story	Medium	DISCARDED	SADM - Drivetrain	-
	USN5-155	The total mass of the SADM, excluding wiring, must be two kilograms or less. This means the drivetrain can only be a certain percentage of this mass and other massive components must be taken into consideration when designing the drivetrain.	Story	Medium	DONE	SADM - Drivetrain	-
	USN5-157	The SADM should be able to connect accordingly to the satellite and the solar arrays	Story	Medium	DISCARDED	SADM Assembly	-
	USN5-172	As a mechanical and electrical engineer, I need to design the SADM to withstand vibrations to ensure functionality through its lifecycle.	Story	Medium	DISCARDED	SADM Assembly	-
	USN5-159	As an electrical engineer, I shall make sure the rotor harness have 500mm flying leads so that it will be easy to connect the leads to future connectors.	Story	Medium	DISCARDED	SADM - Power transfer	-
	USN5-160	As an electrical engineer, I need to make sure the grounding lines from the solar arrays are separated from the SADM grounding.	Story	Medium	DISCARDED	SADM - Power transfer	-
	USN5-162	The motors interface should be designed/chosen by standard sizes(NEMA) so that it will be easier to connect to other standard products	Story	Medium	DISCARDED	SADM - Motor	-

<https://usnbachelor.atlassian.net/secure/RapidBoard.jspa?rapidView=2&projectKey=USN5&view=reporting&chart=versionReport&version=10008>

3/7

5/20/2019

SP board - Agile Board - USN Bachelor group 5



USN5-163	As an electrical engineer, I need to make sure there are connectors on the stator side to be able to transfer power and signals from/to the SADM and rest of the spacecraft.	Story	Medium	DISCARDED	SADM - Power transfer	-
USN5-164	As an electrical engineer, I have to ensure sufficient grounding to maintain optimal operation of the SADM/spacecraft	Story	Medium	DISCARDED	SADM - Power transfer	-
USN5-165	The SADM power consumption shall not exceed a certain level, so that the spacecraft's total power consumption does not exceed the total capacity.	Story	Medium	DISCARDED	SADM - Power transfer	-
USN5-166	The SADM and related equipment shall be designed for operation at given temperature range, so that the environmental temperature does not harm the functionality.	Story	Medium	DISCARDED	SADM Assembly	-
USN5-167	As an electrical and mechanical engineer, I have to design the SADM so that functionality and operation will not be harmed due to humidity during it's life cycle.	Story	Medium	DISCARDED	SADM Assembly	-
USN5-168	As mechanical and electrical engineer, I shuld design the SADM to operate under given values for pressure to ensure functionality during it's entire life cycle.	Story	Medium	DISCARDED	SADM Assembly	-
USN5-171	As a mechanical engineer, I need to design the interface between SADM and solar arrays so that it can withstand given values for forces.	Story	Medium	DISCARDED	SADM - Drivetrain	-
USN5-173	As a mechanical engineer, I need to design the SADM to withstand random vibrations so that the SADM is suited for environmental impacts.	Story	Medium	DISCARDED	SADM Assembly	-
USN5-174	As a mechanical engineer, I need to consider how the SADM should be able to withstand shock.	Story	Medium	DISCARDED	SADM Assembly	-
USN5-175	As a mechanical and electrical engineer, I should design the SADM connectors and interfaces to be operative in given temperature range.	Story	Medium	DISCARDED	SADM Assembly	-
USN5-176	As a mechanical engineer, I should design the SADM mechanically considering the heat flow from the SADM, so that the amount of heat flow will not have a negtive impact to the rest of the spacecraft and it's functionality.	Story	Medium	DISCARDED	SADM Assembly	-
USN5-177	As a mechanical and electrical engineer, I should design the SADM mechanically considering the heat flow from the SADM to the solar arrays, so that the amount of heat flow will not have a negtive impact to the solar arrays.	Story	Medium	DISCARDED	SADM Assembly	-
USN5-178	As a mechanical engineer, I should decide if we are going to do a thermal analysis and define the necessary equipment to this analysis.	Story	Medium	DISCARDED	SADM Assembly	-
USN5-179	As a mechanical engineer, I should design the SADM to be compatible	Story	Medium	DISCARDED		-



5/20/2019

SP board - Agile Board - USN Bachelor group 5

		with thermal conductance between SADM and spacecraft.					
USN5-180	As a electrical and mechanical engineer, I need to ensure contact between each part so that all mechanical parts of the SADM has he same electrical potential.	Story	↑	Medium	DISCARDED	SADM Assembly	-
USN5-181	As a electrical engineer, I should make sure the potential between metallic parts and intrinsically conductive parts that do not perform any electrical funtion is grounded according to specifications.	Story	↑	Medium	DISCARDED	SADM - Power transfer	-
USN5-184	As a mechanical engineer, I must design the required shielding for the SADM (EEE)	Story	↑	Medium	DISCARDED	SADM Assembly	-
USN5-187	As a mechanical engineer, I have to be familiar with standards for spacegrade compability and decide how and if the SADM will ensure vacuum compability.	Story	↑	Medium	DISCARDED	SADM Assembly	-
USN5-188	As a mechanical engineer, I should ensure bimetallic compability to avoid corrosion and other inconveniences.	Story	↑	Medium	DISCARDED	SADM Assembly	-
USN5-189	As mechanical and electrical engineers, we shall take radiation into consideration when selecting materials for the SADM to be able to withstand the radiation levels for orbit and lifetime.	Story	↑	Medium	DISCARDED	SADM Assembly	10
USN5-194	As electrical and mechanical engineers, we have to design the SADM to meet requirements for reliability to ensure functionality trough the SADMs entire lifetime.	Story	↑	Medium	DISCARDED	SADM Assembly	-
USN5-195	As an electrical engineer, I should derate the electronical components so that the will function optimal and not take any damage due to overload, heat and other.	Story	↑	Medium	DISCARDED	SADM - Power transfer	-
USN5-196	As an electrical engineer, I should make sure the power transfers meets acceptable derating performance, so that the power transfer can still be operative in failure cause.	Story	↑	Medium	DISCARDED	SADM - Power transfer	-
USN5-198	As an electrical engineer, I must be familiar with flex-suppliers and characteristics needed for the flex so that we can decide what spesifications are needed.	Story	↑	High	DONE	SADM - Power transfer	5
USN5-199	As an electrical engineer, I should design first prototype for the cabinet so that we can review this for costumer.	Story	↑	Medium	DONE	DS Hardware Layer	5
USN5-208	The SADM should satisfy the requirements for frequency range on ground and max disturbance moment on ground so that it won't be ruined	Story	↑	Medium	DISCARDED		-
USN5-209	The SADM power transfer system should be derated so that if a failure happens it won't affect the other components in the assembly	Story	↑	Medium	DISCARDED	SADM - Power transfer	-
USN5-210	As an electrical engineer, I need to	Story	↑	Medium	DISCARDED	SADM - Power transfer	-

<https://usnbachelor.atlassian.net/secure/RapidBoard.jspa?rapidView=2&projectKey=USN5&view=reporting&chart=versionReport&version=10008>

5/7

5/20/2019

SP board - Agile Board - USN Bachelor group 5

ID	Description	Type	Priority	Status	Category	Points
	design the cables, flex and equipment to have adequate conductivity.					
USN5-212	As a mech. engineer, I need to talk with our external supervisor to clarify some requirements for design	Story	Medium	DONE	SADM Assembly	5
USN5-213	As a mech. engineer, I need to cooperate with an electrical engineer to consider different designs to decide a suitable i/f between the flex and the shaft	Story	Medium	DONE	SADM - Drivetrain	7
USN5-161	The motor should have specific nominal electrical interfaces so that it will satisfy the SADM power needs	Story	High	DISCARDED	SADM - Motor	5
USN5-214	As an electrical engineer, I must design a prototype PCB rigid-flex	Story	Medium	DISCARDED		3
USN5-204	The SADM should include all the necessary components to function	Story	High	DONE	SADM Assembly	7
USN5-222	As designers, we need to decide the final design of our product so we can start choosing materials and setting right dimensions	Story	High	DONE	SADM Assembly	10
USN5-270	As an electrical engineer, I must decide which supplier and kind of sensor to measure the position of the shaft so that the DS can get feedback.	Story	High	DISCARDED		5
USN5-268	As an electrical engineer, I must design a flexible PCB	Story	Medium	DONE		7
USN5-276	Mech: Make a structure for Mark3	Story	Medium	DONE		7
USN5-277	Mech: Make a lid for Mark3 to attach the motor	Story	Medium	DONE		7
USN5-284	Attend meeting with mechanical lab at KONGSBERG to acquire information about the Mark3 parts	Story	Medium	DONE		1
USN5-286	Mech: Redesign flex shaft design	Story	Medium	DONE		10
USN5-287	Mech: Make fittings for D-sub in the Mark3 housing	Story	Medium	DONE		3
USN5-290	Complete order of electrical equipment.	Story	Medium	DONE	DS Hardware Layer	5
USN5-296	Make Mark3 ready for 3D printing	Story	Medium	DONE		7
USN5-297	Start 3D printing of Mark3 parts	Story	High	DONE		10
USN5-298	Design connection interface between flex and wires.	Story	Medium	DONE	SADM - Power transfer	10
USN5-305	Redesign the motor holder of Mark3	Story	Medium	DONE		5
USN5-312	Mech: Create gears for Mark3	Story	Medium	DONE		5
USN5-285	Mech: Make 2D drawings of the Mark3 parts	Story	Medium	DONE		7
USN5-306	Make exits for the flex out of the capsule housing for Mark3	Story	Medium	DONE		5
USN5-309	Make holes for screws on the structure of Mark3	Story	Medium	DONE		5
USN5-323	Write about choice of materials in the report	Story	Medium	DONE		5
USN5-295	Elec: Build cabinet	Story	High	DONE		7

<https://usnbachelor.atlassian.net/secure/RapidBoard.jspa?rapidView=2&projectKey=USN5&view=reporting&chart=versionReport&version=10008>

