Sensur av hovedoppgaver

Høgskolen i Sørøst-Norge
Fakultet for teknologi og maritime fag

Prosjektnummer: **2016-09**
For studieåret: **2015/2016**
Emnekode: **SFHO3201**

Prosjektnavn:
K-Spider

Utført i samarbeid med: Kongsberg Maritime AS
Ekstern veileder: Qui-Huu Le-Viet

Sammendrag:

Prosjektoppgaven som er gitt er å lage en Autonom mobil sensorplattform som kan kontrolleres via Kongsberg Maritime sitt sanntidskontroller, RCU510. I fremtiden skal den mobile plattformen kunne hente sensor data og operere i ulendt og farlig terreng. Fokuset er på kommunikasjon mellom RCU510 og sensorplattformen. Resultatet ble et kommunikasjonsnettverk mellom Kongsberg Maritime sitt sanntidskontroller, RCU510 og sensorplattformen. Systemet er laget slik at sensorplattformen kan enkelt byttes ut så lenge den nye plattformen får tilpasset ett serielt Interface.

Stikkord:
- Mekatronikk
- Signalbehandling
- Programmering

Tilgjengelig: JA

Prosjekt deltagere og karakter:

Navn	Karakter
Aleksander Tokle Poverud	
Masoud Shah Pasand	
Tor Gunnar Finnerud	
Anna Kåsastul	
Paul Knutson Sæther	
Abdurahman Senkaya	

Dato: 9. Juni 2016

Jørn Breivoll	Karoline Moholth	Merethe Gotaas
Intern Veileder	Intern Sensor	Ekstern Sensor

INNHOLDSFORTEGNELSE

- PROSJEKTRAPPORT
- VISJONSDOKUMENT
- PROSJEKTPLAN
- ITERASJONSDOKUMENT
- KRAVSPESIFIKASJON
- KRAVENDRINGSHÅNDTERING
- DESIGNDOKUMENT
- TEKNOLOGIDOKUMENT
- TESTPLAN
- TESTRAPPORT
- ØKONOMIDOKUMENT
- BRUKERMANUAL
- HMS (HELSE, MILJØ OG SIKKERTHET)
- ETTERANALYSE
- VEDLEGG

Høgskolen i Sørøst-Norge

KONGSBERG

PROSJEKTRAPPORT

PROSJEKT	K-Spider		
OPDRAGSGIVER	Kongsberg Maritime AS		
UTFØRT VED	Høgskolen i Sørøst-Norge, avd. Kongsberg		
REVISJON	1.0		
MEDLEMMER	Aleksander Tokle Poverud, Masoud Shah Pasand, Tor Gunnar Finnerud, Hanna Kåsastul, Paul Knutson Sæther, Abdurahman Senkaya		
Dokumenthistorikk	**REVISJON**	**UTGITT**	**BESKRIVELSE**
	1.0	20.05.2016	Første utgave

INNHOLDSFORTEGNELSE

1 DOKUMENTHISTORIE 3
2 FORORD 3
3 SAMMENDRAG 4
4 DOKUMENTOVERSIKT 5

TABELLER

Tabell 1 - Dokumenthistorie 3

FIGURER

Figur 1 - Dokumentoversikt 5

1 DOKUMENTHISTORIE

Tabell 1 - Dokumenthistorie

VERSJON NR:	DATO ENDRET	ENDRET AV	GODKJENT AV	BESKRIVELSE
0.1	16.05.2016	Masoud	Aleksander	• Dokumentet ble opprettet
1.0	20.05.2016	Masoud	Aleksander	• Godkjent og utgitt

2 FORORD

Som en avsluttende del av bachelorstudiet innen Kybernetikk og mekatronikk ved høgskolen i Sørøst-Norge, avdeling Kongsberg, har prosjektgruppa gjennomført et omfattende hovedprosjekt for Kongsberg Maritime AS.

Prosjektgruppen har bestått av fem elektro- og en datastudent som har gjennomført oppgaven i perioden januar 2016 til begynnelsen av juni 2016. Prosjektgruppen hadde allerede i desember 2015 gjort en forstudie av prosjektet, satt sammen medlemmene i gruppen og holdt møter med oppdragsgiveren. Tiden etter nyttår har bestått av å utrede teoretiske løsninger, lage en Autonom mobil sensorplattform, programmere og bygge en robot som kan utføre grunnleggende bevegelseskommandoer operert fra Kongsberg Maritime sitt kontrollsystem.

Resultatet av denne oppgaven skal vise at Kongsberg Maritime sitt kontrollsystem kan anvendes i andre bruksområder enn slik det er brukt til per i dag. I fremtiden skal den mobile plattformen kunne hente sensor data og operere i ulendt og farlig terreng. Fokuset er på kommunikasjon mellom Kongsberg Maritime sitt sanntidskontroller RCU510 og roboten.

Prosjektgruppa vil gjerne benytte anledningen til å takke alle de involverte fra Kongsberg Maritime og Høgskolen i Sørøst-Norge for den hjelpen de har bidratt med. Gruppa ønsker å rette en spesiell takk til veilederne våre: Jørn Breivoll og Qui-Huu Le-Viet som har bidratt under hele prosjektperioden.

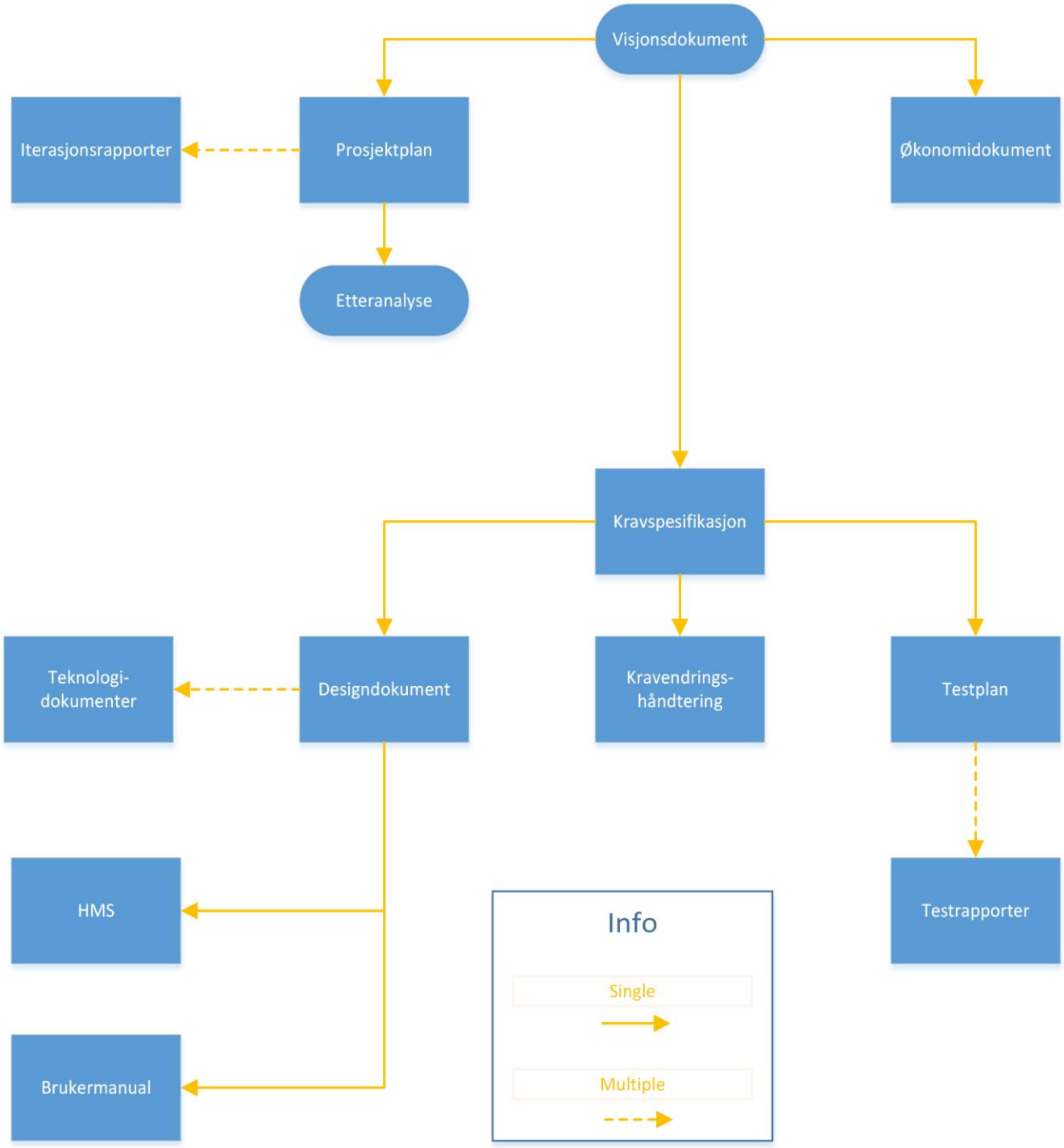
3 SAMMENDRAG

Oppdragsgiver er Kongsberg Maritime AS.

Prosjektoppgaven som ble gitt var å lage en Autonom mobil sensorsplattform som kan kontrolleres via Kongsberg Maritime sitt kontrollsystem, RCU510. I fremtiden skal den mobile plattformen kunne hente sensor data og operere i ulendt og farlig terreng. Fokuset er på kommunikasjon mellom RCU510 og sensorplattformen. Oppgaven er løst ved å ha gjennomført et utviklingsprosjekt med seks medlemmer. For å realisere systemet er det benyttet Raspberry Pi, Arduino, Hexapod 3DOF og Kongsberg Maritime sitt programmeringsverktøy AIM2000. Oppgaven har gitt prosjektgruppa økt kompetanse innenfor prosjektarbeid, samt bruk av kunnskapene vi har tilegnet oss gjennom våre studier, og testet de ut i praksis. Resultatet ble et kommunikasjonsnettverk mellom Kongsberg Maritime sitt sanntidskontroller, RCU510 og sensorplattformen. Systemet er laget slik at sensorplattformen kan enkelt byttes ut så lenge den nye plattformen får tilpasset ett serielt Interface.

4 DOKUMENTOVERSIKT

Figur 1 viser hvordan prosjektrapporten er bygd opp, og hvilke dokumenter som er utarbeidet gjennom prosjektperioden.



Figur 1 - Dokumentoversikt

VISJONSDOKUMENT

PROSJEKT	K-Spider		
OPDRAGSGIVER	Kongsberg Maritime AS		
UTFØRT VED	Høgskolen i Sørøst-Norge, avd. Kongsberg		
REVISJON	3.0		
MEDLEMMER	Aleksander Tokle Poverud, Masoud Shah Pasand, Tor Gunnar Finnerud, Hanna Kåsastul, Paul Knutson Sæther, Abdurahman Senkaya		
Dokumenthistorikk	**REVISJON**	**UTGITT**	**BESKRIVELSE**
	1.0	29.01.2016	Første utgave
	2.0	08.03.2016	Andre utgave
	3.0	20.05.2016	Tredje utgave

INNHOLDSFORTEGNELSE

1 DOKUMENTHISTORIE 3
2 INNLEDNING 3
3 ORGANISERING 4
3.1 Ansvarsfordeling og oversikt 4
3.2 Oppdragsgiver 6
3.2.1 Bakgrunnshistorie for Kongsberg Maritime 6
3.2.2 Dagens Kongsberg Maritime 6
3.3 Veiledere og sensorer 7
3.3.1 Andre interessenter 7
4 OPPGAVEN 8
4.1 Oppgavebeskrivelse 8
4.2 Detaljert oppgavebeskrivelse 9
4.3 Systemarkitektur 10
5 OPPGAVENS MÅL OG FORVENTNING 11
5.1 Prosjektgruppens mål 11
5.2 Oppdragsgivers mål 11
6 PROSJEKTMODELL 12
6.1 Oppbygningen av prosjektmodellen 12
7 TIDSESTIMERING 12
8 REFERANSER 13

TABELLER

Tabell 1 - Dokumenthistorie 3
Tabell 2 - Gruppemedlemmer 4
Tabell 3 - Veiledere og sensorer 7
Tabell 4 - Andre ressurspersoner 7

FIGURER

Figur 1 - Systemoppbygging 10

1 DOKUMENTHISTORIE

Tabell 1 - Dokumenthistorie

VERSJON NR:	DATO ENDRET	ENDRET AV	GODKJENT AV	BESKRIVELSE
0.1	15.01.2016	Masoud	Aleksander	• Dokumentet ble opprettet
1.0	21.01.2016	Masoud	Aleksander	• Endret forside og oppdatert innholdet
1.1	09.02.2016	Aleksander	Masoud	• Byttet økonomi og presentasjonsansvarlige • Lagt til ressursperson
2.0	08.03.2016	Masoud	Aleksander	• Endret bildekvalitet for gruppemedlemmene • Lagt til ressursperson
2.1	10.05.2016	Masoud	Aleksander	• Oppdatert innholdet i kapittel 7 • Lagt til ressurspersoner i del delkapittel 3.3.1
2.2	11.05.2016	Masoud	Paul	• Oppdatert figur 1 (Systemoppbygning) • Endret noe innhold i kap. 4.1 oppgavebeskrivelsen.
3.0	20.05.2016	Masoud	Aleksander	• Godkjent og utgitt

2 INNLEDNING

Visjonsdokumentet er det første og grunnleggende dokumentet som er gjort i forbindelse med oppstarten av det avsluttende hovedprosjektet ved Høgskolen i Sørøst-Norge, avd. Kongsberg. Hensikten med dokumentet er å gi prosjektgruppen en felles forståelse i valg av prosjektmodell, en tydelig forståelse av oppgaven, og en grov tidsestimering av prosjektgjennomføring. Visjonen er ikke en endelig plan som prosjektgruppa vil følge, men det skal sørge for at prosjektgruppen skal jobbe videre med overordnede mål og rammer for prosjektet. Oppgaven er gitt av oppdragsgiver, Kongsberg Maritime og prosjektgruppen skal følge dens mål og forventninger for å gjennomføre oppgaven. Endringer som kommer under prosjektet vil bli ført opp under dokumenthistorikken.

3 ORGANISERING

Dette kapittelet gir en introduksjon av medlemmene i prosjektgruppen og Kongsberg Maritime som er vår oppdragsgiver. Prosjektgruppa består av fem elektroingeniør studenter med spesialisering i '' Kybernetikk og mekatronikk'' og en dataingeniør student.

3.1 Ansvarsfordeling og oversikt

Tabell 2 - Gruppemedlemmer

PERSONLIA	HOVEDANSVARSOMRÅDER	BILDE
Navn: Aleksander Tokle Poverud **E-post:** alektpov@gmail.com **Tlf:** 919 97 792	• Prosjektleder • Presentasjon	
Navn: Masoud Shah Pasand **E-post:** masoud_sp91@hotmail.com **Tlf:** 481 13 904	• Dokumentansvarlig • Økonomi	
Navn: Tor Gunnar Finnerud **E-post:** torgunnar89@hotmail.com **Tlf:** 412 94 495	• Kravansvarlig • Testansvarlig	

Navn: Hanna Kåsastul **E-post:** hanna.kaasastul@gmail.com **Tlf:** 476 19 984	• Utviklingsansvarlig	
Navn: Paul Knutson Sæther **E-post:** paulks93@gmail.com **Tlf:** 913 22 612	• Programmering • Web	
Navn: Abdurahman Senkaya **E-post:** a-senkaya@hotmail.com **Tlf:** 930 10 366	• Design & Konstruksjon	

Hver enkelt gruppemedlem er blitt tildelt ulike ansvarsområder. Dette er for å holde best mulig kontroll over de ulike temaene vi skal gjennomføre i løpet av dette prosjektet. Gruppemedlemmene er eneansvarlige for sitt hovedområdet, men det er ikke meningen at de områdene skal dekkes alene, men heller at den ansvarlige personene står med oversikten innenfor de aktuelle området. Vi ser for oss at dette vil være en ryddig og strukturert måte å føre prosjektet på. Det vil også være et mål for gruppen at alle medlemmene bidrar på hvert enkelt felt om det er ledige ressurser.

3.2 Oppdragsgiver

Oppdragsgiveren er Kongsberg Maritime AS. KM[1] er en veletablert teknologibedrift i Kongsberg Gruppen ASA. De leverer hovedsakelig utstyr til offshore og maritim industrien, men også systemer for posisjonering, overvåking, navigasjon og automasjon til handelsflåte og offshore installasjoner. De er kjent for dynamisk posisjoneringssystemer og viktige målgrupper er land med stor offshore og verftsindustri.

3.2.1 Bakgrunnshistorie for Kongsberg Maritime

Kongsberg Maritime ble etablert 6. februar 1998. Bedriften stammer fra sammenslåingen av de tidligere bedriftene Kongsberg Simrad (tidl. Simrad Albatross), Kongsberg Maritime Ship Systems og Simrad. Samme år endret selskapet navnet til Kongsberg Gruppen og kjøpte Simrad konsernet. Flere oppkjøp er gjort etter denne perioden, og dette har resultert i at KM er en av verdens største leverandører av offshore- og maritime systemer. [1]

3.2.2 Dagens Kongsberg Maritime

Kongsberg Maritime består av syv divisjoner – Global salg & markedsføring, Offshore, Merchant Marine, Subsea, Emerging business, Global kundestøtte og næringsutvikling. KM har 55 kontorer i 18 land, hvorav hovedkontoret ligger i Kongsberg. Totalt har KM 4159 ansatte (pr 31.12.2012). [2]

[1] KM er forkortelse på Kongsberg Maritime.

3.3 Veiledere og sensorer

Under prosjektet så har prosjektgruppen to sensorer og to veiledere, hvorav én av hver er interne og eksterne. Disse kommer til å følge oss under prosjektet, og være til stede på presentasjoner.

Tabell 3 - Veiledere og sensorer

PERSONLIA	FUNKSJON	KONTAKTINFORMASJON
Jørn Breivoll	Intern veileder Høgskolen i Sørøst-Norge	**E-post:** Jorn.Breivoll@hbv.no **Tlf:** 31008903
Karoline Moholth	Intern Sensor Høgskolen i Sørøst-Norge	**E-post:** Karoline.Moholth@hbv.no **Tlf:** 31008898
Qui-Huu Le-Viet	Ekstern veileder Kongsberg Maritime	**E-post:** Qui-Huu.Le-Viet@km.kongsberg.com **Tlf:** 95885196
Merethe Gotaas	Ekstern sensor Kongsberg Maritime	**E-post:** Merethe.Gotaas@km.kongsberg.com **Tlf:**

3.3.1 Andre interessenter

Andre relevante personer som vi har brukt under prosjektet er listet opp i tabell 4. Disse ressurspersoner er med på å hjelpe gruppa gjennom prosjektet.

Tabell 4 - Andre ressurspersoner

PERSONLIA	RESSURSOMRÅDE	KONTAKTINFORMASJON
Hans Ivar Brekke	AIM/RCU	**E-post:** Hans.ivar.brekke@km.kongsberg.com **Tlf:** 958 85 558
Anders Viken	AIM	**E-post:** anders.viken@km.kongsberg.com **Tlf:** 958 30 809
Joakim Bjørk	Låne av utstyr	**E-post:** Joakim.Bjork@hbv.no **Tlf:** 951 54 961

4 OPPGAVEN

Dette prosjektet er den avsluttende oppgaven på bachelorstudiet ved Høgskolen i Sørøst-Norge, avd. Kongsberg. Oppgaven er gitt av Kongsberg Maritime.

4.1 Oppgavebeskrivelse

Oppgaven går ut på lage en Autonom mobil sensorplattform som kan kontrolleres via KM sitt kontrollsystem. I fremtiden skal den mobile plattformen kunne hente sensor data og operere i ulendt og farlig terreng. Fokuset er på kommunikasjon mellom RCU og sensorplattformen. Visjonen består av fire faser, der gruppa har hovedfokus i fase 1.

Fase 1 (2016)

Mål: Bygge en robot som kan utføre grunnleggende bevegelseskommandoer operert fra KM kontrollsystem.

Krav:

- Mekanisk konstruksjon av en robot.
- Finne en kommunikasjonsprotokoll.
- Implementere den anbefalte kommunikasjonsprotokollen.
- Kontroll kommandoer fra AIM skal gjennomføres ved hjelp av sekvens og logiske moduler.
- Roboten skal kunne utføre sine bevegelser i henhold til kommandoer gitt fra KM kontrollsystem.
- Roboten skal også kunne kontrolleres direkte via ekstern kontroller

Fase 2 (2017)

Mål: Legge til oppdrag tilpassede sensorer

Krav: Skal defineres

Fase 3 (2018)

Mål: Behandling av sensordata og bruke denne informasjonen til å lukke kommandokontrollsløyfen.

Krav: Skal defineres

Fase 4 (2019):

Mål: Implementering av autonome egenskaper til roboten.

Krav: Skal defineres

Det ferdige produktet vil være Kongsberg Maritime AS sin eiendom og alle deler dekkes av bedriften.

4.2 Detaljert oppgavebeskrivelse

Oppgaven skal bestå av en rapport og en fysisk utførelse.