6/7

5/20/2019

SP board - Agile Board - USN Bachelor group 5



USN5-325	Make a manual for how to put Mark3 together, in SolidWorks	Story	↑ Medium	DONE		7
USN5-318	Mech: Make an adapter to the stepper motor for Mark3	Story	↑ Medium	DONE		7
USN5-307	Make a mechanical stop for the main shaft	Story	↑ Medium	DONE		5
USN5-308	Redesign middle wall on Mark3 model for the mechanical stop	Story	↑ Medium	DONE		3
USN5-246	The SADMs weight should not exceed 2kg so that the nano satellite will not be affected by the SADM weight	Story	↑ Medium	DONE	SADM Assembly	7
USN5-153	The SADMs weight should not exceed 2kg so that the nano satellite will not be affected by the SADM weight - For Mark2	Story	↑ Medium	DONE	SADM Assembly	7
USN5-154	The volume of the SADM should not exceed 100 x 100 x 200 millimeters so that it will fit inside of an 16u satellite	Story	↑ Medium	DONE	SADM Assembly	7
USN5-158	The motor of the SADM should be marked with a zero reference point so that it will be easier to mate the assembly	Story	↑ Medium	DONE	SADM - Motor	3
USN5-156	The outer spatial dimensions of The Housing of the SADM must be exactly one hundred by one hundred by two hundred millimeters. The inner spatial dimensions depends on the thickness of the housing walls. The drivetrain and other components inside the SADM...	Story	↑ Medium	DONE	SADM - Drivetrain	7
USN5-221	As a mechanical engineer, I need to start doing research on materials so I can be familiar with the possibility of use for coming prototypes	Story	↑ Medium	DONE	SADM Assembly	3
USN5-220	As a mechanical engineer I need to consider materials to use on the shaft so that it will be compatible in space environment	Story	↑ Medium	DONE	SADM - Drivetrain	7
USN5-271	As an electrical engineer, I must make the first physical prototype of the connection between flex and wires.	Story	↑ High	DONE	SADM - Power transfer	5
USN5-288	Mech: Redesign middle wall in the Mark3 structure	Story	↑ Medium	DISCARDED		-
USN5-310	Make a "jumping hill" for the flex shaft	Story	↑ Medium	DISCARDED		-
USN5-319	As an electrical engineer I must design an electrical analogue to the flex	Story	↑ Medium	DONE		-
USN5-324	Finish simulations in SolidWorks for both Three and Two Bearings	Story	↑ Medium	DISCARDED		10

Bachelor of Engineering

Project appendix

Winter semester 2019

Jira Export - Sprint Reports

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	Thomas Mundal and Jon Skjelsbæk
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains every Sprint Report exported from Jira. These reports provides a brief overview of User Stories concerning the given Sprint.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

Contents

- A Sprint Reports** **2**
- A.1 Sprint 1 2
- A.2 Sprint 2 4
- A.3 Sprint 3 5
- A.4 Sprint 4 6
- A.5 Sprint 5 7
- A.6 Sprint 6 8
- A.7 Sprint 7 10
- A.8 Sprint 8 12
- A.9 Sprint 9 14
- A.10 Sprint 10 15
- A.11 Sprint 11 16
- A.12 Sprint 12 18
- A.13 Sprint 13 20
- A.14 Sprint 14 21
- A.15 Sprint 15 22
- A.16 Sprint 16 23

Appendix A

Sprint Reports

5/13/2019

SP board - Agile Board - USN Bachelor group 5



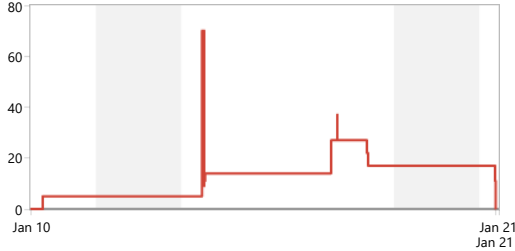
Projects / USN Bachelor group 5 / SP board / Reports

Sprint Report

SP Sprint 1

Closed sprint, ended by Thomas Mundal 10/Jan/19 11:02 AM - 21/Jan/19 9:18 AM 1 linked page

Ha en ferdig prosjektmodell



Status Report

* Issue added to sprint after start time

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (8 → 24)
USN5-10	The group needs a project model in order to complete the project	Story	Highest	DONE	- → 5
USN5-22 *	As a product owner i need to be familiar with my tasks so that I can satisfy the role description	Story	Highest	DONE	- → 3
USN5-24 *	The group needs a project plan to keep track of time and tasks	Story	High	DONE	5
USN5-29 *	Finish the report template	Story	Medium	DONE	- → 5
USN5-35 *	Go through and make necessary changes to the Drive	Story	Medium	DONE	- → 3
USN5-38 *	The group needs a group contract so that everyone agrees on rules of conduct	Story	Medium	DONE	3

Issues Removed From Sprint

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (30 → 13)
USN5-27 *	As requirement supervisor I need to be familiar with my responsibilities so that I can satisfy the role of description	Story	High	PRODUCTION	20 → 3
USN5-48 *	The group needs to deliver documents two days before the presentation because it's required from the sensors	Story	High	DESIGN	10
USN5-49 *	As the one responsible for the presentation tool, the assigned needs to be familiar with Prezi or other presentation tools.	Story	Low	DESIGN	-

5/13/2019

SP board - Agile Board - USN Bachelor group 5

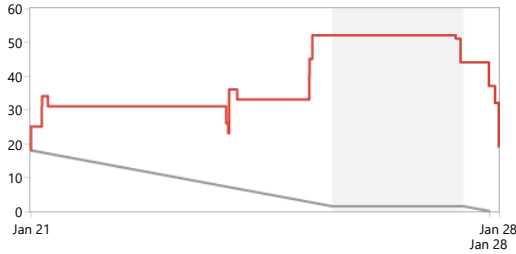


Projects / USN Bachelor group 5 / SP board / Reports

Sprint Report

SP Sprint 2

Closed sprint, ended by Rita Hogstad 21/Jan/19 9:39 AM - 28/Jan/19 12:54 PM [Linked pages](#)



Status Report

* Issue added to sprint after start time

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (27 → 47)
USN5-43	The process model needs to incorporate process retrospectives so that the group can uncover challenges related to the project generally	Story	Medium	DONE	1
USN5-51	The group needs to form and agree upon a description of the task in a document	Story	Medium	DONE	5
USN5-54	Create a chart with elements and terminology from KDA's V-model to be able to implement the stakeholders routines for prototype developemet.	Story	Medium	DONE	3
USN5-55	The group needs a name	Story	Medium	DONE	3
USN5-58 *	Form agenda for meeting with external supervisor	Story	Medium	DONE	- → 3
USN5-59 *	Form agenda for meeting with internal supervisor	Story	Medium	DONE	- → 3
USN5-60 *	As an embedded systems developer I want make fundamental decisions to get a better understanding of the constraints of the hardware related development	Story	Medium	DONE	- → 7
USN5-66 *	Make a Unified Process chart, to compare Gantt	Story	Medium	DONE	3
USN5-69 *	Define requirements for diagnostics system	Story	Medium	DONE	7
USN5-70 *	As a group we need an easy way to present our project plan for the first presentation	Story	Medium	DONE	5
USN5-72 *	As the lead test engineer, I need to create a template and procedure for how to register the verification of the requirements	Story	High	DONE	- → 7

Issues Not Completed

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (19)
USN5-27	As requirement supervisor I need to be familiar with my responsibilities so that I can satisfy the role of description	Story	High	PRODUCTION	3
USN5-45 *	The group needs a first prototype so that we can show this at the first presentation	Story	Medium	DESIGN	10
USN5-56	We need a script draft for the first presentation	Story	Highest	DESIGN	3
USN5-64 *	Do stakeholder analysis	Story	Medium	TESTING	3

5/13/2019

SP board - Agile Board - USN Bachelor group 5

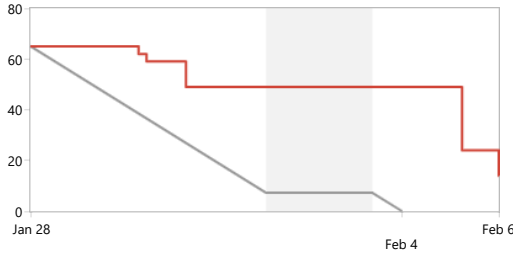


Projects / USN Bachelor group 5 / SP board / Reports

Sprint Report

SP Sprint 3

Closed sprint, ended by Rita Hogstad 28/Jan/19 1:22 PM - 06/Feb/19 9:09 AM [Linked pages](#)



Status Report

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (51)
USN5-45	The group needs a first prototype so that we can show this at the first presentation	Story	Medium	DONE	10
USN5-48	The group needs to deliver documents two days before the presentation because it's required from the sensors	Story	High	DONE	10
USN5-56	We need a script draft for the first presentation	Story	Highest	DONE	5
USN5-57	We need a finalized script for the first presentation	Story	High	DONE	5
USN5-64	Do stakeholder analysis	Story	Medium	DONE	5
USN5-65	Perform a SWOT-analysis	Story	Medium	DONE	3
USN5-71	Revise the task description	Story	Medium	DONE	5
USN5-79	Make a logo	Story	Medium	DONE	3
USN5-80	Finish the slides for the presentation	Story	High	DONE	5

Issues Not Completed

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (14)
USN5-67	Requirement analysis	Story	Highest	OPEN	7
USN5-68	Define verification methods for requirements	Story	Highest	OPEN	7

Issues completed outside of this sprint

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (3)
USN5-27	As requirement supervisor I need to be familiar with my responsibilities so that I can satisfy the role of description	Story	High	DONE	3

5/13/2019

SP board - Agile Board - USN Bachelor group 5



Projects / USN Bachelor group 5 / SP board / Reports

Sprint Report

SP Sprint 4

Closed sprint, ended by Thomas Mundal 06/Feb/19 9:24 AM - 11/Feb/19 9:56 AM [Linked pages](#)



Status Report

* Issue added to sprint after start time

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (14)
USN5-67	Requirement analysis	Story	Highest	DONE	7
USN5-68	Define verification methods for requirements	Story	Highest	DISCARDED	7
USN5-123 *	The SADM physical reference should be illustrated in a sketch so that it's orientation is easy to understand.	Story	Medium	DONE	-



5/13/2019

SP board - Agile Board - USN Bachelor group 5

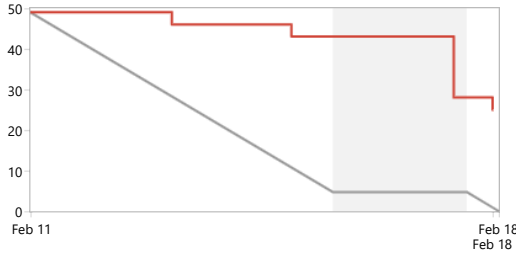


Projects / USN Bachelor group 5 / SP board / Reports

Sprint Report

SP Sprint 5

Closed sprint, ended by Thomas Mundal 11/Feb/19 11:32 AM - 18/Feb/19 9:30 AM [Linked pages](#)



Status Report

* Issue added to sprint after start time

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (31)
USN5-68	Define verification methods for requirements	Story	Highest	DONE	7
USN5-98	As an embedded developer, I need to make sure that instructions can be received even though another instruction is currently executing.	Story	Medium	DONE	5
USN5-99	As an embedded developer, I need to adapt the instructions to the proper format for the receiving device to understand.	Story	Medium	DONE	5
USN5-101	As an embedded developer, I need the receiving end of my hardware to speak the same language as the software layer in order to properly interpret the received instructions.	Story	Medium	DONE	5
USN5-108	As a software engineer, I need the diagnostic system to use a standardized way of exchanging data	Story	High	DONE	3
USN5-193	As a mechanical engineer, I need to start looking at solutions of design that are suitable for us	Story	Medium	DONE	3
USN5-197	As a mechanical engineer, I need to describe how we made the first prototype, so that we can use it in the report.	Story	Medium	DONE	3
USN5-207 *	As mechanical engineers, we should make a disciplinary project plan, so that everyone in the team knows what we have planned to do	Story	Medium	DONE	-

Issues Not Completed

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (25)
USN5-74	As a test engineer, I need a software that can give me diagnostic data in real time	Story	High	TESTING	5
USN5-186	As team members, verifications for every requirements that are reviewed needs to be set in the VMD	Story	Medium	PRODUCTION	3
USN5-198	As an electrical engineer, I must be familiar with flex-suppliers and characteristics needed for the flex so that we can decide what specifications are needed.	Story	High	OPEN	5
USN5-199	As an electrical engineer, I should design first prototype for the cabinet so that we can review this for costumer.	Story	Medium	OPEN	5
USN5-200	As requirement responsible, I must ensure structure and order in the requirement pages in Confluence, so that all requirements and user stories are traceable and structured.	Story	Medium	OPEN	7

5/13/2019

SP board - Agile Board - USN Bachelor group 5

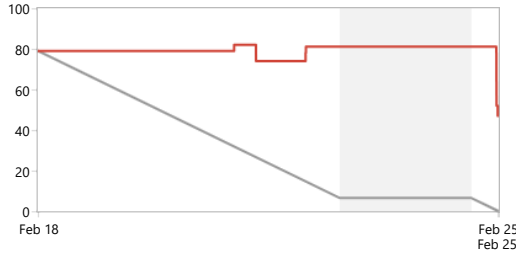


Projects / USN Bachelor group 5 / SP board / Reports

Sprint Report

SP Sprint 6

Closed sprint, ended by Rita Hogstad 18/Feb/19 9:49 AM - 25/Feb/19 9:57 AM [Linked pages](#)



Status Report

* Issue added to sprint after start time

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (30 → 37)
USN5-199	As an electrical engineer, I should design first prototype for the cabinet so that we can review this for costumer.	Story	Medium	DONE	5
USN5-206	Make an electrical process plan	Story	High	DONE	7
USN5-211	As a mechanical engineer, I need to calculate the total moment of inertia that affects our product to make a proper design	Story	Medium	DONE	3
USN5-212	As a mech. engineer, I need to talk with our external supervisor to clarify some requirements for design	Story	Medium	DONE	5
USN5-213	As a mech. engineer, I need to cooperate with an electrical engineer to consider different designs to decide a suitable i/f between the flex and the shaft	Story	Medium	DONE	7
USN5-220 *	As a mechanical engineer I need to consider materials to use on the shaft so that it will be compatible in space environment	Story	Medium	DONE	- → 7
USN5-221 *	As a mechanical engineer, I need to start doing research on materials so I can be familiar with the possibility of use for coming prototypes	Story	Medium	DONE	3

Issues Not Completed

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (47)
USN5-74	As a test engineer, I need a software that can give me diagnostic data in real time	Story	High	TESTING	5
USN5-76	As a test engineer, I need a simple way of configuring the communication between the diagnostics system software and hardware	Story	Medium	OPEN	5
USN5-77	As a software engineer, I need the diagnostic system to be easily expandable to allow for other diagnostics components	Story	High	PRODUCTION	10
USN5-96	As an embedded developer, I need my hardware to be able to control the SADM to be able to measure the result.	Story	Medium	OPEN	3
USN5-100	As an embedded developer, I need to make sure that the correct instructions are executed at the correct device for the test results to be correct.	Story	Medium	OPEN	5
USN5-102	As an embedded developer, I need the transmitting end of my hardware to speak languages that the receiving devices understand in order to transmit data in the correct format.	Story	Medium	OPEN	3

5/13/2019

SP board - Agile Board - USN Bachelor group 5



USN5-131	As a test engineer, I need the ability to calibrate position and speed of the SADM drive shaft so that I can set correct initial conditions for operation	Story	High	OPEN	3
USN5-161	The motor should have specific nominal electrical interfaces so that it will satisfy the SADM power needs	Story	High	DESIGN	5
USN5-198	As an electrical engineer, I must be familiar with flex-suppliers and characteristics needed for the flex so that we can decide what specifications are needed.	Story	High	OPEN	5
USN5-214	As an electrical engineer, I must design a prototype PCB rigid-flex	Story	Medium	OPEN	3

Issues completed outside of this sprint

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (10)
USN5-186	As team members, verifications for every requirements that are reviewed needs to be set in the VMD	Story	Medium	DONE	3
USN5-200	As requirement responsible, I must ensure structure and order in the requirement pages in Confluence, so that all requirements and user stories are traceable and structured.	Story	Medium	DONE	7

Issues Removed From Sprint

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (5)
USN5-109	As a risk manager I need to write an improved risks management plan	Story	Medium	DESIGN	5



5/13/2019

SP board - Agile Board - USN Bachelor group 5

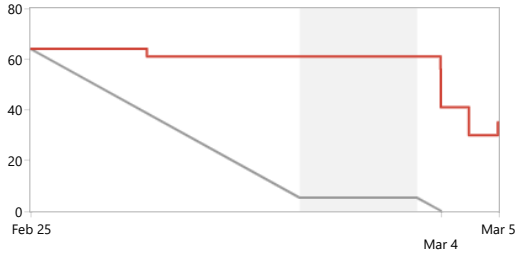


Projects / USN Bachelor group 5 / SP board / Reports

Sprint Report

SP Sprint 7

Closed sprint, ended by Rita Hogstad 25/Feb/19 9:59 AM - 05/Mar/19 9:19 AM [Linked pages](#)



Status Report

* Issue added to sprint after start time

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (34)
USN5-74	As a test engineer, I need a software that can give me diagnostic data in real time	Story	High	DONE	5
USN5-76	As a test engineer, I need a simple way of configuring the communication between the diagnostics system software and hardware	Story	Medium	DONE	5
USN5-77	As a software engineer, I need the diagnostic system to be easily expandable to allow for other diagnostics components	Story	High	DONE	10
USN5-96	As an embedded developer, I need my hardware to be able to control the SADM to be able to measure the result.	Story	Medium	DONE	3
USN5-100	As an embedded developer, I need to make sure that the correct instructions are executed at the correct device for the test results to be correct.	Story	Medium	DONE	5
USN5-102	As an embedded developer, I need the transmitting end of my hardware to speak languages that the receiving devices understand in order to transmit data in the correct format.	Story	Medium	DONE	3
USN5-214	As an electrical engineer, I must design a prototype PCB rigid-flex	Story	Medium	DISCARDED	3
USN5-226 *	As a computer engineer, I need to define a communication protocol so that both hardware- and software-layer can communicate correctly	Story	Medium	DONE	-

Issues Not Completed

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (30 → 35)
USN5-121 *	Information about the solar array position shall be available so that the rest of the system (DS) can use this information.	Story	Medium	DESIGN	- → 5
USN5-131	As a test engineer, I need the ability to calibrate position and speed of the SADM drive shaft so that I can set correct initial conditions for operation	Story	High	TESTING	3
USN5-161	The motor should have specific nominal electrical interfaces so that it will satisfy the SADM power needs	Story	High	DESIGN	5
USN5-198	As an electrical engineer, I must be familiar with flex-suppliers and characteristics needed for the flex so that we can decide what specifications are needed.	Story	High	OPEN	5
USN5-204	The SADM should include all the necessary	Story	High	PRODUCTION	7

5/13/2019

SP board - Agile Board - USN Bachelor group 5

components to function



USN5-222

As designers, we need to decide the final design of our product so we can start choosing materials and setting right dimensions

Story High

DESIGN

10



5/13/2019

SP board - Agile Board - USN Bachelor group 5

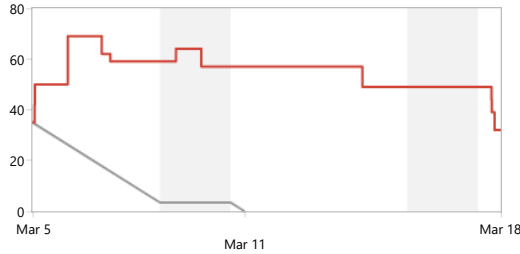


Projects / USN Bachelor group 5 / SP board / Reports

Sprint Report

SP Sprint 8

Closed sprint, ended by Rita Hogstad 05/Mar/19 9:20 AM - 18/Mar/19 3:25 PM [Linked pages](#)