Følgende punkter ønskes gjennomført:

- Finne løsninger, bestille komponenter og bygge systemet
- Få servo komponentene til å samarbeide slik at koordinerte bevegelser av plattformen blir mulig
- Roboten skal kunne manuelt fjernstyres via ekstern kontroller
- Undersøke hvilke kommunikasjonsprotokoller som egner seg best for data overføring mellom AIM og roboten
- Implementere den valgte kommunikasjonsprotokollen
- Konfigurasjon av kontroll kommandoene i AIM
- Roboten skal kunne kontrolleres fra KM kontroller
- Mer avansert robot bevegelser
- Testing
- Presentasjon av prosjektet med demonstrasjon

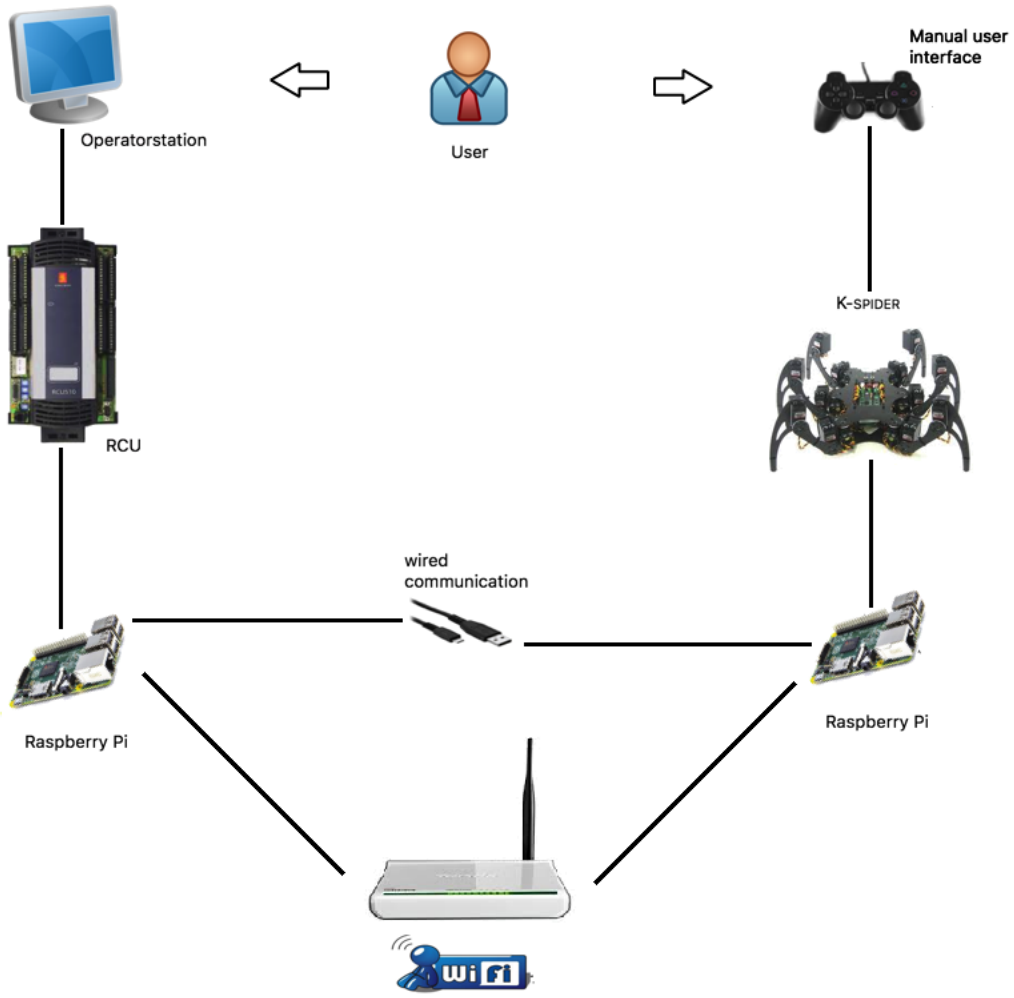
Valgfri hvis det blir tid:

- Trådløs kommunikasjon
- Mer avansert robot bevegelse
- Feedback sensorer
- Mission sensorer

I oppgaven skal det benyttes KM kontrollsystem AIM7 for styring. Operatørstasjon med tilhørende sanntidskontroller RCU510 skal installeres.

4.3 Systemarkitektur

Etter første møtet med KM ble det fremlagt flere forslag på hvordan de ønsker å bygge opp arkitekturen rundt den fysiske utførelsen av oppgaven. Figur 1 viser et eksempel på hvordan oppgaven kan løses. NB! Figur 1 illustrerer det som er oppgitt i oppgaveteksten i kapittel 4.1 og er det første utkastet og ikke den endelige løsningen.



Figur 1 - Systemoppbygging

5 OPPGAVENS MÅL OG FORVENTNING

Prosjektgruppa og oppdragsgiver har flere mål og forventinger gjennom arbeidet med dette prosjektet. Det er valgt få, men konkrete mål, som gjør det lettere for oss å holde fokus på de gitte rammene, slik at prosjektet blir en suksess.

5.1 Prosjektgruppens mål

- Levere et sluttprodukt som tilfredsstiller kundens krav og spesifikasjoner
- Oppnå en tilfredsstillende sluttkarakter som alle i gruppa er fornøyd med
- Økt kompetanse innenfor prosjektarbeid
- Utfordre de kunnskapene vi har tilegnet oss gjennom våre studier, og teste de ut i praksis

5.2 Oppdragsgivers mål

- Kongsberg Maritime har et ønske å undersøke om deres kontrollsystem kan anvendes i andre bruksområder enn slik det er brukt til per i dag.

6 PROSJEKTMODELL

Etter oppstarts undervisning (07.01.2016) av Olaf Hallan Graven gjorde vi research på forskjellige modeller. Modellen valgte vi i fellesskap og er siktet til å lage ett effektivt og målrettet arbeidsmiljø. Som prosjektplanen er modellen ett verktøy for prosjektleder til å både kvalitetssikre arbeid og at innleveringer skjer innen tidsfrister.

Modellen til prosjektet er en viktig del av prosessen til ett komplett produkt. Den avgjør hvordan gruppen skal arbeide, måten det organiseres på og hvordan gruppen skal utvikle seg framover ved å forholde seg til forskjellige faser i prosjektet.

6.1 Oppbygningen av prosjektmodellen

Unified Process er ikke bare en prosess, men en større arbeidsmetodikk som blir tilpasset spesifikt til prosjektet. UP er en inkrementell iterativ metodikk som tillater at vi selv tilpasser den slik vi trenger, som vil gjøre det lettere å tilpasse til utviklingen av vårt prosjekt. Mye av arbeidet som kommer til å bli gjort vil bli gjort i parallell forløpende og integreres.

Gruppen valgte UP for å kunne ha muligheten til å endre på hvilket system som integreres ettersom en del kan være seinere enn ett annet i utviklingsfasen, eller når andre uforutsette problemer oppstår. For ytterligere utredelser av prosjektplanen/prosjektmodellen, se kapittel 8 i dokumentet *Prosjektplan*.

7 TIDSESTIMERING

Dette hovedprosjektet er det avsluttende faget på bachelorstudiet i ingeniørfag ved HSN (Høgskolen i Sørøst-Norge) avdeling Kongsberg og omfatter 20 studiepoeng. Det er vanlig å beregne ca. 30 arbeidstimer pr. studiepoeng og for dette prosjektet er det estimert ca. 600 timer pr. student. Gruppen har et mål å jobbe ca. 28t i uka fra uke 1 til 14, men for å nå frister skal arbeidstiden utvides etter at skriftlig eksamen i Reguleringsteknikk 2 er ferdig i uke 15. For å holde god oversikt over hvilke oppgaver og antall timer som blir jobbet med til enhver tid er det laget en oppfølgningsdokument som føres hver dag og sendes ukentlig til intern veileder. Oppfølgningsdokumentet inneholder også tim elister for hver enkelt student og beskrivelser av aktiviteter som hver enkelt gruppemedlem skal jobbe med.

8 REFERANSER

[1] Wikipedia, 21.01.2016, https://no.wikipedia.org/wiki/Kongsberg_Maritime

[2] Kongsberg Maritimes hjemmeside, 21.01.2016, http://www.km.kongsberg.com/ks/web/nokbg0237.nsf/AllWeb/0B29D39758C415B0C1256DF700560834?OpenDocument

PROSJEKTPLAN

PROSJEKT	K-Spider		
OPDRAGSGIVER	Kongsberg Maritime AS		
UTFØRT VED	Høgskolen i Sørøst-Norge, avd. Kongsberg		
REVISJON	3.0		
MEDLEMMER	Aleksander Tokle Poverud, Masoud Shah Pasand, Tor Gunnar Finnerud, Hanna Kåsastul, Paul Knutson Sæther, Abdurahman Senkaya		
DOKUMENTHISTORIE	**REVISJON**	**UTGITT**	**BESKRIVELSE**
	1.0	02.02.2016	Første utgave
	2.0	03.03.2016	Andre utgave
	3.0	20.05.2016	Tredje utgave

INNHOLDSFORTEGNELSE

1 DOKUMENTHISTORIE 6

2 INNLEDNING 8

3 MÅL OG RAMMER 9

3.1 Bakgrunn 9

3.2 Prosjektmål 10

3.2.1 Prosjektgruppens mål 10

3.2.2 Oppdragivers mål 10

3.3 Prosjektramme 10

4 OPPFØLGING 11

4.1 Oppfølgingsdokument 11

4.2 Timelister 11

4.3 Møtestruktur 12

4.3.1 Interne gruppemøter 12

4.3.2 Møte med intern veileder 12

4.3.3 Møte med ekstern veileder 12

4.4 Hjemmeside 12

5 ORGANISERING 13

6 DOKUMENTASJON 14

6.1 Dokumenthistorie 14

7 PLANLEGNING OG GJENNOMFØRING 15

7.1 Overordnet tidsplan 15

8 RISKOANALYSE 18

8.1 Meningen med dokumentet 18

8.2 Forkortelser 18

8.3 Risikohåndtering 19

8.3.1 Største risikoene 19

8.3.2 Hensikten med en risikoanalyse 19

8.4 Risiko ved gjennomføring av prosjektet 20

8.4.1 Utilstrekkelig opplæring i AIM 20
8.4.2 Hardware tilgang 20
8.4.3 Menneskelig faktorer 21
8.4.4 Dårlig arbeid som gruppe 21
8.4.5 Defekte komponenter 22
8.4.6 Tekniske problemer 22
8.5 Systemrisiko 23
8.5.1 Klarer ikke å få RCU til å kommuniserer med robot 23
8.5.2 Kommunikasjon mellom RCU og RP svikt 23
8.5.3 Brente eller ødelagte komponenter 24
8.6 RISIKOMATRISE 24
9 PROSJEKTMODELL 27
9.1 INNLEDNING 27
9.2 MODELLEN 27
9.2.1 Unified Process 27
9.2.2 Hvordan vi skal jobbe 28
9.2.3 Planen 28
9.2.4 Fasene 29
9.2.5 Skreddersydd prosjektmodell 31
9.2.7 Inception 31
10 REFERANSER 33

TABELLER

Tabell 1 - Vedlegg 5

Tabell 2 - Dokumenthistorie 6

Tabell 3 - Ansvarsbeskrivelse 13

Tabell 4 - Milepæler 15

Tabell 5 Overordnet tidsplan 16

Tabell 6 - Forkortelse 18

Tabell 7 - AIM Opplæring 20

Tabell 8 - Hardware tilgang 20

Tabell 9 Menneskelig faktor 21

Tabell 10 - Arbeid som gruppe 21

Tabell 11 - Defekte komponenter 22

Tabell 12 - Tekniske problemer 22

Tabell 13 - RCU kommunikasjon 23

Tabell 14 - Kommunikasjon svikt 23

Tabell 15 - Komponenter 24

Tabell 16 - Riskomatrise 24

Tabell 17 - Forklaring av fargekode 25

Tabell 18 - Risikoanalyse 25

Vedlegg

Tabell 1 - Vedlegg

Vedlegg	Beskrivelse
1	Gantdiagram
2	Aktivitetsplan Inception
3	Aktivitetsplan Inception 2
4	Aktivitetsplan Elaboration
5	Aktivitetsplan Elabortation 2
6	Aktivitetsplan Construction 1
7	Aktivitetsplan Construction 2
8	Aktivitetsplan Construction 3
9	Aktivitetsplan Transition 1
10	Aktivitetsplan Transition 2

1 DOKUMENTHISTORIE

Tabell 2 - Dokumenthistorie

VERSJON NR:	DATO ENDRET	Endret av:	Godkjent av:	BESKRIVELSE
1	02.02.2016	Aleks Paul Abdu	Paul&Aleks	• Skreddersydd prosjektmodell lagt til • Innholdoppdatert • Vedleggsliste lagt til • Alle under-tittler gjennomgått/Endret • Ganttdiagram lagt til
1.1	10.02.2016	Aleksander	Masoud	• Lagt til Elaboration aktivitetsplan
1.2	17.02.2016	Aleksander	Masoud	• Iterering lagt til
1.3	25.02.2016	Masoud	Aleksander	• Formatering og oppdatering av vedlegg 3 og 4.
2.0	03.03.2016	Masoud	Aleksander	• Formatering av hele dokumentet • Oppdatering av vedlegg 5
2.1	06.04.2016	Aleksander	Masoud	• Lagt til Aktivitetsliste vedlegg • Endring i overordnet tidsplan, faser forandret. Construction 2 og utover.

2.2	13.04.2016	Aleksander	Masoud	• Utfylt vedlegg 7 (Construction 2 fasen) • Oppdatert Construction 3
2.3	27.04.2016	Masoud	Aleksander	• Utfylt timene i vedlegg 8 (Construction 3). • Lagt inn Transition 1 fasen som vedlegg 9 med aktiviteter.
2.4-2.13	10.05.2016	Aleksander	Masoud	• Oppdatert linken til hjemmesiden i kapittel 4.4 • Fortløpende oppdatering av aktivitetslister per uke
2.14	11.05.2016	Aleksander	Masoud	• Lagt til Transition 2 aktiviteter
2.15	12.05.2016	Masoud	Aleksander	• Utfylt aktivitetstimene i Transition 1 (Vedlegg 9).
2.16	19.05.2016	Masoud	Aleksander	• Utfylt aktivitetstimene i Transition 2 (Vedlegg 10).
3.0	20.05.2016	Masoud	Aleksander	• Godkjent og utgitt

2 INNLEDNING

Prosjektplan skal gi prosjektgruppa oversikt over når og hvordan mål skal oppnås, ved å fastsette de viktigste milepælene i prosjektet og gi klare rammer for å tilfredsstille kravene fra oppdragsgiver. Prosjektplanen er et viktig verktøy for å holde oversikt over fremdriften i prosjektet. Planen skal dekke aktivitetene, fremdriften, rollene og prosessen gruppen skal gå gjennom i prosjektperioden for å oppnådd gode resultater.

Dokumentet vil også åpne mulighet for oppdragsgiver til å følge prosjektets gjennomføring og ha formening om sluttprodukt dato. I tillegg vil det være mulig å følge opp kostnader og tidsbruk i forhold til prosjektgjennomføringen, risikovurdering og tiltak ved avvik fra tidsestimat skal også komme med i denne planen.

Prosjektplanen har en oversikt over de forskjellige ansvarsområdene til medlemmene i gruppen, samt en ansvarsliste over arbeidsoppgaver og tidsestimat for disse aktivitetene. Aktivitetene er knyttet mot en overordnet tidsplan som følges gjennom prosjektet, det er denne som legger utgangspunktet for framdriften i prosjektet. Det er også beskrevet i prosjektmodellen hvordan gruppen tenker å jobbe seg framover i faser. Med en solid prosjektplan vil gruppen ligge et steg foran, slik at utsettelser av produktet blir unngått, som igjen vil hindre store økonomiske tap.

3 MÅL OG RAMMER

3.1 Bakgrunn

Som en del av det treårige bachelorstudiet ved Høgskolen i Sørøst-Norge skal det være et avsluttende prosjekt som foregår over ett semester. Som del av prosjektoppgaven er det blitt gitt en kort forelesingsperiode med teoretisk innføring i prosjektstyring. Da disse forelesningene var ferdige var det opp til studentene selv å danne grupper, samt oppsøke bedrifter for å skaffe en prosjektoppgave.

Vi hadde allerede dannet gruppe og opprettet kontakt med Kongsberg Maritime, da vi ønsket et spennende hovedprosjekt som kunne utfordre alle gruppemedlemmene, både faglig, praktisk og teoretisk. Siden har det blitt holdt jevnlig kontakt og møter med Kongsberg Maritime v/Qui-Huu Le-Viet. Det ble først introdusert en oppgave fra Kongsberg Maritime, etter litt fram og tilbake byttet vi oppgaven til noe som prosjektgruppen synes var både spennende og utfordrende. Vi bestemte oss dermed for å takke ja til oppgaven.

Gruppen har bestemt at prosjektet skal utføres så profesjonelt som mulig, slik at gruppens medlemmer får mest mulig erfaring, som vi kan ta det videre til arbeidslivet som ingeniører.

3.2 Prosjektmål

3.2.1 Prosjektgruppens mål

- Tilfredsstille kundens krav og spesifikasjoner å levere et godt sluttprodukt.
- Få en tilfredsstillende slutt karakter som prosjektgruppens medlemmer er fornøyd med.
- Anvende teoretiske kunnskapene vi har blitt tilegnet gjennom skolen og teste disse i praksis.
- Øke kompetansen innen prosjektsamarbeid.
- Takle uventede hendelser som kan sette prosjektet tilbake.

3.2.2 Oppdragivers mål

- Kongsberg Maritime har et ønske å undersøke om deres kontrollsystem kan anvendes i andre bruksområder enn slik det er brukt til per i dag.

3.3 Prosjektramme

Prosjektrammen er satt av Høgskolen i Sørøst-Norge og oppdragsgiver Kongsberg Maritime i form av tidsrammer, tidspunkter og kostnader. Kostnader godkjennes og dekkes av oppdragsgiver. Under prosjekt perioden skal det være i alt tre prestasjoner. De to første prestasjonene skal gi innblikk i arbeidet underveis, mens den tredje prestasjonen tar for seg hele prosjektet i sin helhet. Prosjektmedlemmene er ansvarlig for å finne tid og sted og varsle alle veilederne innen god tid. Slutt prosjektet skal leveres inne den 17.mai

4 OPPFØLGING

4.1 Oppfølgingsdokument

Det er opprettet oppfølgingsdokument for å dokumentere og sikre gruppens fremdrift i prosjektet. Oppfølgingsdokumentet inneholder hva hvert gruppemedlem har gjort den siste uka, og hva som skal gjøres den kommende uka. Dokumentet skal inneholde gjøremål, estimert tid, brukt tid og avvik som forekommer. Dokumentet vil også gi veilederne oppdatering på fremdriften i prosjektet.

4.2 Timelister

Hvert gruppemedlem skal innføre timer i timeliste ved slutten av dagen, som igjen blir ført inn i et felles timelistedokument for gruppen ved slutten av uken. Dokumentet er opprettet i Excel og holder orden på antall timer per pers, uke, mnd. og for helegruppen sammenlagt. I tillegg er det laget en aktivitetsplan, dokumentet inneholder hva som er gjort og hvor mange timer som er brukt på hver enkel aktivitet. Dette er for å ha god kontroll på hvordan gruppen ligger an forhold til estimerte timer, og sikre at aktivitetene blir gjennomført.

4.3 Møtestruktur

4.3.1 Interne gruppemøter

Vi har innført 10 minutters «Scrum» møte om morgenen, for å se på hvordan gruppen ligger an forhold til tidsplanen, og hva som må gjøres for å komme i mål. Dette møtet gir oss struktur i arbeidsoppgavene og gir gruppa kontroll på hva som gjenstår for å komme i mål. Det gir oss også en god mulighet til å dele ut nye, eller bytte aktiviteter mellom gruppemedlemmer.

4.3.2 Møte med intern veileder

Hver Fredag klokken 10:00 er det møte med intern veileder, Jørn Breivoll. Prosjektleder sender møteinnkalling til alle møtedeltakere og sender også oppfølgingsdokument vedlagt ukentlige timelister til intern veileder minimum 24 timer før møtet. Prosjektleder er møteleder under disse møtene, og det refereres til oppfølgingsdokumentet, slik at intern veileder får et mer detaljert innblikk i fremdriften i gruppen. Referat skrives etter hvert møte og sendes til intern veileder senest dagen etter.

4.3.3 Møte med ekstern veileder

Møter med ekstern veileder avtales ved behov. Dette kan foregå per telefon, nettmøte eller i forbindelse med opplæring angående Kongsberg Maritime sitt utstyr. Om det skulle være ønskelig og møtes avtales dette underveis.

4.4 Hjemmeside

Fra skolens side er det bestemt at hver prosjektgruppe skal ha en internettside. Hjemmesiden skal fortelle hva prosjektoppgave går ut på, samt hvem prosjektgruppa består av. Siden skal oppdateres regelmessig slik at interesse kan følge fremdriften i prosjektet.

Hjemmeside til prosjektplan: https://home.hbv.no/web-gr9-2016/main.html

5 ORGANISERING

Som en del av oppgaven har alle gruppemedlemmene fått forskjellige ansvarsområder. På denne måten oppnås god oppfølging på alle de forskjellige områdene. Hvert gruppemedlem skal ha en overordnet oversikt over sitt ansvarsområdet, gruppemedlemmene skal også ha lite innblikk på andres områder.

Tabell 3 - Ansvarsbeskrivelse

Ansvarsområdet	Beskrivelse
Prosjektleder	Har ansvaret for prosjektet gjennomføres etter planen. Følger opp gruppen, fordeler oppgaver. Utfører kvalitetssikring.
Økonomiskansvarlig	Har Hovedansvar for den økonomiske delen av prosjektet, og et budsjett som må holdes innenfor rimelige grenser og er ansvarlig for at økonomidokumentet er i rapporten.
Dokumentansvarlig	Har ansvar for sammenstilling av dokumenter, at dokumentene er kvalitetsmessig godt utført, at maler følges og at dokumenthistorien er tilfredsstillende.
Presentasjon- ansvarlig	Ansvarlig for at presentasjon blir ferdigstilt.
Utviklingsansvarlig	Har ansvar for at alt AIM utviklingen blir utført og blir dokumentert.
Programmering og Web	Tar seg av nettsiden og har ansvaret for utførelsen av programmeringen og orden i kode
Design & Konstruksjon	Har ansvar for design og produktets oppbygging
Data/programmering	Har ansvaret for kontrollere utviklingen av Software
Testansvarlig	Har ansvar for testrapportene, samt utførelsen av alle testene.
Kravansvarlig	Er ansvarlig for kravspesifikasjon og kravspesifikasjon dokument

6 DOKUMENTASJON

Det ble det utarbeidet maler som brukes til dokumentasjonen i oppgaven for å forenkle arbeidet. Dette gir en standard på de forskjellige dokumentene.