Status Report

* Issue added to sprint after start time

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (22 → 56)
USN5-118 *	As a software engineer, I need a standardized interface for communicating with a database so that the system can save historical data	Story	Medium	DONE	7
USN5-131	As a test engineer, I need the ability to calibrate position and speed of the SADM drive shaft so that I can set correct initial conditions for operation	Story	High	DONE	3
USN5-156 *	The outer spatial dimensions of The Housing of the SADM must be exactly one hundred by one hundred by two hundred millimeters. The inner spatial dimensions depends on the thickness of the housing walls. The drivetrain and other components inside the SADM...	Story	Medium	DONE	- → 7
USN5-198	As an electrical engineer, I must be familiar with flex-suppliers and characteristics needed for the flex so that we can decide what specifications are needed.	Story	High	DONE	5
USN5-204	The SADM should include all the necessary components to function	Story	High	DONE	7
USN5-232 *	For documentation purposes I need to finalize the documents related to Alpha 2 HWL so that they're ready for hand in.	Story	Medium	DONE	- → 7
USN5-237 *	As a group we need to prepare a script for presentation 2	Story	Medium	DONE	- → 5
USN5-238 *	As a group we need to prepare slides for presentation 2	Story	Medium	DONE	- → 3
USN5-246 *	The SADM's weight should not exceed 2kg so that the nano satellite will not be affected by the SADM weight	Story	Medium	DONE	- → 7
USN5-248 *	As mechanical engineers, we need to write documentation for our work	Story	Medium	DONE	- → 5

Issues Not Completed

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (32)
USN5-75 *	As a test engineer, I need a software that can display historical diagnostic data	Story	Medium	DESIGN	7
USN5-87 *	As a test engineer, I need an easy and intuitive way of choosing which diagnostics data I want to look at	Story	Medium	PRODUCTION	5
USN5-121	Information about the solar array position shall be available so that the rest of the system (DS) can use this information.	Story	Medium	DESIGN	5
USN5-161	The motor should have specific nominal electrical interfaces	Story	High	DESIGN	5

5/13/2019

SP board - Agile Board - USN Bachelor group 5

so that it will satisfy the SADM power needs



USN5-222 As designers, we need to decide the final design of our product so we can start choosing materials and setting right dimensions Story ↑ High DESIGN 10

Issues completed outside of this sprint

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (5 → 27)
USN5-49 *	As the one responsible for the presentation tool, the assigned needs to be familiar with Prezi or other presentation tools.	Story	↓ Low	DONE	- → 5
USN5-109 *	As a risk manager I need to write an improved risks management plan	Story	↑ Medium	DONE	5
USN5-153 *	The SADM's weight should not exceed 2kg so that the nano satellite will not be affected by the SADM weight - For Mark2	Story	↑ Medium	DONE	- → 7
USN5-154 *	The volume of the SADM should not exceed 100 x 100 x 200 millimeters so that it will fit inside of an 16u satellite	Story	↑ Medium	DONE	- → 7
USN5-158 *	The motor of the SADM should be marked with a zero reference point so that it will be easier to mate the assembly	Story	↑ Medium	DONE	- → 3



5/13/2019

SP board - Agile Board - USN Bachelor group 5

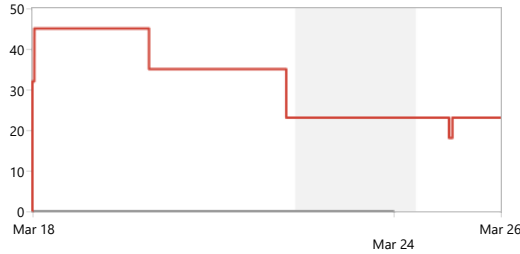


Projects / USN Bachelor group 5 / SP board / Reports

Sprint Report

SP Sprint 9

Closed sprint, ended by Thomas Mundal 18/Mar/19 3:25 PM - 26/Mar/19 9:45 AM [Linked pages](#)



Status Report

* Issue added to sprint after start time

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (27)
USN5-75 *	As a test engineer, I need a software that can display historical diagnostic data	Story	Medium	DONE	7
USN5-87 *	As a test engineer, I need an easy and intuitive way of choosing which diagnostics data I want to look at	Story	Medium	DONE	5
USN5-161 *	The motor should have specific nominal electrical interfaces so that it will satisfy the SADM power needs	Story	High	DISCARDED	5
USN5-222 *	As designers, we need to decide the final design of our product so we can start choosing materials and setting right dimensions	Story	High	DONE	10

Issues Not Completed

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (21 → 23)
USN5-121 *	Information about the solar array position shall be available so that the rest of the system (DS) can use this information.	Story	Medium	DESIGN	5
USN5-251 *	As an embedded developer, I need to implement functionality supporting a subscriber system for sensor data to comply with the SatStat communication protocol.	Story	Medium	PRODUCTION	3
USN5-257 *	As the HWL has to function at all times, error handling must be implemented to ensure that the system doesn't crash.	Story	Medium	PRODUCTION	3
USN5-263 *	As a successor of the project, I need descriptive documents/diagrams of the HWL to further develop the system.	Story	Medium	DESIGN	5 → 7
USN5-270 *	As an electrical engineer, I must decide which supplier and kind of sensor to measure the position of the shaft so that the DS can get feedback.	Story	High	OPEN	5

5/13/2019

SP board - Agile Board - USN Bachelor group 5

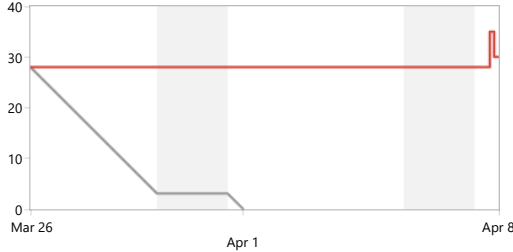


Projects / USN Bachelor group 5 / SP board / Reports

Sprint Report

SP Sprint 10

Closed sprint, ended by Rita Hogstad 26/Mar/19 9:45 AM - 08/Apr/19 4:29 PM [Linked pages](#)



Status Report

* Issue added to sprint after start time

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (5)
USN5-146	As a test engineer, I would like to not have to adjust COM settings every time I have to connect to the HW layer so that the software is easier to use	Story	Medium	DONE	5
USN5-247	Implement error handling and connection timeouts for SWL	Story	Medium	DISCARDED	-

Issues Not Completed

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (30)
USN5-121	Information about the solar array position shall be available so that the rest of the system (DS) can use this information.	Story	Medium	DESIGN	5
USN5-251	As an embedded developer, I need to implement functionality supporting a subscriber system for sensor data to comply with the SatStat communication protocol.	Story	Medium	PRODUCTION	3
USN5-257	As the HWL has to function at all times, error handling must be implemented to ensure that the system doesn't crash.	Story	Medium	PRODUCTION	3
USN5-263	As a successor of the project, I need descriptive documents/diagrams of the HWL to further develop the system.	Story	Medium	DESIGN	7
USN5-268 *	As an electrical engineer, I must design a flexible PCB	Story	Medium	DESIGN	7
USN5-270	As an electrical engineer, I must decide which supplier and kind of sensor to measure the position of the shaft so that the DS can get feedback.	Story	High	OPEN	5

5/13/2019

SP board - Agile Board - USN Bachelor group 5

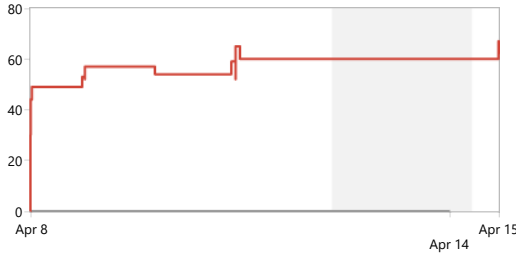


Projects / USN Bachelor group 5 / SP board / Reports

Sprint Report

SP Sprint 11

Closed sprint, ended by Rita Hogstad 08/Apr/19 4:29 PM - 15/Apr/19 9:13 AM [Linked pages](#)



Status Report

* Issue added to sprint after start time

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (21)
USN5-257 *	As the HWL has to function at all times, error handling must be implemented to ensure that the system doesn't crash.	Story	Medium	DONE	3
USN5-270 *	As an electrical engineer, I must decide which supplier and kind of sensor to measure the position of the shaft so that the DS can get feedback.	Story	High	DISCARDED	5
USN5-275 *	As a test engineer, I would like to see the current value of a sensor input so that I can keep an eye on the current status of the system	Story	Medium	DONE	1
USN5-277 *	Mech: Make a lid for Mark3 to attach the motor	Story	Medium	DONE	7
USN5-279 *	As a test engineer, I need the ability to adjust accepted values for sensor readings to be able to verify that the system is operating as intended	Story	Medium	DONE	5

Issues Not Completed

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (50 → 62)
USN5-121 *	Information about the solar array position shall be available so that the rest of the system (DS) can use this information.	Story	Medium	PRODUCTION	5
USN5-251 *	As an embedded developer, I need to implement functionality supporting a subscriber system for sensor data to comply with the SatStat communication protocol.	Story	Medium	TESTING	3 → 10
USN5-263 *	As a successor of the project, I need descriptive documents/diagrams of the HWL to further develop the system.	Story	Medium	DESIGN	7
USN5-268 *	As an electrical engineer, I must design a flexible PCB	Story	Medium	PRODUCTION	7
USN5-276 *	Mech: Make a structure for Mark3	Story	Medium	PRODUCTION	7
USN5-278 *	As a risk manager, I must analyse risks	Story	Medium	PRODUCTION	5
USN5-281 *	As a test engineer, I want the ability to save accepted sensor value ranges to a template, so that I can load the same settings for testing at any point in time	Story	Medium	PRODUCTION	3
USN5-286 *	Mech: Redesign flex shaft design	Story	Medium	PRODUCTION	10
USN5-287 *	Mech: Make fittings for D-sub in the Mark3 housing	Story	Medium	PRODUCTION	3

5/13/2019

SP board - Agile Board - USN Bachelor group 5

USN5-290 * Complete order of electrical equipment. Story Medium DESIGN - -> 5



Issues completed outside of this sprint

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (- -> 1)
USN5-284 *	Attend meeting with mechanical lab at KONGSBERG to aquire information about the Mark3 parts	Story	Medium	DONE	- -> 1



5/13/2019

SP board - Agile Board - USN Bachelor group 5

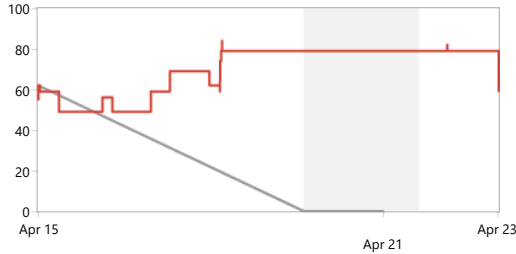


Projects / USN Bachelor group 5 / SP board / Reports

Sprint Report

SP Sprint 12

Closed sprint, ended by Rita Hogstad 15/Apr/19 9:20 AM - 23/Apr/19 9:11 AM [Linked pages](#)