De ulike malene er:

- Rapport
- Dokument
- Møtereferat
- Oppfølgingsdokumenter
- Møteinnkalling

6.1 Dokumenthistorie

Det vil i hvert dokument være historikk som inneholder utgivelsesdato, hvilken utgave det er, hva som er endret fra forrige versjon, personen som har gjort endringer og person som har godkjent endringene. Vi har nummeret utgavene og revisjonene med tall. Versjon 0.1 er første kladd og endelig og godkjent versjon vil ha 1.0. Tallet før punktum representerer offisielle utgivelser og tallet etter interne representer nye revisjoner; endringer som har skjedd.

7 PLANLEGNING OG GJENNOMFØRING

7.1 Overordnet tidsplan

Tabell 4 og 5 gir en overordnet tidsestimering for prosessen fram mot 17. mai 2016 med satte milepæler da arbeidet aktiviteter skal være ferdige. Eksamensøving periode er kuttet fra tabellen og vist i «Skreddersydd prosjektmodell».

Milepæler i et prosjekt beskriver hva som forventes oppnådd, ikke nødvendigvis hvordan. De er en beskrivelse av en tilstand som beskriver en leveranse som prosjektet bør være kommet til i et visst stadium av prosjektet.

Gjennom milepælene fylles prosjektets mål med innhold. Milepæler må planlegges og utarbeides i et samarbeid mellom flere prosjektdeltakere. Dette vil være med på å sikre framdrift i prosjektet, da de som skal gjennomføre prosjektet også får være med å bestemme og gi innhold til hva som faktisk skal gjøres.

Tabell 4 - Milepæler

Dato	Milepæl
11.01.2016	Start på Inception fasen
05.02.2016	Første presentasjon I første presentasjon skal det legges fram visjonsdokument, kravspesifikasjon, testspesifikasjon og overordnet prosjektplan. Dette vil gi en liten kort introduksjon for de som er tilstede, slik at de for med seg hva som skal produseres og hva oppgaven egentlig går utpå.
10.02.2016	Slutt på Inception fasen
10.02.2016	Start på Elaboration fasen
11.03.2016	Andre prestasjon Den andre prestasjonen vil ha fokus på tekniske løsninger. Det skal presenteres hvor langt prosjektet har kommet og hva som må til for å nå målet.
11.03.2016	Slutt på Elaboration fasen
11.03.2016	Start på Construction fasen

27.04.2016	Slutt på Construction fasen
27.04.2016	Start på Transition fasen
23.05.2016	Hovedrapport innlevering
01.06.2016	Definitivslutt på Transition fasen & Presentasjon 3

Tabell 5 Overordnet tidsplan

Fase	Iterasjon	Start	Estimert ferdig	Ferdig	Viktige gjøremål
Inception	1	11.01	30.01	30.01	• Visjonsdokument • Prosjektplan • Kravspesifikasjon • Testspesifikasjon • Systemkurs (AIM, RCU) • Systemarkitektur • Første del teknologidokument • Bestilling
Inception	2	01.02	10.02	10.02	• Presentasjon • Evaluering
Elaboration	1	10.02	24.02	20.02	• Komplett systemarkitektur • Teknologidokument • Mekanisk konstruksjon • AIM oversikt • Finne kommunikasjonsprotokoll
Elaboration	2	24.02	11.03	11.03	• Ekstern kontroller • Implementere kommunikasjonsprotokoll • Presentasjon
Construction	1	11.03	19.03	20.03	• Begynne å integrere systemet • Kontrollkommandoer ferdigstilt i AIM

///	///	///	///	///	///
Construction	2	30.03	13.04	13.04	• Ferdigstille Modbus kommunikasjon • Implementere Wifi til modbus gjennom RPI
Construction	3	13.04	27.04	27.04	• Testing • Hexapod og Rcu nettverkssammenkobling
Transition	1	27.04	12.05		• Test og implementering • Implementere C og B krav
Transition	2	12.05	23.05		• Levere hovedrapport
Transition	3	24.05	01.06		• Presentasjon 3

8 RISKOANALYSE

8.1 Meningen med dokumentet

En risiko er et kriteria eller en hendelse, som hvis skjer kan påvirke prosjektet på enten en positiv eller mer sannsynlig negativ måte. Risiko håndtering er en prosessen for å evaluere, identifisere og holde øyne med mulige risikoer.

Med dette dokumentet håper vi å dokumentere hvordan vi skal indentifisere, analyserer og håndtere mulige risikoer i prosjektet, og prioritere disse. Dette kan hjelpe oss å bestemme prosjekt fremgangen.

8.2 Forkortelser

Ei liste med forkortelser og til hørende beskrivelser av akronymer som er brukt i dette dokumentet.

Tabell 6 - Forkortelse

Forkortelse	Beskrivelse
KM	Kongsberg Maritime.
RCU	Remote control unit. Possessor fra KM
RP	Raspberry pi, computer component.

8.3 Risikohåndtering

Risiko nivået vil det bli holdt et øyne med igjennom hele prosjektet. Alle forandring som måtte blir gjort i prosjektet ettersom behovet oppstår vil bli analysert for å se hvordan det kan påvirke prosjektet. En liste over de største risikoene i prosjektet vil bli ført.

8.3.1 Største risikoene

- Hardware tilgang
- Dårlig arbeid som gruppe
- Menneskelige faktorer

8.3.2 Hensikten med en risikoanalyse

Hensikten med en risikoanalyse er først å identifisere relevante risikoer i prosjektet. Risikoene vil bli identifisert så tidlig så mulig i utviklingsfasen for å reduserer konsekvensene disse kan ha. Utover dette så er analysen med for å bidra til å utforme en strategi for å fullføre prosjekt målene. Dette delen skal beskrive mulige hovedproblemer som kan bli aktuelle og hvordan vi kan elimi nere eller minimere disse.

Risikoens sjanse for hendelse og konsekvenser rangert fra lav til høy hendelse rate og konsekvens.

- **Lav:** Relativt liten påvirkning på prosjektet (tidsplan, ytelse eller kostnad). Skjer relativt sjeldent.
- **Middels:** Kan potensielt påvirke prosjektet i noe grad. Kan muligens skje fra tid til annen.
- **Høy:** Kan påvirke prosjektet i vesentlig grad. Høy sannsynlighet for å skje.

Mulighetene vi har for å håndtere risikoene kan fordele inn i fire.

- **Unngå:** Finne en løsning å håndtere problemet på slik at det ikke påvirker prosjektet.
- **Dempe påvirkningen(Redusere):** Finne løsninger for å redusere konsekvensene når et problem oppstår.
- **Akseptere:** Aksepterer at det problemet blir det ikke gjort noe med.

8.4 Risiko ved gjennomføring av prosjektet

Risikoer som er assosiert med gjennomføring av prosjektet til en slik grad at kvaliteten av produktet er redusert eller at prosjektet blir umulig å gjennomføre.

8.4.1 Utilstrekkelig opplæring i AIM

AIM er konfigurasjons språket KM bruker til RCU-kontrolleren sin. Siden det ikke finnes info på nett eller lignende kilder, så vil det være vanskelig å tilegne seg denne kunnskapen helt på egenhånd.

Tabell 7 - AIM Opplæring

Sjanse for hendelse	**Lav**	Middels	Høy
Konsekvens	Lav	Middels	**Høy**
Håndtering	**Unngå:** Holde en god dialog med KM. Akseptere: Ellers må vi bare akseptere at de ikke kan avse folk til det.		

8.4.2 Hardware tilgang

Nødvendige deler og utstyr kan bli forsinket siden det er bestilt fra utlandet eller være defekt.

Tabell 8 - Hardware tilgang

Sjanse for hendelse	**Lav**	Middels	Høy
Konsekvens	Lav	Middels	**Høy**
Håndtering	**Redusere:** Legge inn bestillingen i god tid på forhånd. Fra butikker med godt rykte og bruke Express frakt muligheter.		

8.4.3 Menneskelig faktorer

Ting kan skje med individer i gruppen som sjukdom, skader, ulykke r eller andre uforutsette ting.

Tabell 9 Menneskelig faktor

Sjanse for hendelse	Lav	Middels	**Høy**
Konsekvens	Lav	**Middels**	Høy
Håndtering	**Redusere:** Få rikelig med søvn, sunn mat og trene litt. Jobbe som et team og ha innblikk i hverandres arbeid slik at prosjektleder kan koordinere av arbeidsoppgaver til andre gruppemedlemmer ved fravær.		

8.4.4 Dårlig arbeid som gruppe

Fremdriften kan gå betraktelig tregere hvis en eller flere gruppemedlemmer ikke gjør jobben de er satt til eller ikke jobber fort nok. Noe som kan oppstå ved dårlig motivasjon, manglende kommunikasjon eller andre begrensninger eller problemer.

Tabell 10 - Arbeid som gruppe

Sjanse for hendelse	**Lav**	Middels	Høy
Konsekvens	Lav	Middels	**Høy**
Håndtering	**Redusere:** Holde en positiv holdning ved god kommunikasjon. Hjelpe og oppmuntre hverandre så godt som mulig og ha demokrati i gruppa.		

8.4.5 Defekte komponenter

Komponenter som vi har bestilt kan være defekte, og dermed ikke fungerer eller svikter under utvikling.

Tabell 11 - Defekte komponenter

Sjanse for hendelse	Lav	**Middels**	Høy
Konsekvens	Lav	**Middels**	Høy
Håndtering	**Redusere:** Bekrefte at alle delene er gode nok og bestille ekstra reserve deler. Hvis det finnes defekte deler som ikke fins i reserve, bestilles det nye så fort som mulig.		

8.4.6 Tekniske problemer

Menneskelig svikt, komponent feil og programvare krasj er de mest vanlige årsakene til tap av data eller tap av arbeid som allerede er gjort.

Tabell 12 - Tekniske problemer

Sjanse for hendelse	**Lav**	Middels	Høy
Konsekvens	Lav	**Middels**	Høy
Håndtering	**Unngå:** Benytte skylagring (dropbox, googledrive etc.) **Redusere:** Alltid lagre alt av arbeid og data, og ha en backup av alt på en ekstern harddisk eller lignende.		

8.5 Systemrisiko

8.5.1 Klarer ikke å få RCU til å kommuniserer med robot

Det er alltid en risiko at vi ikke klarer å få RCU og robot til å kommunisere, siden det er mye programmering og flere forskjellige protokoller som vi har å gjøre med. Dette vil medføre at hele prosjektet feiler.

Tabell 13 - RCU kommunikasjon

Sjanse for hendelse	**Lav**	Middels	Høy
Konsekvens	Lav	Middels	**Høy**
Håndtering	**Reduserer:** Ved å få en god forståelse av hva vi skal gjøre, de forskjellige måtene det kan gjøres på og tilegne kunnskap om verktøyene vi har til rådighet.		

8.5.2 Kommunikasjon mellom RCU og RP svikt

Det er en risiko for at problemet med få å den serielle kommunikasjon mellom RCU og RP ikke blir løst. Dette vil effektivt medføre at kommunikasjon mellom RCU og robot ikke blir oppfylt.

Tabell 14 - Kommunikasjon svikt

Sjanse for hendelse	**Lav**	Middels	Høy
Konsekvens	Lav	Middels	**Høy**
Håndtering	**Redusere:** Ved å tilegne seg kunnskap om serielle kommunikasjons protokoller, og informasjon om RCU og RP.		

8.5.3 Brente eller ødelagte komponenter

Under testing så er det en risiko for at komponentene i roboten kan bli skadet eller ødelagt. Som kan føre til at roboten ikke blir fungerende.

Tabell 15 - Komponenter

Sjanse for hendelse	Lav	**Middels**	Høy
Konsekvens	**Lav**	Middels.	Høy
Håndtering	**Redusere:** For å unngå dette problemet blir det bestilt ekstra komponenter. Skulle dette forekomme allikevel så blir nye deler bestilt så fort som mulig.		

8.6 RISIKOMATRISE

Risikomatrise er en matrise som er markert med farger for å indikere alvorlighetsgrad, både ved konsekvens og sannsynlighet. Tallene i rutene er konsekvens ganget med sannsynlighet.

Tabell 16 - Riskomatrise

Sannsynlighet	**Konsekvens.**				
	Svært liten	**Liten**	**Middels**	**Stor**	**Svært stor**
Svært liten	1	2	3	4	5
Liten	2	4	6	8	10
Middels	3	6	9	12	15
Stor	4	8	12	16	20
Svært stor	5	10	15	20	25

Tabell 17 - Forklaring av fargekode

Høy risiko.	Det er høy risiko og noe må gjøre snarest.
Middels risiko.	Litt over hva som kan aksepteres, og tiltak må vurderes.
Lav risiko.	Liten risiko og tiltak er ikke alltid nødvendig.

Tabell 18 er et matrise basert risikoanalyse som vil gi en annen vinkel å se risikoene fra. Den tar for seg forskjellige typer scenarier som kan oppstå og gir et risikoverdi basert på sannsynlighet og konsekvens.

Risikoverdien R kommer av, S*K=R.

Tabell 18 - Risikoanalyse

Hendelse.	Årsak.	S	K	R	Tiltak.
Ikke nok opplæring i AIM	KM ikke har folk å avsette	1	5	5	Holde en god dialog med KM. Ellers må vi bare akseptere at det er slik.
Hardware ikke tidsnok	Forsinkelse i forsendelse.	1	5	5	Bestille deler og tidsnok og benytte hurtigleverings muligheter. Samtidig sørge for å ha bestille ekstra sett med komponenter som går lett i stykker
Menneskelig svikt	Sykdom, ulykker, osv.	4	2	8	Spise godt, få rikelig med søvn. Få informasjon om hva som må være gjort, for å kompensere for manglende gruppe medlem.
Dårlig arbeid i gruppen	Noen ikke møter opp eller lignende.	2	4	8	Holde en så god kommunikasjon innad i gruppe som mulig, være positive og oppmuntrende.
Defekte komponenter	Feil fra produsent eller lignende.	3	3	9	Sjekke og vurdere komponentene om de er gode nok. Hvis de ikke er det, så må det bestilles nye så fort som mulig.
Data går tapt	Kræsj av pc eller person svikt.	2	3	6	Være påpasselig med å lagre alt en gjør. Og ha backup skylagring eller ekstern disk.

RCU/robot kommunikasjon Svikt	Får ikke til kommunikasjonen mellom RCU/robot.	1	5	5	Tilegne oss så god kunnskap om de forskjellige delene av systemet som mulig.
RCU/RP kommunikasjon	Får ikke RP til å skjønne hva RCU gjør.	1	5	5	Sette oss godt inn i seriell kommunikasjon og vedrørende protokoller og RP egenskaper.
Ødelagte komponenter	Bruker gjør noe feil eller de er defekte.	3	2	6	Skulle dette forekomme, så er det viktig å få bestilt nye så fort som mulig. Bestille ekstra komponenter for å minske stopp i prosessen.

Som vi ser fra analysen så har vi ingen i rødt, så vi må vurdere hvert enkelt tilfelle om det trengs tiltak. Det er ett par punkter som er ganske høyt, så der må det holdes et spesielt godt øye med.

9 PROSJEKTMODELL

9.1 INNLEDNING

Etter oppstarts undervisning av Olaf Hallan Graven gjorde vi research på forskjellige modeller. Modellen valgte vi i fellesskap er siktet til å lage et effektivt og målrettet arbeidsmiljø. Som prosjektplanen er modellen et verktøy for prosjektleder til å både kvalitetssikre arbeid og at innleveringer skjer innen tidsfrister.

9.2 MODELLEN

Modellen til prosjektet er en viktig del av prosessen til et komplett produkt. Den avgjør hvordan gruppen skal arbeide, måten det organiseres på og hvordan gruppen skal utvikle seg framover ved å forholde seg til forskjellige faser i prosjektet.

9.2.1 Unified Process

Unified Process er ikke bare en prosess, men en større arbeidsmetodikk som blir tilpasset spesifikt til prosjektet. UP er en inkrementell iterativ metodikk som tillater at vi selv tilpasser den slik vi trenger, som vil gjøre det lettere å tilpasse til utviklingen av vårt prosjekt. Mye av arbeidet som kommer til å bli gjort vil bli gjort i parallell forløpende og integreres. [1]

Gruppen valgte UP for å kunne ha muligheten til å endre på hvilket system som integreres, ettersom en del kan være seinere enn en annen i utviklingsfasen, eller når andre uforutsette problemer oppstår.

9.2.2 Hvordan vi skal jobbe

Gruppen har satt flere milepæler som vi må rekke for å holde oss på riktig spor. Hver enkel fase har et viktig mål som må gjøres før fasen blir avsluttet og prosjektet kan gå videre. Modellen i dette prosjektet gjør at vi kan fokusere på å gjøre mest mulig på de aller viktigste delene, uten å gjøre ferdig hver bidige detalj, før vi går videre med utviklingen. Så lenge funksjonaliteten er fullstendig kan vi iterere og sette opp brukervennlighet eller detaljer seinere.

9.2.3 Planen

Vi har valgt iterasjoner på to uker, dermed blir hver to ukers periode starten på en iterasjon og arbeids periode vi setter opp i tidsplanen. Iterasjoner på to uker gjør at vi heletiden har en frist i nær fremtid å jobbe etter. Prosjektet har noe større arbeidsoppgaver som trenger nok tid til å gjennomgås før vi kan gå videre, men samtidig kort nok til at vi ikke blir late. Her er det effektivitet i arbeidsukene som telles. Om vi skulle bli seine med en oppgave legger vi ukeslutt til onsdag så vi får vi også lørdag og søndag å rutte med.

9.2.4 Fasene

Unified process kan ha flere sykluser som er delt opp i fire faser, hver fase kan ha flere iterasjoner. Hver iterasjon resulterer i en inkrementell versjon av prosjektet. Hver fase er konkludert avsluttet, når en målbar milepæl har blitt nådd. Og er etterfulgt av en «ja/nei» beslutning av prosjekt gruppen om vi skal gå videre til neste fase. Som et eksempel vil A og B krav bli prioritert, testet og fullstendig ferdigstilt før vi går videre på C krav. Fasene består av Inception, Elaboration, Construction og Transition.

«Inception» er start fasen for prosjektet. Det skal opprettes en prosjektprosedyre og strukturen skal planlegges i en prosjektplan. Først opprettes det strukturdokumenter som; maler, oppfølgingsdokumenter, timelister og møtetime med veiledere planlegges. Dette legger en standard for utseende og oppsett av dokumenter, samt en stor tidsbesparelse og forenkling for gruppens dokumentasjon.

Etter nødvendige arbeidsstrukturdokumenter er laget begynner vi på prosedyredokumenter. Det lages visjonsdokument med kjernekravfunksjonalitet, restriksjoner, kravspesifikasjon, testspesifikasjon og prosjektplan med risk vurdering, samt tidlig utgave av systemarkitektur- «prototype» og useCases. Det skal også lages et overordnet tidsestimat for prosjektet som beskriver hvordan vi skal jobbe framover, og vi skal ha oppdragsgivers (KM) enighet om eventuelle kostnader.

En offentlig webside skal opprettes med informasjon om prosjektet og gruppens medlemmer. Websiden skal være en introduksjon på hvem vi er, hva vi driver med og hvordan framdriften i prosjektet går. Videre vil informasjon av interessant art legges ut.

I denne perioden skal vi ha fått et klart overblikk av systemet og hvordan det skal brukes. Oppgaven er blitt analysert og vi har kartlagt risikoer knyttet til prosjektet. Bestillinger er klargjort og vi kan begynne med elaboration fasen.

«Elaboration» er der vi lager et mer detaljert rammeverk for systemet og et bedre grunnlag for å forstå problemområdet. Det samles mer informasjon om komponenter og utstyr for en mer detaljert systemarkitektur. Systemarkitekturbeskrivelsen skal dermed bli komplett og risikoanalysen blir oppdatert med mer nøyaktig informasjon.

Vi skal lage en mer presis arkitekturprototype for å vise at større riskelementer er identifisert og løst. Produktvisjonen og arkitekturen skal bli framstilt som fornuftig og deretter godkjent. Prosjektets utviklingsplan skal være detaljert og troverdig. Kravspesifikasjon og testspesifikasjon vil bli revidert, useCases vil bli videreutviklet.

Oppdragsgivere (KM) skal være enig om at visjonen kan bli oppnådd med systemarkitekturen som er blitt spesifisert. Det viktigste for oss er å påbegynne en konsistent systemarkitektur. Siden systemet vårt er delt opp i flere deler kan vi jobbe parallelt og seinere integrere delene ettersom de blir ferdig. Når et delsystem er regnet som ferdigbehandlet har det gjennomgått generell testprosedyre før det implementeres.

«Construction» er fasen der videre utvikling, integrasjon og testing er aktivitetene. Med mengden nøyaktighet lagt inn i arkitekturplanlegging og komponentbasert arkitektur fra Inception og Elaboration fasene, skal det være mulig å sette full fokus på utvikling- og ferdigstillingsprosessen.

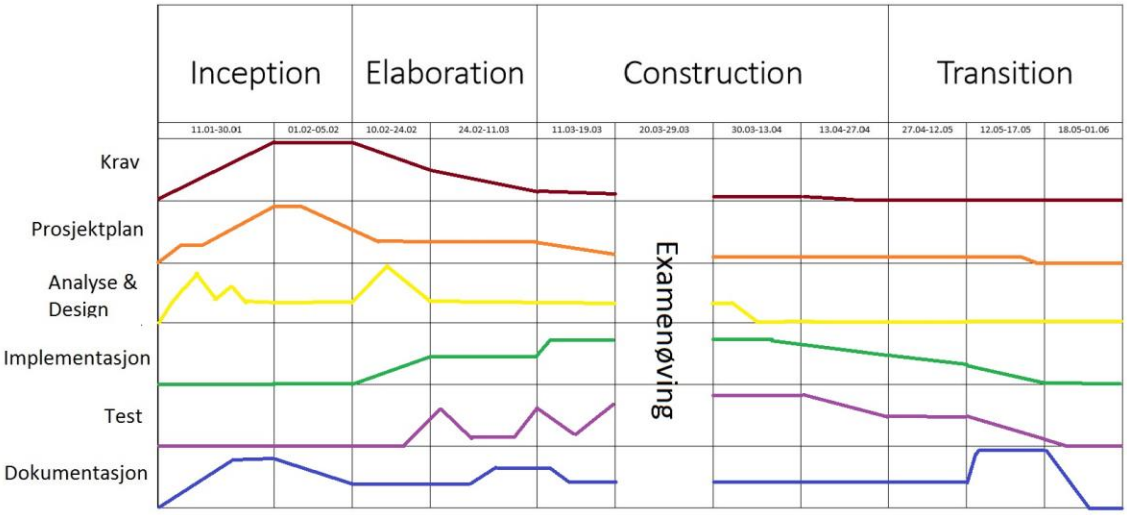
Brukermanual skal ha blitt laget og systemet er integrert og har blitt testet. Produktet skal være bestandig nok til å bli utgitt til sluttbrukerne, vertfall i form av betaversjon av systemet.

«Transition» I denne fasen så er målet å levere en versjon av produktet som er fungerende i så stor grad at vi kan vise den for sluttbruker, for å identifisere eventuelle feil eller andre ønsker i funksjonaliteten som sluttbruker mener mangler.

For å komme frem til disse målene må det fullføres et sett med punkter. Disse er å ferdigstille produktet slik at det kan vises fram for kunden, samt all dokumentasjon som kunden måtte trenge og sette produktet i drift.

For å komme i mål med dette må vi fullføre testrapport, FAT (Factory Acceptance Test), muligens gjennomføre en «blackbox» test og skrive ferdig brukermanual.

9.2.5 Skreddersydd prosjektmodell



9.2.6 Gjennomføring

Her kommer det frem hvordan vi har gjennomført fasene i prosjektet, hvordan prosjektmodellen har fungert og hvilke erfaringen vi har vi har fått.

9.2.7 Inception

Inception fasen var på begynt da vi avsluttet forelesningene med Olaf Han Graven den 11.01. Vi hadde tidligere fått avgjort oppgaven med Kongsberg Maritime, så vi kunne begynne rett på planleggingen av prosjektet. Vi ble enige om at det første som skulle settes opp var prosjektstrukturdokumenter som; maler, oppfølgingsdokument, timelister og aktivitetsplaner.

Etter dette var satt opp delte vi ut ansvarsområder og roller videre i prosjektet. Vi begynte dermed å lage prosedyredokumentene til prosjektgjennomføringen, dette gjelder visjonsdokument, prosjektmodell, kravspesifikasjon og overordnet tidsplan.

Kravspesifikasjon ble utarbeidet hyppig kontakt med ekstern veileder fra Kongsberg Maritime, dette var etter vi hadde ett møte der vi satt opp en mer detaljert oppgave for prosjektet. Prosjektmodell valgte vi så tidlig som mulig for å få verktøyene nødvendig til å styre prosjektet framover. Vi valgte unified process på grunn av blandingen av software utvikling og konstruksjon i oppgaven. Prosjektet

har en spesiell brukerfunksjonalitet som vi hadde lyst til å fokusere på, dette er det lettere å ivareta ved bruk av unified process, UseCases og muligheten til forskyving samt fokuset på eventuelle risikoer passet oss godt.

Valget med unified process satte oss i iterasjon 1 av 2, i inception fasen. Vi valgte hver fase til å ha 2iterasjoner og hver iterasjon til å vare 2uker. Dette passet oss godt siden vi har noe større komplekse arbeidsoppgaver som krever tid til nøye undersøking og gjennomføring. Vi syntes også dette passet bra da vi kunne slippe og stadig henge over hverandres skuldre og få mer rom til å gjennomføre aktivitetene.

Vi valgte først fredag til fredag arbeidsuker, men fant ut at det passet bedre med onsdag til onsdag så vi kunne stille med fullstendig og ferdig oppfølgingsdokument til intern veileder hver uke. Hver uke blei arbeidsmengden satt til 28t per pers, dette var all tiden som er foruten forelesninger, vi regnet med at arbeidsmengden kommer til å øke ett sted i konstruksjonsfasen, når vi er ferdig med andre fag.

I mellomtiden fikk vi kursing og møter med Kongsberg Maritime. Kursingen inneholdt programmering i AIM og informasjon om RCU, møtene gikk ut på system arkitekturen til oppgaven. Vi bestemte oss for å velge en modbuskobling fra RCU istedenfor å lage kretskort til IO portene. Dette valget gjorde at vi kan koble en raspberry pi til RCU og dermed seinere ha mulighet til å overføre signaler trådløst, via en wifi modul vi plasserer på raspberry pi. Løsninger ble gjennomgått og tilslutt endte vi med en bestillingsliste og ett konsept som vi valgte å gå for. Bestillinger ble gjort tidlig så vi kunne jobbe videre på kommunikasjons muligheter og løsninger i neste fase. Med de ferdige delsystemene på plass vil det være lettere å identifisere mangler ved systemet før vi skal integrere det i fase 3.

Fram mot første presentasjon ble fokuset på å få ferdig de nødvendige dokumentene, kravspesifikasjon og visjon var tidlig ferdig, så det ble retting på prosjektplan og testspesifikasjon samt risikoanalyse som ble hovedprioritet. For oss var det viktig å ferdigstille dette så prosjektet kunne holde en god struktur gjennom prosessen til ett ferdig produkt. Den største mengden med papir prosedyre dokumenter ble ferdig her, det var mer arbeid enn vi først hadde planlagt, så vi måtte bruke all tiden satt av i denne fasen til å få det ferdig.

Etter presentasjon 1 brukte gruppen litt tid på å vurdere tilbakemeldingene som ble gitt og hvordan vi kunne forbedre oss som en gruppe. Vi byttet blant annet noen ansvarsområder som vi syntes passet for å bedre effektivitet og evaluerte så reviderte noen dokumenter. Onsdagen etter første presentasjon den 10.02 gikk vi over til fase 2, elaboration fasen.

NB! Mer detaljert beskrivelse av fasegjennomføringen finnes i Iterasjonsdokument 2.0

10 REFERANSER

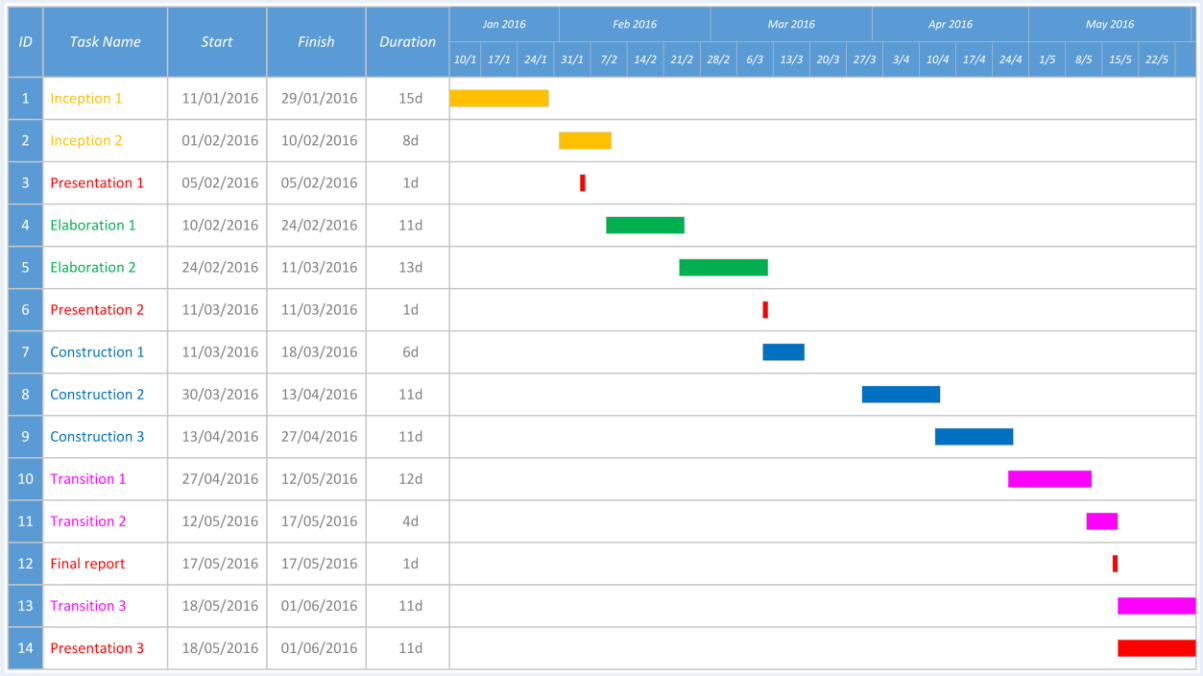
[1] BAWiki: UP - 28.02.16 - http://www.bawiki.com/wiki/concepts/sdlc-process-models/unified-process/

[2] Wikipedia: UP – 28.02.16 - https://en.wikipedia.org/wiki/Unified_Process

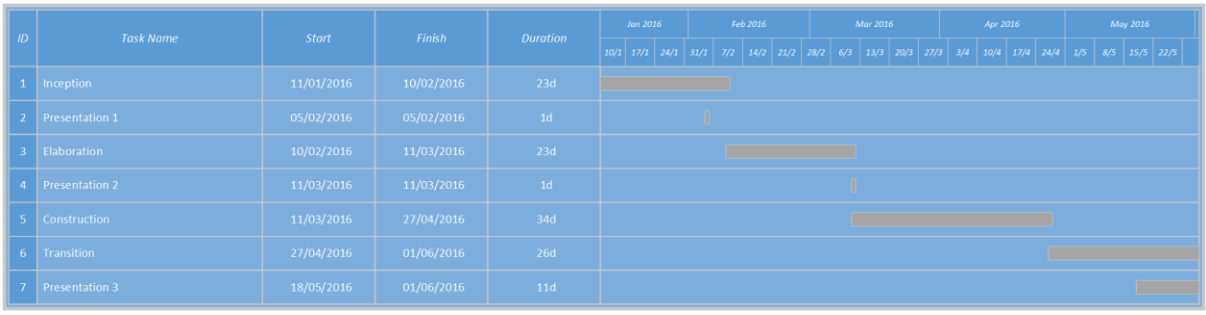
[3] 27.01.2016, http://www.ready.gov/risk-assessment

[4] 28.01.2016, http://www.scientificpapers.org/wp-content/files/07_Dumbrava_Iacob-USING_PROBABILITY__IMPACT_MATRIX_IN__ANALYSIS_AND_RISK_ASSESSMENT_PROJECTS.pdf

Vedlegg 1 - Gantt



ID	Task Name	Start	Finish	Duration
1	Inception 1	11/01/2016	29/01/2016	15d
2	Inception 2	01/02/2016	10/02/2016	8d
3	Presentation 1	05/02/2016	05/02/2016	1d
4	Elaboration 1	10/02/2016	24/02/2016	11d
5	Elaboration 2	24/02/2016	11/03/2016	13d
6	Presentation 2	11/03/2016	11/03/2016	1d
7	Construction 1	11/03/2016	18/03/2016	6d
8	Construction 2	30/03/2016	13/04/2016	11d
9	Construction 3	13/04/2016	27/04/2016	11d
10	Transition 1	27/04/2016	12/05/2016	12d
11	Transition 2	12/05/2016	17/05/2016	4d
12	Final report	17/05/2016	17/05/2016	1d
13	Transition 3	18/05/2016	01/06/2016	11d
14	Presentation 3	18/05/2016	01/06/2016	11d



ID	Task Name	Start	Finish	Duration
1	Inception	11/01/2016	10/02/2016	23d
2	Presentation 1	05/02/2016	05/02/2016	1d
3	Elaboration	10/02/2016	11/03/2016	23d
4	Presentation 2	11/03/2016	11/03/2016	1d
5	Construction	11/03/2016	27/04/2016	34d
6	Transition	27/04/2016	01/06/2016	26d
7	Presentation 3	18/05/2016	01/06/2016	11d

Vedlegg 2 - Aktivitetsplan

Inception 1

Aktivitet nr.	Beskrivelse	Ansvarlig	Estimerte timer	Brukte timer
1.00	**Inception 1**			
1.01	Oppgavebeskrivelse	Alle	8	10
1.02	Timelister	Paul & Aleks	2	3
1.03	Maler	Alle	4	14
1.04	Ukentlig oppfølgingsdokument	Masoud	2	3
1.05	Daglig dokumentasjon	Aleks	2	3
1.06	Logo	Paul	4	4
1.07	Gantt	Tor	4	5
1.08	Aktivitetslister	Aleks	10	10
1.09	Teknologidokument	Paul & Tor	5	4
1.10	Systemarkitektur	Aleks	5	6
1.11	Bestilling	Aleks & Paul	20	18
1.12	Visjonsdokument	Masoud	40	37
1.13	Prosjektmodell	Aleks	40	50
1.14	Prosjektplan	Abdu	40	42
1.15	Risikoanalyse	Tor	30	26
1.16	Kravspesifikasjon	Hanna	40	38
1.17	Testspesifikasjon	Hanna	30	33
1.18	KM møter (RCU, AIM)	Alle	50	60
		Sum	**336**	**366**

Vedlegg 3 – Aktivitetsplan

Inception 2

Aktivitet nr.	Beskrivelse	Ansvarlig	Estimerte timer	Brukte timer
2.00	**Inception 2**			
2.01	Presentasjon	Masoud	40	66
2.02	Samling av manusmateriale	Aleks	32	43
2.03	KM Kurs	Hanna & Tor	20	27
2.04	Webside	Paul	16	2
2.05	Evaluering	Alle	20	14
2.06	Revidering av dokumenter	Alle	40	16
		Sum	**168**	**168**

Vedlegg 4 – Aktivitetsplan

Elaboration 1

Aktivitet nr	Beskrivelse	Ansvarlig	Estimerte timer	Brukte timer
3.00	**Elaboration 1**			
3.01	Fulle UseCases og WBS	Masoud	26	5
3.02	Detaljert system arkitektur	Aleks	20	5
3.03	Mekanisk konstruksjon	Abdu	30	28
3.04	AIM Kurs	Hanna	21	24
3.05	Kommunikasjons protokoll	Tor Gunnar	60	66
3.06	Fasegjennomføring	Aleks	11	28
3.07	Kalibrering & ombygging robot&programmering	Aleks	62	60
3.08	Raspberry pi RCU kommunikasjonsprotokoll	Tor Gunnar	65	54
3.09	Nettside	Paul	10	21
3.10	Økonomidokument	Masoud	4	8
3.11	AIM konfigurering	Hanna	28	45
		Sum	**336**	**344**

Vedlegg 5 – Aktivitetsplan

Elaboration 2

Aktivitet nr	Beskrivelse	Ansvarlig	Estimerte timer	Brukte timer
4.00	**Elaboration 2**			
4.01	RPi/ RCU Kommunikasjonsprotokoll	Tor Gunnar	36	9
4.02	RCU/ RPi Kommunikasjonsprotokoll	Hanna	8	8
4.03	Usecases/Sekvens	Paul	4	1
4.04	Revidering av dokumenter	Masoud	12	15
4.05	Hjemmeside	Masoud	5	24
4.06	Arduino Programmering	Abdu	26	29
4.07	Designdokument	Aleks	21	23
4.08	Teknologidokument	Masoud	21	10
4.09	AIM konfigurering	Hanna	14	19
		Sum	**147**	**138**
4.10	RPI/RCU interface	Tor&Hanna	49	52
4.11	Powerpoint	Paul	4	9
4.12	Manus og visuelt innhold	Aleks	60	54
4.13	Forbredelse av dokumenter	Abdu	6	9
4.14	Designdokument	Masoud	31	34
4.15	Testplan	Masoud	18	15
		Sum	**168**	**173**
4.16	Øving til 2. Presentasjon	Alle	108	108

4.17	Presentasjon + ettermøte	Alle	12	12
4.18	Evaluering&Revidering	Alle	24	24
		Sum	**144**	**144**
	For begge ukene	**Sum totalt**	**459**	**455**

Vedlegg 6 – Aktivitetsplan

Construction 1

Aktivitet nr	Beskrivelse	Ansvarlig	Estimerte timer	Brukte timer
5.00	**Construction 1**			
5.01	Teknologidokumenter	Masoud	7	8
5.02	Designdokument	Masoud	2	2
5.03	Iterasjonsrapport	Aleksander	28	36
5.04	RPi til Arduino komm.	Abdu	23	33
5.05	RCU til RPi / Arduino	Tor Gunnar	52	29
5.06	Arduino functioncall	Masoud	17	17
5.07	AIM	Hanna	16	16
		Sum	**145**	**141**

Vedlegg 7 – Aktivitetsplan

Construction 2

Aktivitet nr	Beskrivelse	Ansvarlig	Estimerte timer	Brukte timer
6.00	**Construction 2**			
6.01	RCU & Modbus Python	Tor Gunnar	37	36
6.02	RPI til Arduino, Python	Abdu	111	86
6.03	Hexapod Functioncall& Modbus Arduino	Aleks	37	38
6.04	RPI til RPI Wifi kontroll, Python	Paul	26	16
		Sum	**211**	**176**

Vedlegg 8 – Aktivitetsplan

Construction 3

Aktivitet nr	Beskrivelse	Ansvarlig	Estimerte timer	Brukte timer
7.00	**Construction 3**			
7.01	Wifi, mottak og sending av modbus til Arduino	Tor Gunnar	111	89
7.02	RPI til Arduino, Python	Abdu	52	57
7.03	Hexapod omprogrammering	Aleks	29	32
7.04	Iterasjonsrapport	Aleks	8	8
		Sum	**200**	**186**
7.05	Koble RPI til Hexapod	Masoud	101	148
7.06	Analoge Hexapod bevegelser/funksjoner	Aleks	44	63
7.07	Sensorer	Hanna	67	47
		Sum	**212**	**258**
	For begge ukene	**Sum totalt**	**412**	**444**

Vedlegg 9 – Aktivitetsplan

Transition 1

Aktivitet nr	Beskrivelse	Ansvarlig	Estimerte timer	Brukte timer
8.00	**Transition**			
8.01	RPI autorun script	Masoud	20	13
8.02	Porting av hexapod kode til Arduino Mega og pixycam tracking eller ultrasoniske sensorer	Aleks	43	45
8.03	Hexapod exterior design + koffert	Abdu	66	80
8.04	Forbedring av funksjonalitet i kommunikasjonsnettverket. (AKA bug squashing)	Tor Gunnar	54	71.5
8.05	Implementere livefeed RPI	Paul	32	4.5
8.06	AIM, bruker interface	Hanna	43	43
		Sum	**258**	**257**
8.07	Pixycam tracking	Aleksander	27	10
8.08	Hexapod exterior design	Masoud	65	72
8.09	AIM - Hexapod (Start og stop funksjon)	Tor Gunnar	58	57
8.10	Utbedre system funksjonaliteten	Paul	42	38
8.11	AIM, bruker interface	Hanna	35	31
8.12	Ultrasoniske sensorer	Abdurahman	25	24
8.13	Iterasjonsrapport	Aleksander	8	8
8.14	Batteri løsninger	Aleksander	10	27
		Sum	**270**	**267**
	For begge ukene	**Sum totalt**	**528**	**524**

Vedlegg 10 – Aktivitetsplan

Transition 2

Aktivitet nr	Beskrivelse	Ansvarlig	Estimerte timer	Brukte timer
9.00	**Transition 2**			
9.01	Designdokument	Tor Grunnar	57	62
9.02	Plakat	Paul	25	20
9.03	Pixycam tracking	Aleks	20	20
9.04	Autonomous	Abdu	20	20
9.05	Brosjyrer/Klistremerker	Masoud	18	4
9.06	Iterasjonsrapport	Aleks	8	16
9.07	Testrapporter	Abdu	70	77
9.08	Teknologidokument	Tor Gunnar	5	7
9.09	Brukermanual	Masoud	27	24
9.10	Kravsporing til aktiviteter	Hanna	13	0
9.11	Forbedring koding RPI(2)	Paul	5	5
9.12	Økonomidokument	Masoud	2	3
		Sum	**270**	**258**
10.01	Forberedelse til presentasjon 3	Alle	94	
10.02	Forberedelse av all dokumentasjon til innlevering	Masoud	160	
10.03	Designdokument	Aleksander	16	
	For begge ukene	**Sum totalt**	**540**	

Høgskolen i Sørøst-Norge

KONGSBERG

ITERASJONSDOKUMENT

PROSJEKT	K-Spider		
OPPDRAGSGIVER	Kongsberg Maritime AS		
UTFØRT VED	Høgskolen i Sørøst-Norge, avd. Kongsberg		
REVISJON	2.0		
MEDLEMMER	Aleksander Tokle Poverud, Masoud Shah Pasand, Tor Gunnar Finnerud, Hanna Kåsastul, Paul Knutson Sæther, Abdurahman Senkaya		
Dokumenthistorikk	**REVISJON**	**UTGITT**	**BESKRIVELSE**
	1.0	20.03.2016	Første utgave
	2.0	21.05.2016	Andre utgave

INNHOLDSFORTEGNELSE

1 DOKUMENTHISTORIE 3

2 INNLEDNING 4

3 Inception fasen 5
3.1 Inception 1 5
3.1.1 Iterasjonsplan for Inception 1 5
3.2 Inception 2 9
3.2.1 Iterasjonsplan for Inception 2 9

4 Elaboration fasen 11
4.1 Elaboration 1 11
4.1.1 Iterasjonsplan for Elaboration 1 11
4.2 Elaboration 2 15
4.2.1 Iterasjonsplan for Elaboration 2 15

5 Construction fasen 20
5.1 Construction 1 20
5.1.1 Iterasjonsplan for Construction 1 20
5.2 Construction 2 23
5.2.1 Iterasjonsplan for Construction 2 23
5.3 Construction 3 26
5.3.1 Iterasjonsplan for Construction 3 26
6.1 Transition 1 31
6.1.1 Iterasjonsplan for Transition 1 31
6.2 Transition 2 36
6.2.1 Iterasjonsplan for Transition 2 36
6.3 Transition 3 40
6.3.1 Iterasjonsplan for Transition 3 40

TABELLER

Tabell 1 - Dokumenthistorie 3
Tabell 2 - Definisjoner og forkortelser 4
Tabell 3 - Overordnet tidsplan for iterasjon 1 (Inception 1 fasen) 5
Tabell 4 - Detaljerte Aktiviteter for iterasjon 1 (Inception 1 fasen) 7
Tabell 5 - Viktige gjøremål for iterasjon 2 (Inception 2 fasen) 9
Tabell 6 - Detaljerte Aktiviteter for iterasjon 2 (Inception 2 fasen) 10
Tabell 7 - Viktige gjøremål for iterasjon 1 (Elaboration 1 fasen) 11
Tabell 8 - Detaljerte Aktiviteter for iterasjon 1 (Elaboration 1 fasen) 13
Tabell 9 - Viktige gjøremål for iterasjon 2 (Elaboration 2 fasen) 15
Tabell 10 - Detaljerte Aktiviteter for iterasjon 2 (Elaboration 2 fasen) 17
Tabell 11 - Viktige gjøremål for iterasjon 1 (Construction 1 fasen) 20
Tabell 12 - Detaljerte Aktiviteter for iterasjon 1 (Construction 1 fasen) 21
Tabell 13 - Viktige gjøremål for iterasjon 2 (Construction 2 fasen) 23
Tabell 14 - Detaljerte Aktiviteter for iterasjon 2 (Construction 2 fasen) 24

Tabell 15 - Viktige gjøremål for iterasjon 3 (Construction 3 fasen) 26
Tabell 16 - Detaljerte Aktiviteter for iterasjon 3 (Construction 3 fasen) 28
Tabell 17 - Viktige gjøremål for iterasjon 1 (Transition 1 fasen) 31
Tabell 18 - Detaljerte Aktiviteter for iterasjon 1 (Transition 1 fasen) 33
Tabell 19 - Viktige gjøremål for iterasjon 2 (Transition 2 fasen) 36
Tabell 20 - Detaljerte Aktiviteter for iterasjon 2 (Transition 2 fasen) 38
Tabell 21 - Viktige gjøremål for iterasjon 3 (Transition 3 fasen) 40

1 DOKUMENTHISTORIE

Tabell 1 - Dokumenthistorie

VERSJON NR:	DATO ENDRET	Endret av:	Godkjent av:	BESKRIVELSE
0.1	16.03.2016	Aleksander	Masoud	• Dokumentet ble opprettet • Satt sammen alle iterasjonsrapporter til et dokument.
1.0	20.03.2016	Aleksander	Masoud	• Første offisielle utgivelse
1.1	22.03.2016	Masoud	Aleksander	• Rettet på feilskriving og formatering. • Oppdatert innholdet i noen av fasene.
1.2	14.04.2016	Aleksander	Masoud	• Lagt til Construction 1-3
1.3	05.05.2016	Aleksander	Masoud	• Construction 3 oppdatert
1.4-5	12.05.2016	Aleksander	Masoud	• Oppdatert Transition 1&2
2.0	21.05.2016	Aleksander	Masoud	• Siste tillegg Transition 3 • Godkjent og utgitt

2 INNLEDNING

Dette iterasjonsdokumentet er for iterasjonsrapportene til gruppen K-Spider.