Status Report

* Issue added to sprint after start time

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (43 → 58)
USN5-251	As an embedded developer, I need to implement functionality supporting a subscriber system for sensor data to comply with the SatStat communication protocol.	Story	Medium	DONE	10
USN5-276	Mech: Make a structure for Mark3	Story	Medium	DONE	7
USN5-281	As a test engineer, I want the ability to save accepted sensor value ranges to a template, so that I can load the same settings for testing at any point in time	Story	Medium	DONE	3
USN5-286	Mech: Redesign flex shaft design	Story	Medium	DONE	10
USN5-287	Mech: Make fittings for D-sub in the Mark3 housing	Story	Medium	DONE	3
USN5-296 *	Make Mark3 ready for 3D printing	Story	Medium	DONE	- → 7
USN5-305 *	Redesign the motor holder of Mark3	Story	Medium	DONE	5
USN5-307 *	Make a mechanical stop for the main shaft	Story	Medium	DONE	- → 5
USN5-308 *	Redesign middle wall on Mark3 model for the mechanical stop	Story	Medium	DONE	- → 3
USN5-312 *	Mech: Create gears for Mark3	Story	Medium	DONE	5

Issues Not Completed

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (44 → 59)
USN5-121	Information about the solar array position shall be available so that the rest of the system (DS) can use this information.	Story	Medium	PRODUCTION	5
USN5-250 *	As an embedded developer, I need to optimize the current version of the HWL to prepare for further development.	Story	Medium	PRODUCTION	7
USN5-268	As an electrical engineer, I must design a flexible PCB	Story	Medium	TESTING	7
USN5-278	As a risk manager, I must analyse risks	Story	Medium	PRODUCTION	5
USN5-290	Complete order of electrical equipment.	Story	Medium	PRODUCTION	5
USN5-297 *	Start 3D printing of Mark3 parts	Story	High	PRODUCTION	- → 10
USN5-298 *	Design connection interface between flex and wires.	Story	Medium	DESIGN	10
USN5-311 *	As a user of the HMI I want a list of available instructions so that I don't have to remember all of	Story	Medium	DESIGN	- → 5

5/13/2019

SP board - Agile Board - USN Bachelor group 5

them



USN5-313 * Create a short description of how we use MS HoloLens in our project. Story Medium DESIGN 5



Issues Removed From Sprint

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (7)
USN5-263	As a successor of the project, I need descriptive documents/diagrams of the HWL to further develop the system.	Story	Medium	DISCARDED	7



5/13/2019

SP board - Agile Board - USN Bachelor group 5



Projects / USN Bachelor group 5 / SP board / Reports

Sprint Report

SP Sprint 13

Closed sprint, ended by Rita Hogstad 23/Apr/19 9:13 AM - 29/Apr/19 10:49 AM [Linked pages](#)



Status Report

* Issue added to sprint after start time

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (44 → 61)
USN5-121	Information about the solar array position shall be available so that the rest of the system (DS) can use this information.	Story	Medium	DONE	5
USN5-250	As an embedded developer, I need to optimize the current version of the HWL to prepare for further development.	Story	Medium	DONE	7
USN5-268	As an electrical engineer, I must design a flexible PCB	Story	Medium	DONE	7
USN5-278	As a risk manager, I must analyse risks	Story	Medium	DONE	5
USN5-290	Complete order of electrical equipment.	Story	Medium	DONE	5
USN5-297	Start 3D printing of Mark3 parts	Story	High	DONE	10
USN5-306	Make exits for the flex out of the capsule housing for Mark3	Story	Medium	DONE	- → 5
USN5-309	Make holes for screws on the structure of Mark3	Story	Medium	DONE	- → 5
USN5-313	Create a short description of how we use MS HoloLens in our project.	Story	Medium	DONE	5
USN5-318 *	Mech: Make an adapter to the stepper motor for Mark3	Story	Medium	DONE	- → 7

Issues Not Completed

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (29)
USN5-285	Mech: Make 2D drawings of the Mark3 parts	Story	Medium	TESTING	7
USN5-298	Design connection interface between flex and wires.	Story	Medium	PRODUCTION	10
USN5-302	As a test engineer I want the ability to define behaviour and parameters to measure during a test run for the SADM	Story	Medium	OPEN	7
USN5-311	As a user of the HMI I want a list of available instructions so that I don't have to remember all of them	Story	Medium	PRODUCTION	5

5/13/2019

SP board - Agile Board - USN Bachelor group 5

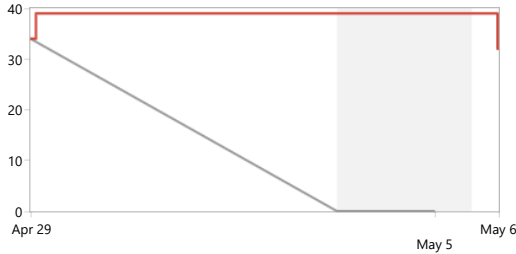


Projects / USN Bachelor group 5 / SP board / Reports

Sprint Report

SP Sprint 14

Closed sprint, ended by Rita Hogstad 29/Apr/19 10:53 AM - 06/May/19 9:36 AM [Linked pages](#)



Status Report

* Issue added to sprint after start time

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (7)
USN5-285	Mech: Make 2D drawings of the Mark3 parts	Story	Medium	DONE	7

Issues Not Completed

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (32)
USN5-298	Design connection interface between flex and wires.	Story	Medium	PRODUCTION	10
USN5-302	As a test engineer I want the ability to define behaviour and parameters to measure during a test run for the SADM	Story	Medium	OPEN	7
USN5-311	As a user of the HMI I want a list of available instructions so that I don't have to remember all of them	Story	Medium	TESTING	5
USN5-323	Write about choice of materials in the report	Story	Medium	DESIGN	5
USN5-326 *	Update Gantt diagram	Story	Medium	PRODUCTION	5



5/13/2019

SP board - Agile Board - USN Bachelor group 5

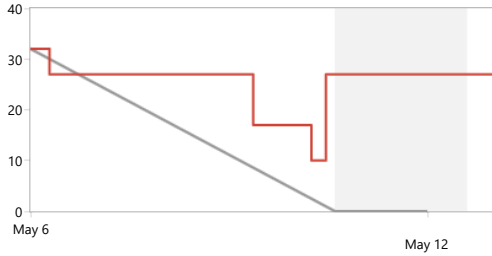


Projects / USN Bachelor group 5 / SP board / Reports

Sprint Report

SP Sprint 15

Active sprint 06/May/19 9:36 AM - 12/May/19 9:36 AM [Linked pages](#)



Status Report

* Issue added to sprint after start time

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (22)
USN5-302	As a test engineer I want the ability to define behaviour and parameters to measure during a test run for the SADM	Story	Medium	DONE	7
USN5-311	As a user of the HMI I want a list of available instructions so that I don't have to remember all of them	Story	Medium	DONE	5
USN5-323	Write about choice of materials in the report	Story	Medium	DONE	5
USN5-326	Update Gantt diagram	Story	Medium	DONE	5



Issues Not Completed

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (27)
USN5-295 *	Elec: Build cabinet	Story	High	DESIGN	7
USN5-298	Design connection interface between flex and wires.	Story	Medium	PRODUCTION	10
USN5-320 *	Elec: Plan B flexprint. Another supplier or alternate solution?	Story	Highest	PRODUCTION	10

5/20/2019

SP board - Agile Board - USN Bachelor group 5

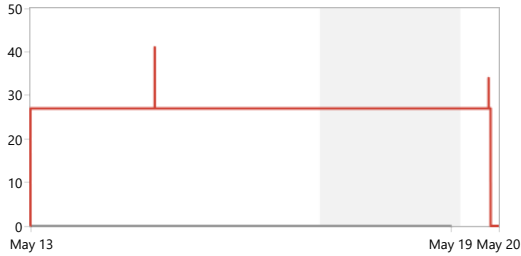


Projects / USN Bachelor group 5 / SP board / Reports

Sprint Report

SP Sprint 16

Closed sprint, ended by Rita Hogstad 13/May/19 9:03 PM - 20/May/19 1:09 PM [Linked pages](#)



Status Report

* Issue added to sprint after start time

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (48)
USN5-295 *	Elec: Build cabinet	Story	High	DONE	7
USN5-298 *	Design connection interface between flex and wires.	Story	Medium	DONE	10
USN5-303 *	As a test engineer I want the ability to save a defined test-run as a template so that I can load the same settings for a future test run	Story	Medium	DONE	7
USN5-304 *	As a test engineer I want the ability to load test-run settings from a template and initiate a test run based on loaded or defined parameters	Story	Medium	DONE	7
USN5-320 *	Elec: Plan B flexprint. Another supplier or alternate solution?	Story	Highest	DONE	10
USN5-325 *	Make a manual for how to put Mark3 together, in SolidWorks	Story	Medium	DONE	7



Bachelor of Engineering

Project appendix

Winter semester 2019

Confluence Export - Meeting notes

This appendix is in partial fulfilment of the bachelors project report.

Writer's Name	SatLight
Group Members	Rita Hogstad, Erica Fegri, Thomas Mundal, Ming Kit Wong, Håvar Østrem, Jon Skjelsbæk
Internal Supervisor	José Manuel Martins Ferreira
External Supervisor	Eirik Demmo Normann
Institution	University of South-Eastern Norway
Company	Kongsberg Defence & Aerospace

Abstract

This document contains every Meeting notes document exported from Confluence. These are notes both from meetings with the internal supervisor and the external supervisor.

I hereby declare that this project report is my own work, that I have only made use of the cited and/or acknowledged documents/resources and that this report has not been previously submitted as academic coursework elsewhere.