Dokumentet inneholder alle iterasjonene vi har i den overordnede fremdriftsplanen, prosjektplandokumentet. Hver iterasjon inneholder en iterasjonsplan og en iterasjonsrapport. Vi bruker prosjektmodellen Unified Process som har fire faser; Inception, Elaboration, Construction og Transition. Hver av fasene har to eller flere iterasjoner på omtrent to uker per.

En iterasjon er en del av en periode i en fase der vi jobber med viktige gjøremål og aktiviteter slik at vi når de milepælene vi har satt. De forskjellige iterasjonene vil ha forskjellige aktiviteter med hensyn til hvilken fase de tilhører. For eksempel i den innledende Inception fasen vil vi ha hovedfokus i planleggingsaktiviteter og dokumentasjon, altså det grunnleggende arbeidet for prosjektet. Elaboration fasen skal det være fokus på å utdype detaljer og forbedrede systemet for integrasjon, dette inkluderer konstruksjon av delsystemer. I konstruksjonsfasen er hovedfokuset på integrasjon av de forskjellige delsystemene og få de til å fungere i samspill, det blir mye programmering i denne fasen. Den avsluttende Transition fasen vil gå til dokumentering og ferdigstilling slik at oppdragsgiver kan overta produktet. Aktivitetene for iterasjonen vil dermed gjenspeile dette.

Planen skal skrives i starten av hver iterasjon og det skal inneholde hva som skal gjøres innen denne iterasjonen er over. Det skal inkludere mål for iterasjonen og hva som ønskes å oppnås med denne iterasjonen. Det skal skrives en aktivitetsliste for alle aktiviteter som skal jobbes med, hvem som er ansvarlig for aktiviteten, hvem som skal jobbe med aktiviteten og hva statusen på aktiviteten er etter iterasjonen er over. Det skal også ett kort sammendrag om hva hver hovedaktivitet spesifikt innebærer, dette gjelder i hovedsak «viktige gjøremål» fra fremdriftsplanen, men flere kan være beskrevet.

Iterasjonsrapporten skal skrives etter hver iterasjon. Denne rapporten inneholder aktivitetslistene fra planen, og hvilket resultat som ble oppnådd. Samt utfordringer og eventuelle tiltak, med konklusjon.

Aktivitetslister og viktige gjøremål er hentet fra prosjektplan.

Tabell 2 - Definisjoner og forkortelser

Navn	Forkortelser
Aleksander Tokle Poverud	ATP
Masoud Shah Pasand	MSP
Tor Gunnar Finnerud	TGF
Paul Sæther	PS
Abdurahman Senkaya	AS
Hanna Kåsatul	HK

3 Inception fasen

3.1 Inception 1

Periode: 11.01.2016 - 30.01.2016

3.1.1 Iterasjonsplan for Inception 1

Mål

Målet i denne iterasjonen var å få plass arbeidsstrukturdokumenter, prosedyredokumenter, ett overblikk over systemarkitekturen, utstyr og komponenter som vi trengte til å gå videre med prosjektet. Bestillinger vi skal foreta oss består da i hovedsak av delsystemer som hovedsystemet skal bygges opp av.

Vi må utrede de forskjellige kravene som skal definere systemets egenskaper. Dermed er det viktig at vi blir enige med Kongsberg Maritime angående hva resultat produktet skal være i denne iterasjonen. Til kravene må vi opprette en tilhørende testplan som skal gi oss muligheten til å evaluere statusen til kravene etterhvert som prosjektet går fremover

For å kunne sette disse testene er det viktig at vi har en konkret plan på hvordan vi har tenkt å løse de tekniske utfordringene. Det innebærer hva slags hardware vi velger, hvilken type kommunikasjon vi skal ha mellom disse komponentene og hva slags type programmeringsspråk vi ender opp å bruke, tilhørende til våre valg.

Vi har allerede fått tildelt ekstern veileder og sensor, og er i god kontakt med veileder. Vi prøver enda å finne ut hvilket rom vi skal få av skolen, og venter på svar angående intern veileder fra skolen.

Tabell 3 - Overordnet tidsplan for iterasjon 1 (Inception 1 fasen)

Fase	Iterasjon	Start	Estimert ferdig	Ferdig	Viktige gjøremål
Inception	1	11.01	30.01	30.01	• Visjonsdokument • Prosjektplan • Kravspesifikasjon • Testspesifikasjon • Systemkurs (AIM, RCU) • Systemarkitektur • Første del teknologidokument • Bestilling

Viktige gjøremål detaljert

I hovedsak må det være fullført en versjon av visjonsdokument, prosjektplan, kravspesifikasjon og testspesifikasjon. Til disse dokumentene må vi lage maler som har lik font, layout og overskrifter. Vi skal også opprette en god forside som er informerende om dokumentet.

1.02: Det lages timelister som gjelder hver enkelt og fulle gruppen sammenlagt, dette dokumentet skal være automatisk sånn at vi slipper å regne timer hver gang vi skal føre opp for gruppen. Samtidig opprettes det dokumenter som skal holde orden på hva det blir brukt tid og ressurser på.

1.09: Teknologidokument for viktige komponenter skal påbegynnes. Dette dokumentet skal inneholde spesifikasjoner og begrunnelse for valgt design, konsept eller komponenter. Mest sannsynlig vil noen av avgjørelsene begrunnes med pughmatriser, ellers vil dokumentet inneholde konklusjon og forklaringer om det valgte designet.

1.10: Det skal også lages en systemarkitektur for å få oversikt over hva som må bestilles av komponenter til systemet slik at vi kan se hva som eventuelt mangler for å lage ett komplett system.

1.12: Visjonsdokumentet skal inneholde en oppgavebeskrivelse, prosjektgruppen skal følge dens mål og forventninger for å gjennomføre oppgaven. Det skal også stå litt om gruppen, arbeidsgiver og kontaktpersoner som er knyttet til. Hensikten med dokumentet er å gi prosjektgruppen en felles forståelse i valg av prosjektmodell, en tydelig forståelse av oppgaven, og en god tidsestimering av prosjektgjennomføring.

1.14: Prosjektplanen skal inneholde:

- Mål og rammer for prosjektet
- Forutsetninger og avgrensninger
- Hvordan vi skal gjennomføre prosjektet
  - Prosjektmodell
  - Framdriftsplan
- Møtestruktur
- Aktivitetslister

1.16: Kravspesifikasjonen setter vi sammen i nært samarbeid med arbeidsgiver. Planen er å dele kravene inn i disse kategoriene:

- Rammekrav
- Funksjonelle krav
- Maskin- og programvarekrav.

1.17: Testspesifikasjonen skal ta for seg de kravene som vi har satt og forteller om hvordan vi skal evaluere tilstanden til kravene. Kravene må først analyseres for å kunne finne best mulig testmetode. Det må også settes opp godkjenningskriterier og rangering av feil eller mangler ved test.

Tabell 4 - Detaljerte Aktiviteter for iterasjon 1 (Inception 1 fasen)

Aktivitet nr.	Beskrivelse	Ansvarlig	Frist	Ressurs	Status
1.00	**Inception 1**		30.01	Alle	Avsluttet
1.01	Oppgavebeskrivelse	Alle	30.01	MSP, ATP, AS	Ferdigstilt
1.02	Timelister	Paul & Aleks	30.01	PS, ATP	Ferdigstilt
1.03	Maler	Alle	30.01	Alle	Ferdigstilt
1.04	Ukentlig oppfølgingsdokument	Masoud	30.01	MSP	Ferdigstilt
1.05	Daglig dokumentasjon	Aleks	30.01	ATP	Ferdigstilt
1.06	Logo	Paul	30.01	PS	Ferdigstilt
1.07	Gantt diagram	Tor	30.01	PS&ATP	Ferdigstilt
1.08	Aktivitetslister	Aleks	30.01	ATP	Ferdigstilt
1.09	1ste Teknologidokument	Paul & Tor	30.01	TGF&PS	Ferdigstilt
1.10	Systemarkitektur	Aleks	30.01	AS&ATP	Ferdigstilt
1.11	Bestilling	Aleks & Paul	30.01	ATP&PS	Ferdigstilt
1.12	Visjonsdokument	Masoud	30.01	MSP	Ferdigstilt
1.13	Prosjektmodell	Aleks	30.01	ATP	Ferdigstilt
1.14	Prosjektplan	Abdu	30.01	AS&ATP	Ferdigstilt
1.15	Risikoanalyse	Tor	30.01	TGF	Ferdigstilt
1.16	Kravspesifikasjon	Hanna	30.01	HK	Ferdigstilt
1.17	Testspesifikasjon	Hanna	30.01	HK	Ferdigstilt
1.18	KM møter (RCU, AIM)	Alle	30.01	Alle	Ferdigstilt

Overholdelse

Aktivitetene vi planla i iterasjonen har blitt overholdt. Vi har produsert alt i tabell 4. Men vi nevner spesielt de viktigste gjøremålene som vi ferdigstilte:

- **1.09:** Første del av teknologidokument
- **1.10:** Systemarkitektur
- **1.11:** Bestilling
- **1.12:** Visjonsdokument
- **1.14:** Prosjektplan
- **1.16:** Kravspesifikasjon
- **1.17:** Testspesifikasjon
- **1.18:** Systemkurs (AIM, RCU)

Vi har følgende veiledere:

- Intern veileder: Jørn Breivoll
- Ekstern veileder: Qui-Huu Le-Viet
- Ekstern sensor: Merethe Gotaas

Utfordringer og tiltak

Det første som ble gjort først var å sette opp en liste med gjøremål som måtte fullføres. Dette ble gjort litt fortløpende ettersom prosjektplanen var under arbeid. Det å finne ut hva alle de viktigste dokumentene skulle inneholde var en tidskrevende prosess. Når vi fikk orden på hva som skulle være med i dokumentene gikk det fort fremover.

Kravspesifikasjonen gikk greit med hyppig kontakt med arbeidsgiver, testspesifikasjonen ble laget deretter.

Når det gjelder det tekniske så brukte vi tid på visjonen til oppgaven. Vi fikk ett større overblikk over hva systemet skulle gjøre, ved å se på en tidlig systemarkitektur kunne vi ta valg av komponenter og løsninger. Deretter ble det gjort en bevisst bestilling i samarbeid med arbeidsgiver som vi hadde laget flere utkast av.

Konklusjon

Vi kan konkludere med at den første iterasjonen har gått som planlagt. Vi har nådd alle de viktige gjøremålene vi hadde satt for iterasjonen. Vi har prøvd så godt vi kan og holde orden på en litt større gruppe, men regner med at det blir lettere framover nå som prosjektplanen er på plass. Vi har i tillegg gjort en del andre tilhørende aktiviteter, blant annet forberedt videre arbeid med delsystemene når vi får komponentene. Neste iterasjon står 1. presentasjon og forberedelse som viktigste gjøremål.

3.2 Inception 2

Periode: 01.02.2016 – 10.02.2016

3.2.1 Iterasjonsplan for Inception 2

Mål

I denne perioden skal vi jobbe med å ferdigstille første presentasjon og ta til oss evaluering fra presentasjonen. Eventuelle tilbakemeldinger kan få oss til å måtte endre prosjektplanen, eller prosjektmodell. Det vil kanskje bli aktuelt å revidere krav eller testspesifikasjon hvis dette er for vagt. Videre blir det å legge opp for overgang til neste fase der vi må utdype systemet.

Tabell 5 - Viktige gjøremål for iterasjon 2 (Inception 2 fasen)

Fase	Iterasjon	Start	Estimert ferdig	Ferdig	Viktige gjøremål
Inception	2	01.02	10.02	10.02	• Presentasjon • Evaluering

Viktige gjøremål detaljert

Først skal vi gå over og rette på dokumenter før de skal skrives ut og leveres 48 timer før presentasjonen.

2.01: Presentasjon - det skal settes opp en presentasjon som tar for seg:

- Hvem vi her
- Hva oppgaven vår går ut på
- Hvordan vi har planlagt prosjektperioden
- Prosessmodell
- Hvordan vi skal jobbe fremover

Samt litt lett teknisk om systemet vi prosjekterer. Samling av manus materiale går i hovedsak ut på å hente inn og korte ned informasjon fra dokumenter eller teknisk arbeide vi allerede har utført.

2.03: KM kurs er kurs i AIM konfigurering hos KM.

2.04: Webside er å sette opp en hjemme portal for oppgaven som gir oss muligheten til å legge ut informasjon om prosjektet og dets framdrift for offentligheten.

Tabell 6 - Detaljerte Aktiviteter for iterasjon 2 (Inception 2 fasen)

Aktivitet nr.	Beskrivelse	Ansvarlig	Frist	Ressurs	Status
2.00	**Inception 2**				Avsluttet
2.01	Presentasjon	Masoud	10.02	ATP	Ferdigstilt
2.02	Samling av manusmateriale	Aleks	10.02	Alle	Ferdigstilt
2.03	KM Kurs	Hanna & Tor	10.02	HK, TGF,PS	Påbegynt
2.04	Webside	Paul	10.02	PS	Ikke ferdig
2.05	Evaluering	Alle	10.02	Alle	Ferdigstilt
2.06	Revidering av dokumenter	Alle	10.02	MSP, ATP	Ferdigstilt

Overholdelse

Nesten alle planlagte aktiviteter ble gjennomført, vi kan gå videre siden alt avgjørende for prosjektets framgang er ferdigstilt.

Vi fikk laget presentasjonen, og presentert. Dokumentene ble rettet og skrevet ut. Vi fikk påbegynt AIM kurs og vi har revidert dokumenter etter tilbakemelding fra sensor.

Utfordringer og tiltak

Inception iterasjon 2 har vært en kort iterasjon, og er den siste iterasjonen der planlegging og tilrettelegging av dokumentasjon er hovedfokus. Det har vært få utfordringer i denne fasen.

Presentasjonen ble muligens noe manglende, vi har tatt til oss tilbakemeldingene fra sensor angående manus, bilder, også videre. Vi har laget plan for å forbedre dette til presentasjon 2 og det krever mer effektivitet fra enkeltpersoner framover. Ekstern sensor kunne ikke møte, men vi foretar enda tidligere innkalling og håper vi får oppmøte til neste presentasjon. Vi har også byttet ansvarlige fra økonomi til presentasjon og vi håper dermed å få styr på neste presentasjon. Det meste av bestillinger er allerede gjort, men det skal snart lages økonomidokument. Websiden ble ikke satt opp, men vi setter av mer tid til dette i neste iterasjon.

Nå som Inception fasen er over skal vi gå videre fra hovedfokuset på prosjektplanlegging over til konstruksjon av delsystemer. Vi skal se videre på mer detaljert system arkitektur og oppbygning, i Elaboration fasen, ved å få opp delsystemene kan vi lettere se mangler ved systemet.

Konklusjon

Kan konkludere med at denne iterasjonen har gått greit, noe mangler og litt mindre effektivitet enn forventet, men vi er fortsatt i rute i forhold til fremdriftsplanen. Vi har tatt til oss endel tilbakemeldinger som vi skal jobbe med videre. Vi går videre til Elaboration fasen.

4 Elaboration fasen

4.1 Elaboration 1

Periode: 10.02.2016 - 24.02.2016

4.1.1 Iterasjonsplan for Elaboration 1

Mål

Vi har allerede bestemt delsystemene og jobben blir å koble disse sammen, teste de, passe på at de fungerer og rangere eventuelle feil etter alvorlighetsgrad. Om det finnes mangler blir det som sidearbeid å anskaffe mellom koblingselementer for kommunikasjonsnettverket.

I samkjør med det å fysisk konstruere skal vi begynne å se på hvordan vi skal foreta tilpassingen og programmeringen for de delsystemene som skal integreres. Det blir mest å se etter riktig kommunikasjonsprotokoll mellom de komponentene vi skal sette sammen. Det blir satt av en god del tid til å sette seg inn i AIM, å få en oversikt over konfigurasjonen av det.

Vi skal også opprette nye teknologidokument og økonomidokument for valg og bestillinger vi har gjort i Inception fasen.

Tabell 7 - Viktige gjøremål for iterasjon 1 (Elaboration 1 fasen)

Fase	Iterasjon	Start	Estimert ferdig	Ferdig	Viktige gjøremål
Elaboration	1	10.02	24.02	20.02	• Komplett systemarkitektur • Teknologidokument • Mekanisk konstruksjon • AIM oversikt • Finne kommunikasjonsprotokoll

Viktige gjøremål detaljert

3.02: Detaljert systemarkitektur: Gjøre de siste valgene, angående hva vi skal ha i kommunikasjonsnettverket mellom robot og RCU. Det gjelder også å få laget noen flere UseCase tilfeller og diagrammer for å få mer forståelse over systemet som vi skal bygge.

3.03: Mekanisk konstruksjon: Dette går ut på å sette sammen den fysiske rammen til roboten, koble på servoer til servo kort, kalibrere å passe på at alle servoer har riktig utgangsstilling så roboten kan gå rett.

3.05: Finne kommunikasjonsprotokoll: Enkelt og greit avgjøre de forskjellige kommunikasjons mulighetene mellom enheter i systemet, vite hva som er mest aktuelt, eventuelt finne nok informasjon angående dette til å starte på koblinger. Da mest fokus på modbuskoblingen mellom RCU og RPI.

3.07: Ombygging & programmering: Rette opp konstruksjonsfeil. Sette arduinokort på servokort, koble TxRx seriell kobling, koble på IR mottaker, legge til biblioteker, endre pins i kode og sette baudrate, få robot til å bevege seg, om det blir tid.

3.09: Nettside: Få denne opp siden den ikke ble opprettet i Inception fasen.

3.10: Økonomidokument: Føre prosjektkostnader og samle innkjøpsinformasjon og leverandør informasjon i ett oversiktlig dokument. Dette skal inkludere metode for framgang ved bestillinger og ekstra kostnader som ikke gjelder bestillinger av komponenter.

3.11&3.04: AIM: Jobbe med å finne de løsningene som vi trenger for å få kommunikasjon og kontroll over hexapoden vi skal styre. Dette inkluderer å møte opp på kurs med instruktør fra KM.

Tabell 8 - Detaljerte Aktiviteter for iterasjon 1 (Elaboration 1 fasen)

Aktivitet nr	Beskrivelse	Ansvarlig	Frist	Ressurs	Status
3.00	**Elaboration 1**				Avsluttet
3.01	Fulle UseCases og WBS	Masoud	24.02	ATP, PS	Ferdigstilt
3.02	Detaljert system arkitektur	Aleks	24.02	AS	Ferdigstilt
3.03	Mekanisk konstruksjon	Abdu	24.02	ATP, MSP, AS	Ferdigstilt
3.04	AIM Kurs	Hanna	24.02	HK, TGF,PS	Ferdigstilt
3.05	Kommunikasjons protokoll	Tor Gunnar	24.02	TGF, HK, ATP	Funnet
3.06	Kravendringeshåntering /Fasegjennomføring	Aleks&Masoud	24.02	MSP/ATP	Ferdigstilt
3.07	Kalibrering & ombygging robot&programmering	Aleks	24.02	ATP, AS, MSP	Ferdigstilt
3.08	Raspberry pi RCU kommunikasjonsprotokoll	Tor Gunnar	24.02	TGF, MSP, PS	Opprettet kommunikasjon
3.09	Nettside	Paul	24.02	PS, MSP	Ikke opprettet
3.10	Økonomidokument	Masoud	24.02	MSP	Ferdigstilt
3.11	AIM konfigurering	Hanna	24.02	HK	Underarbeid

Overholdelse

Vi er fortsatt i rute når det gjelder framdriftsplanen, se tabell 8. Alt som skal ha blitt gjennomført for å kunne gå videre med prosjektet har blitt gjennomført, modellen ser ut til å passe oss veldig godt hittil.