City, Date Kongsberg, May 20, 2019	Signature
---------------------------------------	-----------

Contents

- A Meeting notes 2**
- A.1 Jan 10 2019 2
- A.2 Jan 11 2019 5
- A.3 Jan 16 2019 6
- A.4 Jan 18 2019 9
- A.5 Jan 23 2019 11
- A.6 Jan 25 2019 13
- A.7 Jan 30 2019 15
- A.8 Feb 01 2019 17
- A.9 Feb 08 2019 18
- A.10 Feb 13 2019 20
- A.11 Feb 15 2019 22
- A.12 Mar 27 2019 23
- A.13 Mar 06 2019 25
- A.14 Mar 18 2019 27
- A.15 Apr 10 2019 29
- A.16 Apr 17 2019 31
- A.17 Apr 18 2019 32
- A.18 Apr 24 2019 34
- A.19 May 05 2019 36

Appendix A

Meeting notes

2019-01-10 Meeting notes

Date

10 Jan 2019

Participants

- Thomas Mundal
- Jon Skjelsbek
- Erica Fegri
- Rita Hogstad
- Ming Kit Wong
- Håvar Østrem

Goals

- Establish a project model
- Agree on elements in project model

Discussion topics

Item	Presenter	Notes
Project Model	Thomas Mundal	<ul style="list-style-type: none"> • Based on SCRUM • Story workflow based on v-model
Weekly customer meeting		The time to meet with the external supervisor / customer
Project management tool	Thomas Mundal	What project management software tool to use.

Action items

- Write document describing the process model in detail for the report
- Incorporate the cost of project management tool into the budget
- Order subscription for Jira and Confluence
- Identify and assign group roles
- Agree on weekly process meeting

Decisions

✓

- 1 Use Jira and Confluence. In total \$20 per month
- Y Use a project model based on scrum with a workflow inspired by v-model
- Y Customer meeting every wednesday

2019-01-11 Meeting notes

Date

11 Jan 2019

Participants

- Thomas Mundal
- Erica Fegri
- Jon Skjelsbek
- Rita Hogstad
- Håvar Østrem
- Ming Kit Wong

Goals

- Form process model
- Agree on and assign roles

Discussion topics

Time	Item	Presenter	Notes
	Roles	Ming Kit Wong	See document describing process model
	Process model	Thomas Mundal	See document describing process model

Action items

- Agree on sprint start every week
-

Decisions

✔ Meetings with internal supervisor every thursday at 10:00

✔ Process model established based on SCRUM

✔ Roles defined and assigned

2019-01-16 Meeting notes

Date

16 Jan 2019

Participants

- Thomas Mundal
- Håvar Østrem
- Erica Fegri
- Jon Skjelsbek
- Ming Kit Wong
- Rita Hogstad
- Eirik Demmo Normann

Goals

- Get scope of the assignment
- Budget
- Show the group progress so far
 - Process model
 - Roles
 - Project tools (software)

Discussion topics

Time	Item	Presenter	Notes
	Assignment scope	Eirik Demmo Normann	<ul style="list-style-type: none"> • Scope is not complete • The group should form task description based on the information discussed <ul style="list-style-type: none"> • A preliminary definition is formed and discussed
	Budget		<ul style="list-style-type: none"> • KSS will pay for prototype 3D print if applicable • Budget will be further discussed between Eirik and Aage • Possibly use third parties for professional 3d prints for prototype
	Group progress		<ul style="list-style-type: none"> • Group has an agile process model based on SCRUM • Group has Jira/Confluence as project tools • Group has selected roles

Free notes

- Got an example of how they shape spec on a fullsize satellite
- Electric performance
- Reliability
- Cubesat size 12u (find kg)
- Solid radar mechanism at 2h. Max 3kg
- SADM for cubesat
- Design, prototype and testing to a certain level.
- Pay attention to vib and shock test, but not necessarily do so.
- Software: Baseline that can be built on later.
- Requirements: About it from powerpoint
- Budget
- Didn't talk about budget, "yes we'll get things done"
- Prototype: KSS stands for purchasing it
- Envision parts for a few thousand for 3d printing
- Can go up to 50-60 thousand kroner to build a prototype.
- Apworks in germany 3d print in metal. Uses software that spits out a price for the 3dprint of a model.
- Mechanical analysis should be done (verification)

Actions:

Eirik:

- Next time, go through with Åge regarding budget
- Remind Aage regarding contract
- Find an example of VCD
- Possible to find some information on risk analysis
- Along with spec, set up a verification model

Us:

- Send suggestions for the text that we have formulated
- Send link to confluence: <https://usnbachelor.atlassian.net/wiki/spaces/USA/>
- Read about cubesat, familiarize yourself with terminology and common challenges.
- Verification verification document VCD
- Doors (software)
- Produce prototype, test equipment and control and monitoring software.

Action items

- Eirik: Talk to Aage about budget
- Eirik: Follow up information about the contract with Aage
- Eirik: Find example on VCD
- Eirik: Possibly find example or information about risk analysis
- Eirik: Together with spec, create verification model
-

- Thomas Mundal Send preliminary task description text to Eirik for review
- Thomas Mundal Send link to confluence: <https://usnbachelor.atlassian.net/wiki/spaces/USA/>
- Group: Read about CubeSat, learn about terminology and usual challenges

Decisions

- Spec should be ready for next meeting



2019-01-18 Meeting notes

Date

18 Jan 2019

Participants

- Thomas Mundal
- Ming Kit Wong
- Rita Hogstad
- Jon Skjelsbek
- Håvar Østrem
- Erica Fegri

Goals

- Inform supervisor about the model
- Feedback on our project model
- Feedback on our project plan
- Feedback on our plan for first presentation
- Inform supervisor about the project

Discussion topics

Item	Presenter	Notes
Model	Thomas Mundal	<ul style="list-style-type: none"> • why did you consider only three stages in jira? Req, validate req, design, prototyping, verification of prototype, production and testing, validation (is the customer satisfied?). • Epics Atlassian ha en link til Atlassian guide for å ha oversikt over terminologien • what are your stakeholders?
Project plan	Rita Hogstad	<ul style="list-style-type: none"> • Fix decimal in gantt • Not certain if the presentations alone are milestones. Has to account for why we make the choices that we make.



First presentation	Ming Kit Wong	<ul style="list-style-type: none"> • Make sure we know the company's objective before the presentation • Clear description of the outcome of the project, what are we making, what is the goal? • Include the state of the art of the field, how does it compare to the existing market, what is our competition. Can have a link at the press if there's no time to present it • Write a script, weigh your words carefully. Speak at a normal speed • Tell about roles, picture of us and our roles with a short description • Record the presentation on beforehand, introduction to what we are going to talk about.
Other topics		<ul style="list-style-type: none"> • Should we focus on prototype or pretend it is going into real production
Next meeting		<ul style="list-style-type: none"> • Skype for business or hangouts, meeting room in the fifth floor with mic and cam • Send everyone's email address to Jose

Action items

- Fix decimal numbers in gantt [Rita Hogstad](#)
- Share useful documents with Jose [Jon Skjelsbek](#) [Thomas Mundal](#)
- Write a script for first presentation
- Record a video for the first presentation about 5 minutes
- Make ready for video conference for next meeting
- Discuss the workflow based on feedback from Jose, and the v-model from KSS
- Do stakeholder analysis

Decisions

- Video conference will happen from our group room using google hangouts

2019-01-23 Meeting notes

Date

23 Jan 2019

Participants

- Rita Hogstad

Thomas Mundal Erica Fegri Ming Kit Wong Jon Skjelsbek Håvar Østrem

Eirik Demmo Normann

Goals

- Clarify whether DSS has any specific technical or functional requirements for the prototype. Which environment is reasonable for the prototype to be developed / tested for? (Ground / Launch / Orbit)
- VCD
- Clarify the status of contract and budget.
- Examples of risk analysis
- Jon: Wants to know where Eirik is developing the FPGA / SADE. It is desirable for us to develop such a device ourselves, since we want to facilitate testing. If such a device is to be developed, it is interesting to know something about what comes from "Main Computer" or, if so, if this is not relevant.
- Submit revised draft text
- Submit a plan for handling requirements (Erica) - How does Eirik want to have insight into this?
- Submit project plan

Discussion topics

1. Clarify whether KSS has any specific technical or functional requirements for the prototype. Which environment is reasonable for the prototype to be developed / tested for? (Ground / Launch / Orbit)

- Eirik Starts on a spec, filled out the requirements they know they should have. Made a framework, set up requirements that they think are important to them, and we should put in demands ourselves.
- Electrical at kss will go through spec with us.
- Specen has its own environmental agenda, ground launch orbit, drop ground conditions
- We only get spec by SADM
- We have to make the spec ourselves

2. VCD

- Eirik can show us an example of how not to run VCD, do not use excel

3. Clarify contract and budget status.

- The contract looks nice and it has been forwarded to HR. Send mail to Aage.
- Aage confirmed that around 50k should be ok

4. Examples of risk analysis

- Obtained risk sheet from eirik
- Wouldn't spend much time at risk
- Assess what risks exist and take it into account
- Our main focus is project risk

5. Jon: Wants to know: where Eirik stands for the development of FPGA / SADE. It is desirable for us to develop such a device ourselves, since we want to facilitate testing. If such a device is to be developed, it is interesting to know something about what comes from "Main Computer" or, if so, whether this is not relevant.

- there is not much that comes from main computer, sends the number of steps and one direction
- We can care about position, if it costs little money and we have time.
- In the spec it says that we should not use position sensor
- In relation to the test system, we can have position sensor, it seems interesting.
- What the engine sees is the most important thing
- The satellite must be compatible with the program we use
- We can use arduino to control.

6. Present revised project task text

- Review SADO?
- Have we understood the fact that it is new to kss and want to get on the market? It's OG

7. Submit a plan for handling requirements (Erica) - How does Eirik want to have insight into this?

- Like to distinguish requirement spec with requirement motivation - which describes the background for the requirement.
- Seems excel is not suitable
- They want insight into req: Update weekly is fine
- Dynamic loads and environmental
- Write reference for documents from KSS.
- Important to be able to link documents to each other
- Thermal environment and radiation environment, dynamic loads (?), What are the normal requirements for nano / micro satellites?
- It will be us who owns the document, so we have to decide on the requirements

8. Project plan

- Lead time on products, correct chronology, correct parallel, Action items
- Eirik will send us documents of requirements etc.
- Eirik will send us a link to supplier on flex

2019-01-25 Meeting notes

Date

25 Jan 2019

Participants

- Rita Hogstad Thomas Mundal Ming Kit Wong Erica Fegri Håvar Østrem
Jon Skjelsbek Jose M. M. Fereirra

Goals

- Present revised task text
- Explain the hardware related choices for the test environment
- Preview stakeholder analysis draft

Discussion topics

1. Present revised task text

Short, but what we had was good

2. Explain the hardware related choices for the test environment

Jose: Are we supposed to make a test for testing what parts fail.