Gjennomført denne iterasjonen:

- **3.03:** Mekanisk konstruksjon. Hexapoden er nå fullt aktiv, 2 uker før planlagt med ekstern kontroller.
- **3.05:** Funnet en god kommunikasjonsprotokoll.
- **3.06:** Fasegjennomføring ble endret til kravendringshåndtering, begge er utført.
- **3.07:** Hexapod er bygget, kalibrert og programmert.
- **3.08:** Modbus protokollen er opprettet så vi kan begynne å få kontakt med Rpi
- **3.10:** Økonomidokument er laget.
- **3.11:** AIM arbeidet pågår, men det begynner å se bedre ut.

To teknologidokumenter er blitt skrevet som ifølge prosjektplan, mer av dette kommer etterhvert som vi velger teknologier.

Utfordringer og tiltak

Mekanisk konstruksjon med all ombygging og programmering er ferdigstilt. Roboten er funksjonell og har blitt testet. Alle funksjoner fungerer som de skal og den kan bevege seg fritt, men noe slark i beinene, og noe litt høye PWM signaler får beinene til å kollidere. Ved endring i PWM bredde på beina må vi passe på at bevegelsesmønster(gaits) ikke påvirkes, da disse allerede er godt optimalisert for å bære vekten til Hexapoden, dette er for å unngå kjapp utlading.

Vi har ikke gjort noen valg angående systemarkitekturen i denne omgang og utsetter det til neste iterasjon. Vi føler det blir litt tidlig siden vi akkurat har fått opp de to systemene vi skal sette sammen. Kommunikasjonsprotokollen er opprettet og vi forsøker å få kontakt med RPI gjennom modbus. Dette er en stor risikofaktor i prosjektet ettersom det er veldig vanskelig og det er usikkerhet i om det i det heletatt er mulig å få til. Industriverktøyet RCU510 og Rpi kan være inkompatible.

Nettsiden er enda ikke opprettet, heldigvis stopper ikke det framgangen i prosjektet, vi får heller legge ut ett par ekstra innlegg når den endelig kommer på plass.

Vi skal fortsette å se på utdypende systemarkitektur, da gjerne programmeringsløsninger i neste iterasjon.

Konklusjon

Vi har fått satt opp de delsystemene vi trenger for å teste, se mangler eller finne komponenter vi ønsker i systemet. Det meste går etter planen, vi er litt foran tidsplanen når det gjelder hexapoden og arduino. Prosessmodellen passer oss godt og vi er på god vei.

4.2 Elaboration 2

Periode: 24.02.2016 - 11.03.2016

4.2.1 Iterasjonsplan for Elaboration 2

Mål

Vi skal ha fokuset på å utdype og tilrettelegge systemet, finne feil og mangler slik at vi kan begynne med Construction fasen etter Elaboration fasen. Vi skal ha stilt ferdig, eller ha god nok kunnskap til delsystemene våre til å kunne begynne å integrere de. Det er viktig for oss å ha en komplett forståelse, da dette blir den siste delen der vi ser på delsystem løsninger. Etter denne fasen kommer det altså ingen nye komponenter som skal endre programmeringsspråkene eller måten vi skal gå framover på. Det blir derimot mulighet til å se på alternativer til systemet som gjelder forbedringer i hardware eller tilleggs sensorer.

Vi skal få den forståelsen vi trenger rundt systemet, være klar over det vi holder på med og teste at komponenter og delsystemer fungerer som det skal. Videre skal vi formidle det tekniske vi har holdt på med til presentasjon 2. Vi skal gjøre rede for det vi har planlagt og det tekniske det innebærer for prosjektet.

Vi skal ha laget starten på designdokumentet slik slutt systemet skal være, samtidig som vi skal utvide testplan for å dekke de nye delsystemene vi tilfører nå i denne iterasjonen. Vi skal ha en komplett oversikt, ett komplett system som skal jobbes med videre og dokumentasjonen skal være revidert og klargjort.

Tabell 9 - Viktige gjøremål for iterasjon 2 (Elaboration 2 fasen)

Fase	Iterasjon	Start	Estimert ferdig	Ferdig	Viktige gjøremål
Elaboration	2	24.02	11.03	11.03	• Ekstern kontroller • Implementere kommunikasjonsprotokoll • Presentasjon

Viktige gjøremål detaljert

Vi skal lage en presentasjon som skal være en stor forbedring fra presentasjon 1. Det skal være en presentasjon komplett uten manus og mer teknisk informasjon.

4.12&4.11: Oppsett av presentasjonen som skal lages:

- Kort introduksjon
- Kort og GODT om oppgaven
- Lite tilbakeblikk fra hva vi snakket om i første presentasjon
- Hvordan vi har tenkt/tenker når vi jobber
    - Møte de kravene vi har satt
    - Hvordan vi jobber med implementasjon av delsystemene
    - Aktuelle tillegg til prosjektet eksempel: sensorer.
- Kvalitetssikring av det vi produserer
    - Testplan (testfaser), risikoanalyse og pughmatriser
- Kort oversikt om hva som har blitt gjort hittil
    - Innkjøp, kobling, kalibrering, Modbuskobling, RCU, RPI, test, forberedelse av programmering

Presentasjon av vårt valgt design:

- Bilder: UseCase, system oppbygging, sekvensdiagram
- Modbus (Hva er dette, hvordan, hvorfor bruker vi det?)
- RPI (Hva er dette, hvordan, hvorfor bruker vi det?)
- Hexapod forklaring av Konstruksjon med bilder
    - Består av etc.
    - Hvorfor valg av denne

Avslutning av presentasjon:

- Innkjøp
- Hvordan går det i forhold/sammenheng med prosjektplan
- Hva skjer videre
- Avslutning

Dette er kort og godt hva vi skal ha med og hva vi har jobbet med hittil. Noe av dette skal vi jobbe med videre som man kan se i den detaljerte aktivitetslisten, tabell 10.

4.01&4.02&4.10: hovedsak er vi mest interessert i å få til modbus kommunikasjonen nå i denne iterasjonen. Vi har allerede stilt ferdig eksternkontroller (hexapod RF kontroll) fra fremdriftsplanen. Modbus kommunikasjon er industristandard som vi har mellom en Raspberry pi og RCU'en, dette er en veldig komplisert del av prosjektet med veldig høy risikofaktor.

4.05: Nettside aktiviteten er å få websiden opp, den har manglet lenge nok.

4.06: Det skal i tillegg gjøres noe arbeid i form av arduino programmering, dette blir mest sannsynlig lagt på en enkeltperson for å legge klart forskjellige integrasjonsmetoder og hva vi skal jobbe med av programmering generelt framover. Inneholder også metode for sammenkobling av arduino og Rpi, i tillegg til programmering av hexapodens funksjoner som skal styres fra RCU.

4.07: Designdokumentet er viktig og skal inneholde alle elementer av det tekniske vi har jobbet med, mesteparten av dette har blitt ramset opp i presentasjonsforklaringen.

Tabell 10 - Detaljerte Aktiviteter for iterasjon 2 (Elaboration 2 fasen)

Aktivitet nr	Beskrivelse	Ansvarlig	Frist	Ressurs	Status
4.00	**Elaboration 2**				Avsluttet
4.01	RPi/ RCU kommunikasjonsprotokoll	Tor Gunnar	11.03	TGF	Opprettet kontakt
4.02	RCU/ RPi kommunikasjonsprotokoll	Hanna	11.03	TGF, HK	Opprettet kontakt
4.03	Usecases/Sekvens	Paul	11.03	ATP,PS	Ferdigstilt
4.04	Revidering av dokumenter	Masoud	11.03	MSP	Ferdigstilt
4.05	Hjemmeside	Masoud	11.03	MSP,PS	Opprettet
4.06	Arduino Programmering	Abdu	11.03	ATP	Ferdigstilt
4.07	Designdokument	Aleks	11.03	ATP,MSP, HK, AS	Opprettet
4.08	Teknologidokument	Masoud	11.03	AS	Endret
4.09	AIM konfigurering	Hanna	11.03	HK	Endret
4.10	RPI/RCU Interface	Tor& Hanna	11.03	TGF, HK	Endringer
4.11	PowerPoint	Paul	11.03	ATP	Ferdigstilt
4.12	Manus og visuelt innhold	Aleks	11.03	ATP,MSP,TGF,PS ,HK	Ferdigstilt
4.13	Forberedelse av dokumenter	Abdu	11.03	ATP,MSP,PS	Ferdigstilt
4.14	Designdokument	Masoud	11.03	ATP,MSP, HK	Ferdigstilt
4.15	Testplan	Masoud	11.03	MSP, TGF	Ferdigstilt
4.16	Øving til presentasjon2	Alle	11.03	Alle	Gjennomført
4.17	Presentasjon + etter møte	Alle	11.03	Alle	Gjennomført

4.18	Evaluering & Revidering	Alle	11.03	Alle	Gjennomført

Notat: *Elaboration 2 ble på en ekstra arbeidsuke, det er derfor vi i noen tilfeller stiller med samme aktivitet gjentatte ganger, men med et annet aktivitetsnummer. Den ekstra uken var satt av spesifikt til presentasjon.*

Overholdelse

Nevnbare gjennomførte oppgaver:

- **4.17:** Presentasjonen gikk som planlagt, stor forbedring, mer klar og mer teknisk.
- **4.10:** Opprettet kommunikasjonsprotokoll mellom RPI og RCU, får ikke helt det vi ønsker som utgang, men når vi finner ut hvordan vi kan konvertere/sortere dette i RPI så kan vi sende signalet videre. Dette betyr at det skal være mulig å kommunisere med systemet fra RCU510 via programmering i AIM, og oppgaven vår skal ha en mulig løsning.
- **4.07&4.15:** Designdokument er laget, sammen med en utvidet testplan.
- **4.06:** Har laget oppsett for hva og hvordan vi skal gå framover med programmering nå i neste iterasjon. Har også bestilt en ekstra komponent, til å gjøre modbus på arduino mulig.
- **4.09:** Vi har fått ett greit oppsett i AIM og vurderer å legge mindre vekt på dette framover.
- **4.12:** Diagrammer og bilder vi har laget gir en bedre forståelse for systemet vi setter sammen.
- **4.05:** Hjemmesiden er opprettet

Vi har bestilt en ekstra RPI som vi har valgt skal være på hexapoden, denne drar litt strøm, men vi håper på at vi får håndtert sensor data i større grad (eks. live feed via denne). I tillegg håper vi på å kunne flytte mye av den fremtidige logikken, inkludert wifi over i denne enheten.

Utfordringer og tiltak

Vi har gjennomført presentasjon 2, og har vært fornøyd med resultatet av framføringen denne gangen. Vi gikk gjennom det vi jobber med i den tekniske, og noe av det teoretiske delen. Tilbakemeldinger var i større grad positive, vi har tatt tiltak angående noen dokumenter som ble nevnt.

Det viser seg at programmering av modbus mottak i RPI er mer komplisert enn antatt, vi får ikke de resultatene vi får ved simulering. Det ble satt av litt tid til å finne en alternativ løsning, det ble funnet en arduino modbus kobling som vi har anskaffet. Dette gjør det mulig å løse modbus koblingen med C programmering, men er ett steg tilbake siden det blir ett ekstra kompliserende ledd ved bruker (RCU) siden av oppgaven. Da spørs det om vi videre skal ha direkte wifi fra arduino eller arduino som master med RPI slave og wifi, noe som vi trenger stepdown regulator og mer C programmering for. Uansett må vi få til modbus koblingen. Vi skal begynne å teste disse løsningene dualt fremover.

En annen utfordring er å se på sammenkoblingene i systemet, dette er foruten hvordan vi skal ta kontroll over hexapoden på en måte som gjør det mulig for ett datasystem som Rpi å kontrollere hexapoden. Først og fremst må vi opprette en type kommunikasjon mellom Rpi og arduino.

Vi skal se på sammenkoblinger for:

- Modbus: RCU-> RPI og RCU -> Arduino
- Functioncall: RPI -> Arduino
- Wifi: RPI -> RPI

Konklusjon

Vi er i rute med prosjektplanen, det viser seg at det vi driver med er en smule komplisert. Med 3 programmeringsspråk, 5 kontrollere og mange smådeler, samt minst 4forskjellige signaltyper og/eller sammenkoblinger så begynner prosjektet å se stort ut.

Men vi har så langt kommet oss i mål med det vi skal ha gjort ifølge fremdriftsplanen. Det er noen mangler i forhold til hva vi har forventet, men vi regner på å rette på dette framover. Vi har fått den oversikten vi trenger, vi har bestilt noen ekstra komponenter og skal begynne på integrasjon nå, og fortsette etter eksamensperioden.

5 Construction fasen

5.1 Construction 1

Periode: 11.03.2016 - 19.03.2016

5.1.1 Iterasjonsplan for Construction 1

Mål

Dette er en kort iterasjon som ligger rett etter andre presentasjon og rett før eksamensperioden vår. For å få mest mulig ut av denne perioden har vi satt den av til å redegjøre for sammensettingen av systemet som er nødvendig for å få fullført oppgaven. Det er sånn at vi begynner med programmering på ett bredt nivå der vi tar forskjellige type løsninger og snevrer inn etterhvert som vi ser hvilke som passer. Det har blitt gjort forhåndssøk og funnet mulige løsninger, som vi nå skal teste.

Det kommer til å bli mye prøving og feiling, test eller research på områder vi er usikre på.

Vi skal gjøre oss mest mulig ferdig med alt som har om AIM og legge det til siden til fordel for annen programmering.

Tabell 11 - Viktige gjøremål for iterasjon 1 (Construction 1 fasen)

Fase	Iterasjon	Start	Estimert ferdig	Ferdig	Viktige gjøremål
Construction	1	11.03	19.03	19.03	• Begynne integrering • Kontrollkommandoer ferdigstilt i AIM

Viktige gjøremål detaljert

Vi skal se på sammenkoblinger for:

- **5.05:** Modbus: RCU-> RPi:
  Vi jobber her med python programmering, vi har allerede fått RPI til å ta imot signalene, det vi jobber med er å få tolket og lagret de slik av vi får muligheten til å sende de videre.
- **5.06:** Functioncall: RPi -> Arduino
  Dette går ut på å se om vi kan kalle på ferdiglagde funksjoner i Arduino og få de til å bli utført via en RPI med i2c kobling mellom de to enhetene.
- **5.03:** Iterasjonsrapport for alle iterasjoner.

  Hvis det blir tid: Wifi: RPi(1) -> RPi(2): Opprette wifi kommunikasjon mellom to RPi enheter gjennom en router.

  Evt: RCU -> Arduino
  Om alternativet til modbus RCU -> RPI hvis python delen av oppgaven viser seg å være uløselig, eller for komplisert.

Tabell 12 - Detaljerte Aktiviteter for iterasjon 1 (Construction 1 fasen)

Aktivitet nr	Beskrivelse	Ansvarlig	Frist	Ressurs	Status
5.00	**Construction 1**				Avsluttet
5.01	Teknologidokumenter	Masoud	19.03	MSP,TGF	Endret
5.02	Designdokument	Masoud	19.03	MSP	Endret
5.03	Iterasjonsrapport	Aleksander	19.03	ATP	Ferdigstilt
5.04	RPi til Arduino komm.	Abdu	19.03	AS	Underarbeid
5.05	RCU til RPi / Arduino	Tor Gunnar	19.03	TGF, HK, PS	Underarbeid
5.06	Arduino functioncall	Masoud	19.03	AS,MSP	Underarbeid
5.07	AIM	Hanna	19.03	HK	Ferdigstilt

Overholdelse

- **5.03:** Iterasjonsrapporten er opprettet fra tidligere lagret informasjon om prosjektet
- **5.07:** AIM er så å si ferdigstilt for nå. Vi skal ikke jobbe mer med dette før vi vet resten av prosjektet går i mål. Det må påregnes at alt det vi har gjort må endres når kommunikasjonsnettverket er ferdigstilt og vi ser hva vi ønsker av signaler eller oppsett.
- Wifi løsningen ble ikke på begynt, men vi regnet heller ikke med dette.

Ellers har vi gjort de endringene vi syntes passet i dokumenter. Det meste av programmering er i startfasen og vi må tilegne oss litt mer kunnskap før vi kan løse problemene vi står ovenfor.

Utfordringer og tiltak

Det ble veldig lite tid til å se på Wifi løsningen denne uken, vi setter det av til neste iterasjon i fasen isteden. Vi ser nå, og har sett tidligere at det er noe mangel på kunnskap innad i gruppen på noen steder i programmeringsdelen. Vi brukte denne iterasjonen på å hente oss inn i dette feltet, men vi ser for oss at det kommer til å gå en del tid til å bli på gode eller friske opp programmeringskunnskapene våre.

Dette har ført til at vi har måtte gjøre en endring i prosjektplanen. I hovedsak var det tiltenkt å bruke denne uken på å integrere systemet, men det å integrere ett slikt system på én uke er veldig optimistisk. Vi har gjort endringer fra å integrere systemet, til å begynne på integrering, dette er også slik det var tiltenkt siden konstruksjonsfasen er der vi skal konstruere selve systemet.

Vi har enda for oss de aller største delene av systemet, som innebærer utvikling av kompliserte enkelt systemet som må settes sammen til ett enda mer komplekst system. Det er viktig at vi prater mye sammen framover slik at vi tilpasser delsystemene til hverandre etter hvert som de blir utviklet. Vi må tilegne oss kunnskapen som gjør det mulig å koble disse delene sammen og får de til å fungere sammen, dette krever at vi har god kommunikasjon fordi vi må kjenne til signaltypene og kommunikasjonsmetodene til de forskjellige delene som må tilpasses.

Konklusjon

Vi har fått sett litt på hva vi skal jobbe med framover nå i konstruksjonsfasen, alle har fått en ide om hva som kreves, og vi kommer til å jobbe videre utefra den endrede prosjektplanen. Vi ser at hvis vi følger denne de endringene som omfatter integreringen, så kommer vi i mål med prosjektet som planlagt.

5.2 Construction 2

Periode: 30.03.2016 - 13.04.2016

5.2.1 Iterasjonsplan for Construction 2

Mål

Vi må ferdigstille kommunikasjonsnettverket mellom RCU og hexapoden slik at vi kan begynne å teste implementasjonen til neste iterasjon. Vi har tidligere jobbet litt bredt på kunnskapene våre, vi skal fortsatt tilegne oss litt mer kunnskap. Men i hovedsak skal vi jobbe med problemer knyttet til overføringen og mottak av data, mellom signal typene i de forskjellige delsystemene.

Tabell 13 - Viktige gjøremål for iterasjon 2 (Construction 2 fasen)

Fase	Iterasjon	Start	Estimert ferdig	Ferdig	Viktige gjøremål
Construction	2	30.03	13.04	13.04	• Ferdigstille Modbus kommunikasjon • Implementere Wifi til modbus gjennom RPi

Viktige gjøremål detaljert

- **6.01:** Ferdigstille Modbus kommunikasjonen enten via RPI eller Arduino løsning. Slik at vi kan tolke, lagre, og videresende informasjonen til nesteledd i nettverket.
- **6.02:** Rpi til arduino kommunikasjonsprotokoll, vi må finne beste metode å overføre kommandoer fra RPi til arduino og kontrollere den.
- **6.03:** Hexapod Functioncalls blir å utrede, og programmere en kontroll protokoll inne i hexapoden slik at vi kan kontrollere den på en annen måte enn via PS2 kontrolleren. Denne typen interface inkluderer både enkle digitale funksjoner og mer avanserte analoge funksjoner.
- **6.04:** Implementere Wifi kommunikasjon sånn at det blir mulig å videreføre meldinger fra RCU kontrolleren gjennom RPi/Arduino til neste RPi på hexapoden

Alle disse aktivitetene er avgjørende for at prosjektet skal gå videre, uten hvilken somhelst del vil prosjektet stoppe opp under integrasjonen.

Tabell 14 - Detaljerte Aktiviteter for iterasjon 2 (Construction 2 fasen)

Aktivitet nr	Beskrivelse	Ansvarlig	Frist	Ressurs	Status
6.00	**Construction 2**				Avsluttet
6.01	RCU & Modbus Python	Tor Gunnar	13.04	TGF	Ferdigstilt
6.02	RPI til Arduino, Python	Abdu	13.04	MSP,AS,HK	Påbegynt
6.03	Hexapod Functioncall& Modbus Arduino	Aleks	13.04	ATP	Påbegynt/Ferdig
6.04	RPI til RPI Wifi kontroll, Python	Paul	13.04	PS	Ferdigstilt

Overholdelse

- Modbus er fullført, vi får verdier fra RCU som vi kan hente ut og bruke.
  (Modbus kommunikasjon er ett A krav, som nå er fullført i tide.)
- Wifi mellom RPI til RPI er opprettet og vi får RCU signaler gjennom, gjenstår småting. (Wifi er ett B krav som nå er blitt fullført som planlagt, i tide.)
- Scrappet Arduino modbus løsning da denne var halvferdig, fikk så vidt tak i kabling til løsningen før RPI modbus var ferdigstilt.
- Det er opprettet i2c kommunikasjon mellom RPI og arduino, utbedrer dette
  (Dette er en del av A krav for kommunikasjonsnettverket, fullført i tide)
- Arduino kontroll protokoll til hexapod er påbegynt, kan skrus av og på serielt så langt.

Utfordringer og tiltak

Utfordringene nå blir å kunne lagre, hente ut og gi riktig informasjon videre til Arduino gjennom i2c eller annen kobling fra RPi. Vi må også rette opp i småfeil og legge til funksjonalitet der det trengs. Videre blir det å se hvordan vi skal takle informasjonen som er mellom RPI og hexapod, og det å lage kontroll kode i hexapoden slik at den kan ta imot og utføre oppgaver.

Disse problemene går hånd i hånd, det kommer til å bli mye prøving og feiling i dette området. Etter endringene vi gjorde i Konstruksjon 1 fasen måtte vi gjøre tilsvarende små endringer videre i prosjektplanen for at ting skulle stå etter hvordan vi jobber.

Vi må regne med å støte på bugs framover, selv om delsystemer for så vidt fungerer betyr ikke dette at vi ikke trenger å forbedre eller endre på ting ettersom prosjektet utvikler seg videre. Det er mange ting som står igjen men det begynner å løsne i noen ledd, og det ser lysere ut.

Konklusjon

Vi er godt på vei, har gjort ett ganske stort sprang i kommunikasjonsnettverket da vi har jobbet på dette lenge. Gjennombruddet med brukbare modbus signaler fra RCU gjør at vi ser lysere på oppgaven da dette lenge har vert ett mørkt punk i utviklingen av systemet. Vi har kommet relativt langt i prosjektet nå og det som står igjen er å programmere hexapoden og den trådløse delen til mottak av kommandoene fra RCU.

5.3 Construction 3

Periode: 13.04.2016 - 27.04.2016

5.3.1 Iterasjonsplan for Construction 3

Mål

Fokuset vårt blir på å integrere de mest komplekse delene av systemet. Få kommunikasjonsnettverket til å fungere fra RCU til Hexapoden på den andre siden. Integrasjonen og funksjonaliteten av kommunikasjonsnettverket vårt inneholder alle våre A krav, og er den viktigste delen av oppgaven. Dette er funksjonaliteten som legger grunnlaget for at en annen gruppe skal kunne ta over vårt arbeid og jobbe videre med den autonomiske delen av systemet. Dette er også ansett som den mest kompliserte delen fordi det ikke er noe sikkerhet om at dette i det heletatt er mulig med den hardwaren vi har til rådighet. Vi har kommet så langt som vi har gjort med god planlegging og mye hardt arbeid. Vi skal nå komme oss i mål med det viktigste vi var ilagt fra oppdragsgiver.

Det må nevnes at dette blir eventuelt første sammenkobling av delsystemene og full funksjonalitet er ikke påregnet uten mer testing og programmering. Så i tillegg skal vi fortsette med å lage og tilpasse kontrollkommandoer i hexapoden slik at vi kan ta imot beskjeder og utføre oppgaver i suksesjon, via både direkte styring og pakkestrying ved sending av flere kommandoer fra RCU brukersiden. I tillegg skal det legges inn livefeed fra hexapod via wifi slik at bruker kan hvor hexapoden går, dette vil oppfylle C krav for sensorer, og vil gi hexapoden den mest nødvendige feedbacken til å fungere som en ROV sensor plattform. Vi begynner på C krav litt tidligere sammen med testingen av systemet siden arbeidsoppgavene begynner å bli mer og mer spesifikke, og vi må fordele arbeidsoppgaver i forhold til kompetansen i området for mer effektivitet.

Tabell 15 - Viktige gjøremål for iterasjon 3 (Construction 3 fasen)

Fase	Iterasjon	Start	Estimert ferdig	Ferdig	Viktige gjøremål
Construction	3	13.04	27.04	26.04	• Testing • Hexapod og Rcu nettverkssammenkobling

Viktige gjøremål detaljert:

- **7.01:** Wifi, mottak og sending av modbus til Arduino
  - Dette går ut på å ta imot informasjonen man får over modbus kommunikasjonsprotokollen og håndtere den riktig inne i RPI. Dette krever python programmering, signalene må bli tatt imot, og tolket. Vi må så opprette en internettkobling gjennom en router, sende informasjonen gjennom wifi kommunikasjonen der en annen RPI tar imot og bevarer signalet. Begge enhetene må programmeres slik at de kan sende informasjon fram og tilbake.
  - Nettverket skal fungere fra RCU->RPi(1)->over Wifi->RPi(2)->Hexapod, og styre hexapden. (Dette krevet at hexapoden er programmert til å mota seriell kobling og håndtere/tolke disse signalene/kommandoene som funksjoner/bevegelser)
- **7.02&7.05:** Opprette kommunikasjon mellom RPI&ardino, for å så sammenkoble de.
  - Vi må finne ut om vi skal bruke i2c kobling eller miniusb kobling for at de 2 enhetene skal kunne kommunisere på en måte som får enhetene til å respondere slik vi ønsker. Det må også bli tatt hensyn til hva som er mest robust i forhold til at dette skal sitte på en hexapod i bevegelse.
  - Det må muligens også gjøres noen endringer i kommunikasjonshastighet osv. for at systemet skal fungere optimalt. Uforutsette hendelser er antatt her siden dette er den siste sammenkoblingen mellom bruker og plattform siden av systemet.
- **7.03&7.06:** Omprogrammering av hexapoden, analoge og digitale funksjoner, samt bevegelser og mottak av seriell kommunikasjon.
  - Det skal opprettes seriell kommunikasjonsmuligheter slik at dataenheter (RPi) kan gi hexapoden kommandoer.
  - Bevegelsesmønstret/hastighet skal være tilpasset bruken av plattformen med sensorer og som ROV, ingen opphenging, hard gange etc.
  - Hexapoden skal ha fungerende analoge og digitale funksjoner, og kunne lese inputs fra RPI og tolke de med riktig inputdelay/hastighet for å få jevn, konstant bevegelse.
  - Alle hexapod bevegelser og funksjoner skal være tilgjengelig og funksjonelle.
- **7.07:** Sensorer, begynne å se på forskjellige typer sensorer, først og fremst livefeed kamera.

Tabell 16 - Detaljerte Aktiviteter for iterasjon 3 (Construction 3 fasen)

Aktivitet nr	Beskrivelse	Ansvarlig	Frist	Ressurs	Status
7.00	**Construction 3**				Avsluttet
7.01	Wifi, mottak og sending av modbus til Arduino	Tor Gunnar	27.04	TGF,MSP,PS	Ferdigstilt
7.02	RPI til Arduino, Python	Abdu	27.04	AS,HK	Ferdigstilt
7.03	Hexapod omprogrammering	Aleks	27.04	ATP	Ferdigstilt
7.04	Iterasjonsrapport	Aleks	27.04	ATP	Oppdatert
7.05	Koble RPI til Hexapod	Masoud	27.04	MSP, TGF, PS, AS	Ferdigstilt
7.06	Analoge Hexapod bevegelser/funksjoner	Aleks	27.04	ATP	Ferdigstilt
7.07	Sensorer	Hanna	27.04	HK, TGF	Påbegynnt

Overholdelse

Vi har fullført implementasjonen av RCU til Hexapod kommunikasjonsnettverket. Dette er blitt testet og fungerer som planlagt i oppgaveteksten. Vi kan kontrollere både direkte som en «håndkontroller», og via pakkestyring, sending av flere kommandoer i pakker med tidsvariabel som bestemmer lengden på signalet/kommandoen og funksjonen som blir utført i hexapoden, pakken blir utført fortløpende. Ved pakkestyring kan vi unngå å bli påvirket av eventuell forsinkelse som oppstår i nettverket, selv om dette er ikke-eksisterende akkurat nå kan det bli ett problem over lengre distanser med dårligere nettverkskobling.

Systemet er laget slik at det er RPI(2), oppe på hexapoden som tar kommandoer, for å så sende de videre, dette betyr at så lenge vi får tilgang til kontroll signalene på en annen plattform, så kan vi kontrollere den også. Hexapoden er derfor utbyttbar slik oppdragsgiver ønsket. Hexapoden har fått lagt opp signalene slik at det er lett å ta kontroll over den, det samme må eventuelt gjøres med den nye plattformen.

- **7.01:** Wifi, mottak og sending av modbus til Arduino
  - Wifi kommunikasjon er opprettet mellom RPi(1) og RPi(2)
  - RCU kan nå kommunisere gjennom RPiene.
- **7.02&7.05:** Opprette kommunikasjon mellom RPI&arduino, for å så sammenkoble de.
  - Seriell kommunikasjon er laget mellom RPi og arduino
  - Arduino kontrolleres av RPi
- **7.03&7.06:** Omprogrammering av hexapoden, analoge og digitale funksjoner, samt bevegelser og mottak av seriell kommunikasjon.
  - Arduino interface for RPi kontroll er laget
  - RCU kan derfor sende modbus signaler til RPi(1) gjennom wifi til RPi(2) og kontrollere hexapoden.
- **7.07:** Sensorer, begynne å se på forskjellige typer sensorer, først og fremst livefeed kamera.
  - Kamera modul er testet

Utfordringer og tiltak

Kommunikasjonsnettverket fungerer, men det er fortsatt mye for oss å ta tak i. Først og fremst må vi forbedre brukerfunksjonaliteten, og implementere sensorer slik at plattformen faktisk er brukbar som ett ubemannet fartøy. Kamera er den første sensoren vi skal gå for, valget vårt av en datamaskin(RPI) på hexapoden skal gjøre dette mulig.

Ved pakkeløsning kan vi forsette å programmere hexapoden, få den til å reagere autonomt i forskjellige situasjoner og ta valg for seg selv. Dette er ikke en del av krav for oppgaven vår nå i år, men det er mulig vi får påbegynt å autonomisere noe. Ettersom avanserte hexapod bevegelser er en del av oppgavebeskrivelsen kan vi tøye definisjonen som mer enn de analoge bevegelsesmønstrene.

Men det er alt for lite plass i BotBoardet på Hexapoden til at vi kan programmere noe mer. Flashminnet er på 87% og dynamisk minne er på 93%, vi må bytte arduino platform til en sterkere enhet, dette krever porting for å gjøre koden kompatibel med en annen arduino. Dette problemet må løses først, og er av større nytte dersom neste gruppe etter oss skal kunne forsette å utvikle hexapodens egne funksjoner. Autonomisasjon burde flyttes opp i logikken til RPI, men dette er ikke vår prioritet, siden nettverket forbedres i RPI framover.

I tillegg må lage ett autokjør script for RPI logikken på hexapoden, slik at den starter automatisk sammen med arduino og servokort ved «powerup», dette er ikke standard. Vi må forbedre exteriøret til hexapoden, forbedre kommunikasjonsnettverket, se på AIM bruker interfacet, og implementere livefeed.

Vi har planlagt å trappe opp arbeidsmengden framover mot siste deadline for prosjektet 23.05.2016.

Konklusjon

Vi har nå ved hardt og effektivt arbeid oppnådd A og B kravene til systemet, de er testet og har D feil eller lavere (se testplan), altså eventuelt småforbedringer kan gjøres. Hittil har vi hele veien blitt ferdig med arbeidsoppgavene i forhold til prosjektplanen og det viser seg at «Unified Process» prosjektmodellen har passet oss perfekt. Vi har blitt veldig komfortable med fremdriftsplanen og måten vi arbeider på gjør at vi leverer som planlagt gang på gang.

Det hjalp med en uke research på starten av konstruksjonsfasen før påsken, dette forberedte oss på hva som kom til å komme. Vi er hittil veldig fornøyde med arbeidet vi har gjort. Dette i og med at systemet består av mange forskjellige kompliserende deler som virker sammen integrert i ett system. Samt at det er flere programmeringsspråk og flere programmer vi må bruke for konstruksjonen av systemet.

6.1 Transition 1

Periode: 27.04.2016 – 12.05.2016

6.1.1 Iterasjonsplan for Transition 1

Mål

Vi skal gjøre systemet klart til at oppdragsgiver kan ta over. Når denne iterasjonen er over er det nesten bare gjenstående dokumentasjon, blant annet brukermanual. Fokuset er på å legge til den siste funksjonaliteten som kan være nyttig for sluttbruker, samt få systemet til å bli relativt presentabelt.

For å passe på at systemet er i orden så må vi gjøre en del tester og jobbe videre med den siste delen av implementasjon. Den siste delen består av å se videre på C krav.

Tabell 17 - Viktige gjøremål for iterasjon 1 (Transition 1 fasen)

Fase	Iterasjon	Start	Estimert ferdig	Ferdig	Viktige gjøremål
Transition	1	27.04	12.05	12.05	• Test og implementering • Implementere C og B krav

Viktige gjøremål detaljert

- **8.01:** RPI autorun script
  - Få RPIene til å starte de nødvendige programmene automatisk ved startopp.
- **8.02&8.07&8.12:** Porting av kode til Mega, programmering av tracking og ultrasoniske sensorer
  - Dette må bli utført i rekkefølge der Mega trengs å omprogrammeres først, dette innebærer endring i biblioteker, pin nummer, koblinger og funksjoner som er spesifikke til Atmega2560 prosessoren.
  - Tracking gjelder å lære pixycam5 å følge objekter, for så å hente informasjon om objektstørrelse og innlært signatur slik at hexapoden kan programmeres til å følge eller reagere på innlærte signaturer.
  - Ultrasoniske er enten programmering av generell autonomisk autopilot eller sikkerthet «stopp før vegg» funksjon.
- **8.03&8.08:** Hexapod Exteriør design og koffert beholdere.
  - Må finne passende beskyttelse og transportbeholder for systemet
  - Hexapod må være koblet ryddig og lagt opp så det ikke er altfor mye kabelrot.
  - Eventuelle showfaktor med ledlys må kobles og programmeres.
- **8.05:** Implementere livefeed mellom RPIene
  - Legge dette inn i hovedkoden til RPI(2), gjøre de endringene som trengs og få testet kameraet via wifi.
- **8.06&8.011:** AIM bruker interface
  - Designe det nye operatørside konseptet og utføre tilsvarende konfigurering i AIM.
  - Opprette bedre brukerfunksjonalitet for bruker av AIM programvaren.
  - Tilpasse det nye konseptet til funksjonaliteten på RPi side.
- **8.09&8.10:** AIM – Hexapod (Start og stop funksjon) og forbedring av funksjonalitet:
  - Dette er til å overskrive pakker som allerede er sendt slik at bruker kan avslutte bevegelses sekvenser.
  - Må lage funksjon i AIM, og så må funksjonen programmeres følgende i de 2 RPIene.
  - Forenkling og endring i kommunikasjonsnettverket
- **8.13:** Iterasjonsrapport
  - Utfylle fasegjennomføringen
- **8.14:** Batteriløsninger
  - Finne løsning slik at roboten kan kjøre på 11.1V lipo batterier uten skade.
  - Trenger varslingssystem for å advare mot skadelig utladning.
  - Systemet må få nok strøm, men samtidig ikke for mye spenning, regulering av spenning uten å strupe strøm trengs.
  - Støyfilter må vurderes.

Tabell 18 - Detaljerte Aktiviteter for iterasjon 1 (Transition 1 fasen)

Aktivitet nr	Beskrivelse	Frist	Ressurs	Status
8.00	**Transition**			Avsluttet
8.01	RPI autorun script	12.05	MSP	Ferdigstilt
8.02	Porting av hexapod kode til Arduino Mega og pixycam tracking eller ultrasoniske sensorer	12.05	ATP,	Mega Ferdigstilt
8.03	Hexapod exterior design + koffert	12.05	MSP,AS,ATP	Ferdigstilt
8.04	Forbedring av funksjonalitet i kommunikasjonsnettverket. (AKA bug squashing)	12.05	TGF, PS	Ferdigstilt
8.05	Implementere livefeed RPI	12.05	PS	Ferdigstilt
8.06	AIM, bruker interface	12.05	HK	Påbegynnt
8.07	Pixycam tracking	12.05	ATP	Mangler komponent
8.08	Hexapod exterior design	12.05	MSP,AS, PS	Ferdigstilt
8.09	AIM - Hexapod (Start og stop funksjon)	12.05	TGF,PS,	Ferdigstilt
8.10	Utbedre system funksjonaliteten	12.05	TGF,PS	Ferdigstilt
8.11	AIM, bruker interface	12.05	HK	Ferdigstilt
8.12	Ultrasoniske sensorer	12.05	AS	Påbegynnt
8.13	Iterasjonsrapport	12.05	ATP	Ferdigstilt
8.14	Batteri løsninger	12.05	ATP	Bestilt

Overholdelse

- **8.01:**
  - Autorun skript har blitt opprettet for RPIene, de starter automatisk ved strømtilgang.
- **8.02&8.07&8.12:**
  - Atmega328p har blitt byttet med Atmega2560 (10xminne, 4xdynamisk minne, 2xprosessor kraft)
  - Mangel på komponent gjør at vi kun har startet på koding av PixyTracking uten fysisk kobling eller test.
  - Ultrasonisk distansesensor for øyer er laget, autonomisk oppførsel modus påbegynnt.
- **8.03&8.08:**
  - Det har blitt anskaffet polstret oppbevaring for Hexapod og RCU
  - Hexapod er blitt ryddet opp med bedre kobling
  - Led øyer er installert
- **8.05:**
  - Livefeed sensor er implementert og fungerer over wifi
- **8.06&8.011:**
  - Ny brukerfunksjonalitet for AIM interface tilpasset til funksjonaliteten i RPI
- **8.09&8.10:**
  - Stopp og pause er innlagt i brukerfunksjonaliteten
  - Start er automatisk ved kommando mottak, skrus av automatisk etter en stund uten mottatt kommando
  - Annen funksjonalitet er forbedret
  - VNC gjør det mulig å programmere Hexapodens Arduino og RPI(2) fra brukersiden uansett distanse med wifi.
- **8.13:**
  - Utført
- **8.14:**
  - UBECs er bestilt for regulering av lipo batterier, har sikkerhets varsel og støyfilter.

Vi har nå ved hardt og effektivt arbeid oppnådd A og B kravene til systemet, vi har også implementert ett C krav, alt er testet og vi har oppnådd D feil eller lavere (se testplan). Vi skal fortsette med testing av systemet og videre implementering av C krav hvis det blir tid til overs. Den generelle funksjonaliteten er god, men vi ser at det fortsatt er mye vi ønsker å gjøre med systemet.

Utfordringer og tiltak

På grunn av mangel på komponenter har det vert vanskelig å implementere flere C krav, og uten disse komponentene har det også vert vanskelig å ferdigstille noe fast exteriør design siden disse trengs å implementeres.

Vi har akkurat byttet til Atmega2560 og for å drive dette kortet, på grunn av mangel på enkle koblingspunkter så venter vi på regulatorer for kraftigere strømkilder. Vi driver systemet enn-så-lenge på en bypass løsning.

Forhåpentligvis får vi tid til å gå tilbake å se på programmering av noen mer komplekse sensorer, men dette kommer helt ann på tid, og kan hende blir utelatt fra dokumentasjon selv om det blir implementert. Dokumentasjon er det neste vi må ta for oss.

Konklusjon:

Vi har hele veien blitt ferdig med arbeidsoppgavene i forhold til prosjektplanen og det viser seg at «Unified Process» prosjektmodellen har passet oss perfekt. Vi har blitt veldig komfortable med framdriftsplanen og den framstillingen vi har fått via UseCases og illustrasjoner av systemet gjør at vi vet hva vi sikter til av funksjonalitet. Dette har gjort at vi leverer som planlagt gang på gang.

Sen levering og lite tid til programmering av ekstra sensorer gjør at disse nylig er satt under arbeid.

6.2 Transition 2

Periode: 12.05.2016 – 23.05.2016

6.2.1 Iterasjonsplan for Transition 2

Mål

Ferdigstille dokumentasjonen som omhandler systemet vi har konstruert, og levere hovedrapporten til sensorer og veiledere for prosjektet.

Tabell 19 - Viktige gjøremål for iterasjon 2 (Transition 2 fasen)

Fase	Iterasjon	Start	Estimert ferdig	Ferdig	Viktige gjøremål
Transition	2	12.05	23.05	23.05	• Levere hovedrapport

Viktige gjøremål detaljert

- **9.01:** Designdokument
  - AIM
  - RPI(1)
  - RPI(2)
  - Hexapod
- **9.02:** Plakat
  - Visuell framstilling av prosjektet
- **9.03:** Pixycam tracking
  - Lese gjenkjente signaturer og få hexapod til å respondere
- **9.04:** Autonomous
  - Bruk av ultrasonisk sensor til å få hexapod til å manuvrere automatisk uten å kollidere med objekter
- **9.05:** Brosjyrer/klistremerker
  - Reklame brosjyrer for produktet
  - Logo og innholds merker til kofferter
- **9.06:** Iterasjonsrapport
  - Fullstendig iterasjonsrapport
- **9.07:** Testrapport
  - IAT og FAT
- **9.08**: Teknologidokument
  - Kommunikasjonselementer
    - Modbus (RCU til Rpi)
    - Wifi (Rpi til Rpi)
    - Serial (Rpi/data)
    - RF (Ps2)
- **9.09:** Brukermanual
  - Enkel brukermanual med nødvendiginformasjon for oppstart og hyppig bruk av system.
- **9.10:** Kravsporing til aktiviteter
  - Bevise sammenheng mellom aktiviteter og krav vi har ferdigstilt
- **9.11:** Forbedring koding RPI(2)
  - Gjennomgang og forbedring
- **9.12:** Økonomidokument
  - Innføring av alle kostnader medfølgende prosjektet
  - Nødvendig informasjon om leverandører

Tabell 20 - Detaljerte Aktiviteter for iterasjon 2 (Transition 2 fasen)

Aktivitet nr	Beskrivelse	Frist	Ressurs	Status
9.00	**Transition 2**			Avsluttet
9.01	Designdokument	23.05	ATP,MSP,TGF,HK,PS	Ferdigstilt
9.02	Plakat	23.05	ATP,PS	Ferdigstilt
9.03	Pixycam tracking	23.05	ATP	Underarbeid
9.04	Autonomous	23.05	AS	Underarbeid
9.05	Brosjyrer/Klistremerker	23.05	ATP,MSP	Underarbeid
9.06	Iterasjonsrapport	23.05	ATP	Ferdigstilt
9.07	Testrapporter	23.05	Alle	Ferdigstilt
9.08	Teknologidokument	23.05	TGF	Ferdigstilt
9.09	Brukermanual	23.05	ATP,TGF,HK,PS	Ferdigstilt
9.10	Kravsporing til aktiviteter	23.05	HK	Underarbeid
9.11	Forbedring koding RPI(2)	23.05	PS	Ferdigstilt
9.12	Økonomidokument	23.05	MSP	Ferdigstilt

Overholdelse:

- **9.01:** Designdokument
  - Ferdigstilt
- **9.02:** Plakat
  - Ferdigstilt
- **9.03:** Pixycam tracking
  - Dette er underarbeid, akkurat fått enheten og vi mangler litt strømkapasitet til å dra den ordentlig for testing, dette skal ordnes så fort vi får de nye regulatorene.
- **9.04:** Autonomous
  - Endel bugs i koden, men vi ser noe fram og tilbake bevegelse ved ultrasonisk input sensor, mer tid vil kanskje gjøre dette mulig.
- **9.05:** Brosjyrer/klistremerker
  - Må få respons fra trykker angående bestilling
- **9.06:** Iterasjonsrapport
  - Fullstendig bortsett fra transition 3 som ikke kan medfølge i denne rapporten.
- **9.07:** Testrapport
  - Tester er utført og utfylt.
- **9.08**: Teknologidokument
  - Kommunikasjonselementer er vedlagt i teknologidokument.
- **9.09:** Brukermanual
  - Laget.
- **9.10:** Kravsporing til aktiviteter
  - Ser fortsatt på om dette er nødvendig
- **9.11:** Forbedring koding RPI(2)
  - Småforbedringer i koden
- **9.12:** Økonomidokument
  - Ferdigstilt

Utfordringer og tiltak:

- Vi mangler fortsatt regulatorer for at systemet skal fungere optimalt, alternativet er å koble systemet tilbake til Botboarduino og kjøre en eldre kode midlertidig.
- Autonomous mode fungerer relativt greit, strømmangler fører til at systemet henger, er noen bugs som må fjernes før det er godt nok til bruk.
- Pixytracking er påbegynnt, men uten regulatorer vil ikke dette fungere. Mangler testing, foruten gjenkjenning av signaturer og innhenting av blokkinformasjon som er utprøvd og gjennomført.
- Systemet fungerer, men hadde vi hatt regulatorene ville vi hatt mulighet til mer testing og videreutvikling.