- Useful: identify clearly what the tasks are.
- Would be interesting if it would be followed with explanations comments.
- The documents we send needs more information in the form of text. Especially for diagrams.

3. Preview stakeholder analysis draft

- How did we get to this information?
- Including references will help us understand the objective.
- Why do we use this diagram?
- Good for self organization, good for knowing who we interact with, good for development strategy, we should have a text describing the diagram

Other:

Erica will make the new requirement sheet available for Jose

- Jose is adding a link for a slide template.
- Action: Progress plan up to first presentation, jose wants to see how we are dealing with the first presentation
- Will we make video?
- On the link we gave to Jose we should give an explanation of what the different pages are

- Thomas must fix so that Jose can see the sprints in confluence
- Jose can get all the "powerpoints" and everything.

Action items

- Make a plan for what we are going to do each the before the first pres
- Jose wants to look at all presentations before the presentation, to give us tips
- Erica will send the requirement document when it is "finished"
- Make a description of what the sites are, on the links shared with Jose
-

Decisions

2019-01-30 meeting notes

Date 30 Jan 2019

Participants

- Håvar Østrem
- Erica Fegri
- Rita Hogstad
- Jon Skjelsbek
- Ming Kit Wong
- Eirik Demmo
- Carsten Haavardtun

Goals

- Verification Management Document
- SADO has been renamed DS (Diagnostics System)
 - Software layer
 - Hardware layer
 - Present requirements for software layer
 - Present preliminary form of requirements for hardware layer
 - Present description of hardware layer
- Inform externals about first presentation

Discussion topics

Item	Presenter	Notes
Before meeting agenda	Eirik Demmo	<ul style="list-style-type: none"> • Eirik is invited to the presentation. • Eirik recommends stepper motor. Phytron • Eirik strongly recommends conventional grade components. Space grade is extremely expensive. • Eirik recommends not to pay attention to ISO standards.
Verification Management Document	Ming Kit Wong	<ul style="list-style-type: none"> • Ming gave a brief presentation of the VMD to Eirik and Carsten • Eirik approved. • Carsten recommended a column for referencing to • Carsten asks where we close our requirements • Carsten recommended shortening the requirements list and only use must and should requirements.

SADO has been renamed DS	Jon Skjelsbek	<ul style="list-style-type: none"> • Jon gives a brief presentation of DS • Eirik recommends using MIL IO standard
Test in Vacuum Chamber	Eirik Demmo	<ul style="list-style-type: none"> • We might be able to test prototypes in vacuum chamber <ul style="list-style-type: none"> • We don't have green light as of now. • The prototype must be vakuum compatible.
Vibration and inertia	Carsten Haavardtun	<ul style="list-style-type: none"> • Carsten recommends: Pay attention to vibration induced by moving components in the SADM such as gears. • Solar arrays have a large mass and inertia • Solar arrays parametres are extremely important • Yes. Satellites may use light sensors to verify position. • Carsten recommends: evaluate all requirements, some might not be necessary.

Action items

- Eirik: Verify that Aage is aware of the presentation next monday.
- Eirik: MIL-IO standard

2019-02-01 Meeting notes

Date

01 Feb 2019

Participants

- Håvar Østrem
- Erica Fegri
- Rita Hogstad
- Jon Skjelsbek
- Ming Kit Wong
- Thomas Mundal (streaming via Hangout)
- José Ferreira (streaming via Hangout)

Goals

- Documents
- Presentation

Discussion topics

Item	Presenter	Notes
Documents	SatLight	<ul style="list-style-type: none">• José did not receive documents yet• Document issue was resolved
Presentation	SatLight	<ul style="list-style-type: none">• Satlight did a presentation trial.• José gave us feedback on our performance

2019-02-08 Meeting notes

Date

08 Feb 2019

Participants

- Ming Kit Wong
- Rita Hogstad
- Håvar Østrem
- Thomas Mundal
- Jon Skjelsbek
- Jose

Goals

- Requirements definition for Diagnostics System software layer
- Requirements definition for Diagnostics System hardware layer
- Availability for 2nd and 3rd presentation

Discussion topics

Presenter	Item	Notes
Thomas Jon	Requirements definition for Diagnostics System software layer Requirements definition for Diagnostics System hardware layer	<ul style="list-style-type: none"> • Req def in the title as well. • A place where acronyms are explained • What FPGA? Maybe drop the word? • Looking for milestones • How do the team validate the requirements? • Both layouts were shown. Jose suggests making the first column more visible to separate the cards.
Rita	Availability for 2nd and 3rd presentation	<ul style="list-style-type: none"> • Jose is available 15th of March for the second presentation. Needs to get confirmation from Aage • Jose is available 6th and 7th of June for the third presentation. Aage needs to be informed and invited.

Action items

- Get Aage to confirm 15th of March for second presentation

- Get Aage to confirm 6th or 7th of June for the third presentation

Decisions

2019-02-13 Meeting notes

Date

13 Feb 2019

Participants

- Jon Skjelsbek
- Thomas Mundal
- Erica Fegri
- Ming Kit Wong
- Rita Hogstad
- Håvar Østrem
- Eirik Demmo Normann

Goals

1. Prototype presenting
2. Chosen requirements (flex, cabinet, solar arrays)
3. Software layer and DS

Discussion topics

Time	Item	Presenter	Notes
10:05 - 10:15	Prototype	Rita Hogstad Ming Kit Wong Thomas Mundal	<ul style="list-style-type: none"> • Contactless Position Sensor lager alt for stor sensor, samarbeid er ikke aktuelt. • Flying leeds er fint, pinne jobb tar lang tid i realiteten, vurder lukka/åpen shaft.
10:15 - 10:45	Chosen requirements (flex, cabinet, solar arrays)	Erica Fegri Håvar Østrem Ming Kit Wong Rita Hogstad	<ul style="list-style-type: none"> • 16U (2x8U) not 8U. • Output voltage between 17.5 and 19V, 14 electrical sections • Presented idea of cabinets, positive response (KS uses racks / cabinets for testing themselves) • Board equipment from TESS • mekklab (place to borrow equipment for making cupboards) • Flex may be designed in E3?
10:45 - 10:55	Software layer and DS	Thomas Mundal Jon Skjelsbek	<ul style="list-style-type: none"> • Initial qualification test, not acceptance testing

Action items

- Eirik: organize meeting with meklab
- Eirik: organize meeting with electronics people
- Eirik: organize meeting with testing people
- Eirik: book meeting rooms at KS and invite some people
- research 16U SA Ming Kit Wong Rita Hogstad Håvar Østrem Erica Fegri

Decisions

Y No cooperation with Contactless Position Sensor group

Y Blackboard / automation cabinets are relevant.

Y 16U (2x8U) not 8U

2019-02-15 Meeting notes

Date

15 Feb 2019

Participants

- Jon Skjelsbek
- Thomas Mundal
- Håvar Østrem
- Ming Kit Wong
- Rita Hogstad
- Erica Fegri
- José M. M. Ferreira

Goals

- Timeline
- Updated requirements document SW layer

Discussion topics

Item	Presenter	Notes
Timeline	Rita Hogstad Jon Skjelsbek Thomas Mundal Ming Kit Wong	<ul style="list-style-type: none"> • Presented the timeline for mechanical and computer science • By the second presentation (15th of March), both the software- and hardware layer should be able to receive and interpret data, as well as transmitting in the correct format.
Updated requirements document SW layer	Thomas Mundal	<ul style="list-style-type: none"> • No noteworthy feedback, seems ok.

Action items

- [Jon Skjelsbek](#) have to catch up on work.

Decisions

2019-02-27 Meeting notes

Date

27 Feb 2019

Participants

- Erica Fegri
- Ming Kit Wong
- Rita Hogstad
- Håvar Østrem
- Jon Skjelsbek
- Thomas Mundal
- Eirik Normann
- Aage Vejlgaard Sørensen
- Lars Helge Surdal

Goals

- Sign declaration of confidentiality
- Questioning and meeting Lars Helge

Discussion topics

Time	Item	Presenter	Notes
10:00			Ongoing questions regarding flexprint, solar arrays and other electromechanical were discussed and answered. Very helpful, Lars Helge will return next week if its possible.
11:30	Declaration of confidentiality	Aage V. Sørensen	Statement of secrecy was signed by the students and Aage has completed a security brief.
12:00	2. presentation		The time for the 2nd presentation has been moved to Wednesday 13.03.19 at 09:00.

Action items

- Eirik will send us spreadsheets for derating of wires (find power conductivity at several leaders)
- Thomas Mundal Bring a passport / driver's license for identification on Wednesday 06.03.19

- Jon Skjelsbek Bring a passport / driver's license for identification on Wednesday 06.03.19
- Aage establishes an internal account for the use of Meklab workshop and services

Decisions

The time for the 2nd presentation has been moved to Wednesday 13.03.19 at 09:00.

Y Aage meets us Wednesday 06.03.19 for verification of ID

Y Tidspunkt for 2. presentasjon er flyttet til onsdag 13.03.19 klokken 09:00.

2019-03-06 Meeting notes

Date

06 Mar 2019

Participants

- Thomas Mundal
- Ming Kit Wong
- Rita Hogstad
- Jon Skjelsbek
- Håvar Østrem
- Erica Fegri

Goals

- Demo prototype
- Revise requirements for mechanical and electro

Discussion topics

Mechanical (Rita):

- Effect gear: advantageous to gear a lot when the motor should move slowly
- Constant effect or constant current?
- Stepper motors can hit eigen frequency when stepping
- Anti backlash gear
- Stefan Boltzmann
- Flex tape
- Temperatures on other nanosats

Mechanical (Ming Kit):

- Constant voltage gives greater loss of power at higher speed due to inductance
- More torque out of courage
- 10: 1 might give 8: 1
- Kt-factor
- Anti backlash gear
- Potensiometer
- What happens to motors in vacuum? Fumigation, look in ESA / NASA documents
- R2-14: not 8 Nm, closer to 8 mNm, but that's too small
- Amplification factor = Safety Factor = 2
- Quasi-static requirement: interpreted correctly. Can, not must. Insert a value other than 70 Nm
- R2-20 arcsec requirement,
- Normal temperature range is -30 til +60 celsius

Electrical (Håvar):

- Kt .. regulation technique
- current and rpm
- constant voltage / constant current
- torque, current, stability
- measure kt
- Boltzman mechanical heating
- simply R2-15 (from Eirik)
- Examine the D-sub connector

Action items

- Eirik: Find and send test plan
- Erica: Find motor driver
- Research D-sub connector

Decisions

- We vil order motor driver ourselves

2019-03-18 Group meeting

Date

18 Mar 2019

Participants

- Rita Hogstad
- Håvar Østrem Jon Skjelsbek Thomas Mundal Erica Fegri Ming Kit Wong

Goals

Discussion topics

Risk:

- Poor risk management in Scrum
- Learn along the way
- Incorporated into retrospective, short sprinters that reduce risk
- Stops inside the twist capsule so the flex does not rotate so far that the flex becomes dead
- Read through risk management document. Need revision
- How to document risk and what to do with these risks?
- Everyone must take into account risk when it is documented. Mention that risks are taken into account and what we did with it
- Write about technical risks and how to solve it in relation to the process model.
- Explain project risks
- ALL: Enter project risks and technical risks into SWOT

Dates and Milestones:

- Jon exams April 2, machine and electrical exam April 5, April 4 and April 5 thomas will have an oral exam. Rita must enter this into the project plan
- We will use google calendar to enter dates
- Allows to completely free from core time from March 21st until the exam is finished.

Documentation

- Ask Karoline which delivery method is best.
- Must be done regarding report:
 1. Merge documentation: May 12th.
 2. Review content: May 13th
 3. Attachments: May 14th
 4. Source control: May 15th

5. Proofreading: May 16th.
6. Print: 21. May 21st
7. Delivery: May 22

Work schedule:

Mechanical:

- April 8: Talk to Meklab
- April 12: Mark3 Finished CAD + finished deciding materials
- April 17: M3 Finished in plastic
- April 23: Production of Mark3 starts
- April 26: Mark3 must be ready for test

Electro:

- March 20 : Talk to Lars Helge
- March 22: Order components
- March 27: Order flex
- April 8: Decide connector / interface
- April 12: Finish final concept for Mark3

Data:

- March 22: Arduino bracket ordered / made
- March 30: Automated test v1
- April 12: Gear ratio clear
- April 12: Implement position sensor
- April 13: Automated test done
- April 15: Position sensor ready
- April 17: DS ready for use

Action items



Decisions

2019-04-10 Meeting notes

Date

10 Apr 2019

Participants

- [Ming Kit Wong](#)
- Rita
- Erica
- Håvar
- Jon
- Thomas
- Lars Helge
- Carsten


Discussion topics

Item	Presenter	Notes
Progress plan and tips from Carsten	Thomas	<ul style="list-style-type: none"> • Most such projects are overgrown. Choose something to focus on and document what might have to be taken over by others or not. • Be sure to have intermediate goals so that the final product can be measured in how far we have come in the process, if we should not reach goals. • Write a diary for product development! Must be low threshold to use this
Ordering of flex	Electro	Lars Helge said that the flexman did not have time to look at the design of the students, but that it was certainly good. Carsten suggested calling the manufacturer to hear about the risk of short circuit in the design. Lars Helge said book in the way.
Miscellaneous		The electro students got time alone with Lars Helge. Carsten stayed with the rest and considered the design of the M3 and came up with tips on ball bearings and tolerances in general. The design of the M3 will probably be completed by April 12, as planned.

Action items



Decisions

 Order Flex

2019-04-17 Meeting notes

Date

17 Apr 2019

Participants

- Erica Fegri
- Rita Hogstad
- Thomas Mundal
- Ming Kit Wong
- Jon Skjelsbek
- Håvar Østrem
- Eirik
- Lars Helge

Discussion topics

Item	Notes
Verification of M3	<ul style="list-style-type: none"> • M3 screws can be a little small. Should be reassessed • Optimize the structure of M2 for 3D printing in metal and get a new price from Tronrud. Way for and against 3D printing in documentation. • We can drop the center ball bearing to reduce costs and use the wedge track of the gear to prevent movement in the axial direction. • Use 1mm alloy plates to wall
Update from electro	<ul style="list-style-type: none"> • Use the EC 2216 Resin as an adhesive in the documentation, but not necessary in the prototype. • Multiturn sensor ordered • Waiting for closet
Update from data	<ul style="list-style-type: none"> • Told, among other things, the rig for testing torsion • Was recommended to make graphs

2019-04-18 Meeting notes

Date

18 Apr 2019

Participants

- Jose
- Rita
- Erica
- Thomas
- Jon
- Ming Kit (author)

Discussion topics

Item	Notes
------	-------

Report outline	<ul style="list-style-type: none"> • Jose asked if we have got any requirements from Karoline about the report. There is no requirement given from Karoline, only that it shouldn't be too long and that all the work should be documented. <p>Following is a suggestion of the report outline. The bold text is level 1</p> <ul style="list-style-type: none"> • Introduction • The introduction should present the work in a short form and how the document is organized. • The last part of the intro should be about the content of the chapters • State of the art • The project it self • Background • What is expected of us • Organization • Requirement (handling?) in a readable way • Verification (handling?) • process model • process plan • Risk • the work • Challenges • How we implement the project setup • Different versions • the results • Final prototype - Detailed description • Final test environment • the conclusion • Summery • reflections • self assesment • what remains/ future work
Quotes/decisions made	<ul style="list-style-type: none"> • Good idea to tell about every versions, but have focuse on the last one. One page per version is good. • What is achievement? If that is what you have learned and what you achieved, it should be in the Conclusion. • Name of the chapters should be easy to understand and describe the content of the chapter. "Verification of process and technical" can be misunderstood

2019-04-24 Meeting notes

Date

24 Apr 2019

Participants

- Rita Hogstad
- Erica Fegri
- Thomas Mundal
- Ming Kit Wong
- Håvar Østrem
- Jon Skjelsbek
- Eirik
- Lars Helge
- Carsten

Goals

- Short status update from machine, electrical and data
- Demonstration of software layer
- If time: a little about the plan for Space Night

Discussion topics

Item	Presenter	Notes
Short status update from machine, electrical and data		<ul style="list-style-type: none"> • If mek.lab provides ready signal, design is complete (lacking analysis). • 2 models, 1 for print and one for mek.lab. • Worth checking out 3D printing of the structure for M2 for 25,000 kr. • Status flex: Waiting for subcontractor. Can go before the last presentation, but not for sure. • Run analyzes on fit of ball bearings in relation to temperature fluctuations. • Test ball bearings at different temperatures. Typically behaves differently when it is cold..

Demonstration of software layer	Thomas Mundal	<ul style="list-style-type: none">• Sending provides ratio from software.• Save both handled data and raw data.• Have information such as gives ratio together with raw data.• Copy selected steps in the test instead of / in addition to storing the entire test.• When the same test runs several times, it may be good to compare test results to see if it behaves equally every time.• Run power up and down and check temperature.
---------------------------------	---------------	--



2019-05-08 Meeting Notes

Date

08 May 2019

Participants

- Erica Fegri
- Rita Hogstad
- Ming Kit Wong
- Jon Skjelsbek
- Håvar Østrem
- Eirik
- Lars Helge
- Carsten

Discussion topics

Item	Presenter	Notes
Analysis	Ming Kit and Rita	<ul style="list-style-type: none"> • Analysis should start at 23 degrees • Justify all choices regardless of whether they claim to be wrong and document steady state • Discard suggestions on rubber socks around the ball bearing on the basis of alignment • Do vibration tests on only the main shaft, not the entire drivetrain • Perform sinus analysis first - it is most important
Testing of electronics	Erica and Håvar	<ul style="list-style-type: none"> • We aim to have flex by 20.05.19. • We contact Lars Helge for test equipment and possibly access to test lab. • Mechlab has glue • Testing of the flex itself will be prioritized, but testing of flex that is fully implemented is desirable.
Final presentation + EXPO		<p>We bet on "lunch & learn" Monday 03.06 at about 11:30-12:00</p> <ul style="list-style-type: none"> • EXPO - look at the Copy Center for any equipment

Action items

- Eirik examines opportunities for lunch & learn Monday 03.06.19

**Avtale om elektronisk publisering av materiale via
USN Open Archive ved Universitetet i Sørøst-Norge**

Mellom Universitetet i Sørøst-Norge

og

forfatter(e): Jon Skjelsbæk, Rita Henriksen Hogstad, Ming Kit Wong, Håvar Østrem,

Thomas Mundal og Erica Fegri

(nedenfor kalt forfatteren)

er det inngått avtale om å tilgjengeliggjøre forfatterens verk,

tittel: SatLight

på de vilkår som er angitt nedenfor.

Fylles ut for studentoppgaver	<input type="checkbox"/> Masteroppgave	<input checked="" type="checkbox"/> Bacheloroppgave	År: <u>2019</u>
Tilgjengelighet:	<input checked="" type="checkbox"/> Åpen	<input type="checkbox"/> Klausulert til (dd/mm/åååå): _____	<input type="checkbox"/> Klausulert permanent/ingen publisering
Fakultet/studium: <u>Faculty of Technology, Natural Sciences and Maritim Sciences</u>			
Privat e-postadresse: <u>ritahogstad@gmail.com</u>			

Kongsberg 22.05.19

Sted, dato

Thomas Mundal Håvar Østrem
Underskrift forfatter

Underskrift USN

Rita H. Hogstad Jon Skjelsbæk
Underskrift forfatter

Ming Kit Wong Erica Fegri
Underskrift forfatter

1 Tillatelse til å publisere elektronisk materiale

1.1 Forfatteren gir herved USN en vederlagsfri, ikke-eksklusiv rett til å gjøre innlevert elektronisk materiale, nedenfor kalt materialet, tilgjengelig i elektronisk form via USN Open Archive, som innebærer publisering på Internett.

1.2 Forfatteren har satt seg inn i, forstår og aksepterer de konsekvenser en publisering via Internett medfører. Blant annet innebærer en slik tilgjengeliggjøring at andre nettsteder kan lenke til materialet. Hvis forfatteren har planer om å publisere materialet ved forlag eller i tidsskrift må forfatteren være klar over at dette kan ha konsekvenser når materialet også tilgjengeliggjøres i USN Open Archive. Se pkt. 3.3.