Konklusjon:

Vi har gjennomført de oppgavene som har vert mest kritisk for oss i denne iterasjonen. Dokumentasjonen er dermed ferdigstilt i den grad vi har ønsket. Vi har en del tid igjen til å koble på

tekniske detaljer som vi enda venter på. Om vi skal bruke mer tid på programmering blir vurdert framover i neste iterasjon. Fram til nå så er vi fornøyd med arbeidsinnsatsen og ser at vi stiller med ett ferdig produkt som bestilt av Kongsberg Maritime.

6.3 Transition 3

Periode: 24.05.2016 – 01.06.2016

6.3.1 Iterasjonsplan for Transition 3

Mål:

Lage en presentasjon som omhandler prosjektet og gjenspeiler kvaliteten i systemet som vi har konstruert.

En presentasjon skal være laget for veiledere, sensorer og publikum på Krona. To andre presentasjoner skal være laget for KM, en for sponsoravdeling og en for hele KM.

Tabell 21 - Viktige gjøremål for iterasjon 3 (Transition 3 fasen)

Fase	Iterasjon	Start	Estimert ferdig	Ferdig	Viktige gjøremål
Transition	3	24.05	01.06	02.06	• Presentasjon 3

NB! Grunnet innlevering av bacheloroppgaven, så er ikke detaljerte aktiviteter for iterasjon 3 (Transition 3 fasen) med i dette dokumentet. Jobben for de to ukene videre er forberedelse til siste og tredje presentasjon.

KRAVSPESIFIKASJON

PROSJEKT	K-Spider		
OPDRAGSGIVER	Kongsberg Maritime AS		
UTFØRT VED	Høgskolen i Sørøst-Norge, avd. Kongsberg		
REVISJON	2.0		
MEDLEMMER	Aleksander Tokle Poverud, Masoud Shah Pasand, Tor Gunnar Finnerud, Hanna Kåsastul, Paul Knutson Sæther, Abdurahman Senkaya		
Dokumenthistorikk	**REVISJON**	**UTGITT**	**BESKRIVELSE**
	1.0	28.01.2016	Første utgave
	2.0	19.05.2016	Andre utgave

INNHOLDSFORTEGNELSE

1 DOKUMENTHISTORIE 3

2 INNLEDNING 3

3 KRAV 4

3.1 Rammekrav 5
3.2 Funksjonelle krav 6
3.3 Maskin og programvarekrav 7
3.4 Dokumentasjonskrav 7
3.5 Designkrav 7

TABELLER

Tabell 1 - Dokumenthistorie 3
Tabell 2 - Kravprioritet 4
Tabell 3 - Kravkategorier 4
Tabell 4 - Rammekrav 5
Tabell 5 - Funksjonelle krav 6
Tabell 6 - Maskin- og Programvarekrav 7
Tabell 7 - Dokumentasjonskrav 7
Tabell 8 - Designkrav 7

1 DOKUMENTHISTORIE

Tabell 1 - Dokumenthistorie

VERSJON NR:	DATO ENDRET	ENDRET AV	GODKJENT AV	BESKRIVELSE
0.1	15.01.2016	Hanna	Masoud	• Oppretting av dokumentet
1.0	28.01.2016	Hanna	Masoud	• Godkjent og utgitt
1.1	11.05.2016	Masoud	Hanna	• Lagt inn kapittel 3.4 (dokumentasjonskrav) • Endret tabelloppsett
1.2	13.05.2016	Masoud	Tor Gunnar	• Lagt inn designkrav i kapittel 3.5
2.0	19.05.2015	Masoud	Aleksander	• Godkjent og utgitt

2 INNLEDNING

Dette dokumentet beskriver kravene som er utarbeidet av prosjektgruppa, på bakgrunn av krav som oppdragsgiver Kongsberg Maritime (KM) v/Qui-Huu Le-Viet har fremsatt. Kravspesifikasjonen er en beskrivelse av hvordan det ønskede produktet skal se ut og hvilke funksjoner produktet skal oppfylle, og spesifikasjonen skal legge vekt på å beskrive ønskede funksjoner/ytelser for det som skal anskaffes.

Kravene i dette dokumentet er utredet ut ifra visjonsdokumentet, kapittel 4. Spesifikasjonen danner grunnlaget for testspesifikasjon som prosjektgruppa skal utarbeide.

3 KRAV

I dette kapittelet skal prosjektgruppa presentere de ulike kravene som fremstilles, samt en definisjon av hvordan kravene er presentert. De ulike kravene er delt inn i forskjellige kategorier og de har forskjellig prioritering. Noen krav er mer avgjørende enn andre. Kravene er også fordelt i kategorier.

Tabell 2 - Kravprioritet

PRIORITERT	BESKRIVELSE
A	Absolutte krav som er nødvendige for at prosjektet skal være fullført.
B	Krav som er viktige, men systemet kan fungere uten.
C	Krav som er mindre viktige, og skal oppfylles kun hvis gruppen får tid.

Kravene er fordelt i tre kategorier, som gir en bedre forståelse av hele utviklende systemet.

Tabell 3 - Kravkategorier

KATEGORIER	BESKRIVELSE
RAMMEKRAV	Krav som er satt som rammebetingelser til oppgaven.
FUKSJONELLE KRAV	Krav til funksjoner systemet skal utføre.
MASKIN- OG PROGRAMMEKRAV	Krav til bruk av maskin- og programvare.

3.1 Rammekrav

Tabell 4 - Rammekrav

ID	PRIORITET	FREMSATT DATO	FREMSATT AV	KRAV
KR01	A	25.11.15	Kongsberg Maritime	Det skal bygges en mobil sensor plattform (robot) som kan operere i vanskelig terreng.
KR02	A	25.11.15	Kongsberg Maritime	Roboten skal kunne utføre bevegelser.
KR03	A	25.11.15	Kongsberg Maritime	Roboten med kontroll systemet skal være utført slik at den skal kunne vises til kunder og evt. besøkende.
KR04	A	25.11.15	Kongsberg Maritime	Det skal bevise noe av hva kontroll systemet til Kongsberg Maritime kan utføre.
KR05	C	25.11.15	Kongsberg Maritime	Roboten skal kunne utføre avanserte bevegelser ved hjelp av misjon sensorer.

3.2 Funksjonelle krav

Tabell 5 - Funksjonelle krav

ID	PRIORITET	FREMSATT DATO	FREMSATT AV	KRAV
KF01	A	25.11.15	Kongsberg Maritime	Bevegelsene til roboten skal være koordinert.
KF02	A	25.11.15	Kongsberg Maritime	Det skal være utvidet kommunikasjon mellom AIM og roboten via RCU.
KF03	B	25.11.15	Kongsberg Maritime	Roboten skal kunne fjernstyres manuelt via ekstern kontroller.
KF04	C	25.11.15	Kongsberg Maritime	Roboten skal kunne gi feedback til operatørstasjon.
KF05	C	25.11.15	Kongsberg Maritime	Roboten skal kontrolleres trådløst via WiFi.

3.3 Maskin og programvarekrav

Tabell 6 - Maskin- og Programvarekrav

ID	PRIORITET	FREMSATT DATO	FREMSATT AV	KRAV
KM01	A	25.11.15	Kongsberg Maritime	Det skal installeres en AIM operatørstasjon for kontroll av roboten.
KM02	A	25.11.15	Kongsberg Maritime	Roboten skal kontrolleres av Kongsberg sanntidskontroller RCU510.
KM03	A	25.11.15	Kongsberg Maritime	Det skal benyttes KM kontrollsystem AIM7 for styring.
KM04	C	25.11.15	Kongsberg Maritime	Det skal implementeres feedback sensorer.
KM05	C	25.11.15	Kongsberg Maritime	Det skal implementeres misjon sensorer.

3.4 Dokumentasjonskrav

Tabell 7 - Dokumentasjonskrav

ID	PRIORITET	FREMSATT DATO	FREMSATT AV	KRAV
KD01	A	11.03.16	Kongsberg Maritime	Brukermanual til systemet.

3.5 Designkrav

Tabell 8 - Designkrav

ID	PRIORITET	FREMSATT DATO	FREMSATT AV	KRAV
KX01	B	29.04.16	Kongsberg Maritime	En enkel løsning for å frakte utstyret til eventuelle messer.

Kravendringshåndtering

PROSJEKT	K-Spider
OPDRAGSGIVER	Kongsberg Maritime AS
UTFØRT VED	Høgskolen i Sørøst-Norge, avd. Kongsberg
REVISJON	2.0
MEDLEMMER	Aleksander Tokle Poverud, Masoud Shah Pasand, Tor Gunnar Finnerud, Hanna Kåsastul, Paul Knutson Sæther, Abdurahman Senkaya

Dokumenthistorikk	**REVISJON**	**UTGITT**	**BESKRIVELSE**
	1.0	02.03.2016	Første utgave
	2.0	20.05.2016	Andre utgave

INNHOLDSFORTEGNELSE

1 DOKUMENTHISTORIE 3

2 INNLEDNING 3

3 Ønskede endringer fra oppdragsgiver 4

3.1 Metode for ønskede endringer 4

4 Eventuelle svikt med prosjektet 5

5 Ønskede endringer fra prosjektgruppa 5

6 Godkjenningsprosessen 5

7 Oversikt over endringssaker (Loggføring) 5

8 Arkivering 5

9 Referanser 6

TABELLER

Tabell 1 - Dokumenthistorie 3

FIGURER

Figur 1 - Prosess for kravendringer 4

1 DOKUMENTHISTORIE

Tabell 1 - Dokumenthistorie

VERSJON NR:	DATO ENDRET	ENDRET AV	GODKJENT AV	BESKRIVELSE
0.1	11.02.2016	Masoud	Aleksander	• Dokumentet ble opprettet
1.0	02.03.2016	Masoud	Aleksander	• Oppdatert innholdet og lagt inn figur 1.
1.1	11.05.2016	Masoud	Aleksander	• Endret oppsett i dokumentet
2.0	20.05.2016	Masoud	Aleksander	• Godkjent og utgitt

2 INNLEDNING

Dette dokumentet er laget for å gjennomgå nøkkelhinder for prosjektsuksess, det å endre kravspesifikasjoner under prosjektlivssyklusen og utforske hvordan vi kan håndtere endringene på en agil måte. Endringer er uunngåelig og et ønske om endring er en naturlig del av et prosjekt. Det kan være tekniske problemer som medfører endringer av kravene, og i dette tilfellet så vil prosjektet initiere til en endring mot kunden. Et annet tilfelle kan være at kunden vil ha ekstra funksjonaliteter og ber da om endringer.

Det er viktig å ha en formell prosess for å fremlegge, evaluere og godkjenne endringer og de resulterende implikasjonene det har på prosjektet. Hvis ikke disse endringene er håndtert ordentlig fra start kan man forvente tidsplanforskyvninger, ekstra kostnader og minsket prosjektkvalitet.
En viktig del av prosjektet er å kontrollere prosjektkrav. Utviklingsprosjekter er veldig utsatt for disse problemene, da prosjektet vårt er en god blanding av konstruksjon og utvikling, så vi må ta godt hensyn til og ha en god plan for eventuelle endringer.

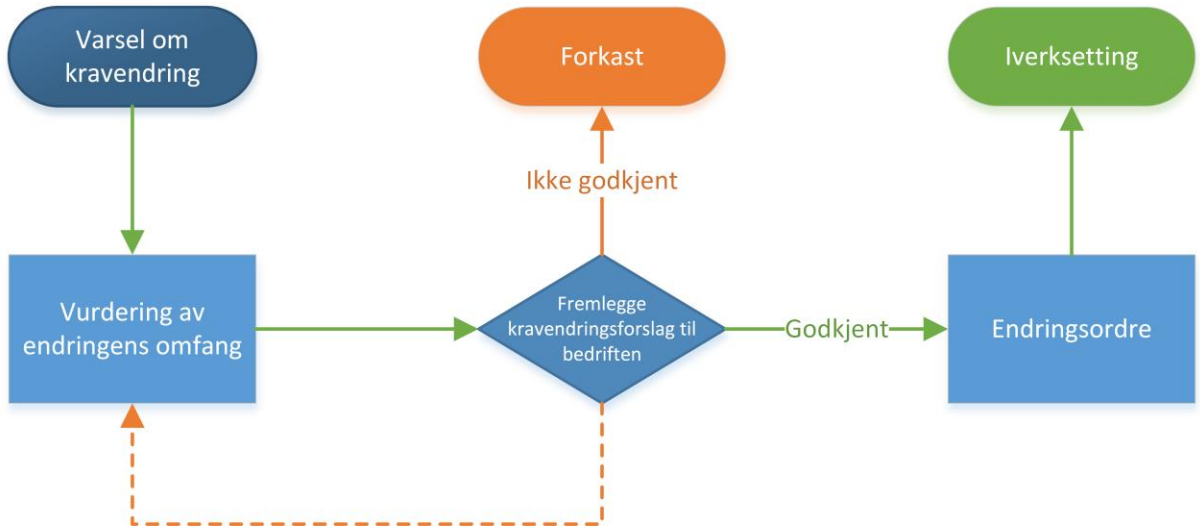
3 Ønskede endringer fra oppdragsgiver

Når oppdragsgiveren ønsker en kravendring eller vil ha ekstra funksjonalitet, skal det behandles separert fra de pågående prosjektaktivitetene. Dette skal gjøre det mulig for oss å opprettholde omfanget av prosjektet og sørge for at den holder seg innenfor tidsplan og budsjett. Med en eller flere nye kravforespørsler kan det ha betydelig innvirkning på prosjektet. Vi har valgt å ha endringsønsker adskilt fra løpende oppgaver, slik at vi er i stand til å identifisere hvilke forespørsler som kan gjennomføres.

Å holde prosjektoppgaver og kommende forespørsel separert gir bedre oversikt og det blir lettere for prosjektlederen å gjenkjenne og spore endringer i prosjektet. Kunden vil være fornøyd ved å fullføre prosjektet i tide og teamet vil slippe mye overtid. [1]

3.1 Metode for ønskede endringer

Ønskede endringer fra oppdragsgiver vil følge det løpet som er vist i *Figur 1 prosess for kravendringer*, nedenfor.



Figur 1 - Prosess for kravendringer

1. Oppdragsgiver sender en skriftlig forespørsel til prosjektleder på email om varsel for endringer/tillegg. Det skal inneholde hva endringen består av og årsak til endring.

2. Prosjektleder sammen med prosjektgruppa tar vurdering av endringens omfang, fremdriftskonsekvens og forslag til kravendring.

3. Fremlegge kravendringsforslag til bedriften:
   - Hvis endringen blir godkjent vil det bli laget en endringsordre.
   - Eventuelt videre diskusjon kan avtales mellom prosjektgruppa og arbeidsgiveren.
   - Hvis ikke, blir det sendt avslag på forespørselen med begrunnelse.

4. Dersom prosjektgruppa vil kreve fristforlengelse som følge av endringsordren, skal det varsles skriftlig til oppdragsgiveren.

4 Eventuelle svikt med prosjektet

Hvis det er noe svikt som påvirker at prosjektet ikke blir levert eller ferdig til tide, så skal dette varsles snarest til oppdragiveren. Prosjektgruppa skal angi sin vurdering av forbyggende og begrensende tiltak og fastsette en frist for korrigerende tiltak. [2]

5 Ønskede endringer fra prosjektgruppa

Endringer sendes som skriftlig forespørsel til oppdragiveren. Det burde også diskuteres muntlig.

6 Godkjenningsprosessen

Godkjenningsprosessen bør inneholde nødvendige og gunstige forespørsler som må tilrettelegges med passende planlegging, kostnadsjusteringer, samt som upassende forespørsler blir luket ut. [1]

7 Oversikt over endringssaker (Loggføring)

Prosjektlederen skal sørge for at det føres logg over alle endringsønsker. En god praksis er å registrere og spore alle endringsønsker i en endringslogg. Denne loggen vil bli en sentral kilde til informasjon om innsendte endringer og en viktig kommunikasjonsverktøy mellom prosjektets interessenter og gruppemedlemmene.

8 Arkivering

Endringshåndteringen skal dokumenteres ved at dokumentene produseres, ekspederes og journalføres fortløpende i arkivverktøyet.

9 Referanser

[1] Enterprise Systems Journal, 11.02.2016, https://esj.com/Articles/2010/10/05/Managing-Requirement-Changes.aspx?Page=2.

[2] Difi, 11.02.2016, http://www.prosjektveiviseren.no/h%C3%A5ndtere-endrings%C3%B8nsker-underveis.

[3] Undervisningsbygg, 11.01.2016 https://kgv.doffin.no/app/docmgmt/downloadPublicDocument.asp?DVID=452747&FMT=1&AT=15&ID=126082.

Høgskolen i Sørøst-Norge

KONGSBERG

DESIGNDOKUMENT

PROSJEKT	K-Spider		
OPDRAGSGIVER	Kongsberg Maritime AS		
UTFØRT VED	Høgskolen i Sørøst-Norge, avd. Kongsberg		
REVISJON	2.0		
MEDLEMMER	Aleksander Tokle Poverud, Masoud Shah Pasand, Tor Gunnar Finnerud, Hanna Kåsastul, Paul Knutson Sæther, Abdurahman Senkaya		
Dokumenthistorikk	**REVISJON**	**UTGITT**	**BESKRIVELSE**
	1.0	11.03.2016	Første utgave
	2.0	22.05.2016	Andre utgave

INNHOLDSFORTEGNELSE

1 DOKUMENTHISTORIE 5
2 INNLEDNING 5
3 UTSTYR 7
3.1 RCU 8
3.2 Mobil plattform 9
3.3 Bindeledd mellom RCU og Plattform. 9
3.4 Trådløse kommunikasjons muligheter 10
4 SYSTEMDESIGN 11
4.1 Utførelse og implementasjon 12
4.2 Modbus kommunikasjon 13
4.3 AIM2000 13
4.3.1 Operatørstasjonen (OS) 14
4.3.2 Prosess-stasjonen (PS) 14
4.3.3 Konfigurering i operatørstasjonen (OS) 15
4.3.4 Telegrammer 20
4.3.5 Simulering 25
4.5 Systemtopologi 27
..... 28
4.6 RCU til Raspberry design 29
5 Valg og utførelse av Mobil plattform 30
5.1 Konstruksjon av Mobilt Platform 31
5.1.2 Rekonstruksjon av hexapod med Atmega2560 34
5.2.2 Kommunikasjon mellom Atmega2560 og SSC-32U 39
5.2.3 Koble sammen Arduino og PS2 mottaker sammen 40
3.4 Komponenter 41
5.1.1 Kobling av servo til SCC32U 42
5.3 Servomotor 45
5.3.1 Vår bruk av Servo 46
5.3.2 Kalibrering av servomotor 47
5.3.3 Ekstern kontroller 47
5.3.4 Funksjoner til ekstern kontroller 48
5.4 Programmering av Hexapod 50

5.4.1 Fil oppsett....51
5.4.2 K-spider med Atmega2560:....52
5.4.3 Bruken av PS2 kontroller (PS2_controller.cpp):....54
5.4.4 Digitale hexapod funksjoner:....55
5.4.5 Analoge hexapod funksjoner:....57
5.5 Raspberry Pi 2....60
5.5.1 Kommandosettprotokoll....61
5.5.2 Kontrollkommandoer....65
6 REFERANSER....66

TABELLER

Tabell 1 - Dokumenthistorie.... 5
Tabell 2 - Utstyrsliste for systemet....11
Tabell 3 - Utstyrsliste mellom RCU og RPi....30
Tabell 2 - Arduino og SCC32U....39
Tabell 3 - PS2 mottaker....40
Tabell 4 - Komponenter....41
Tabell 5 - SCC32U VS1....44
Tabell 6 - SCC32 VS2....44

FIGURER

Figur 1 - Systemet som tilfredstiller kundens krav.... 6
Figur 2 - RCU510.... 8
Figur 3 - AIM200 Hardware component....14
Figur 4 - Bildeparametrer i AIM world space....15
Figur 5 - Typer av bilder i AIM....16
Figur 6 - Backup Control....16
Figur 7 - Modulbiblioteket....17
Figur 8 - Visning av kontrollmoduler i flowview....18
Figur 9 - HotSpot....19
Figur 10 - PopUp vinduet i flowview....20
Figur 11 - IO Manager....21
Figur 12 - Telegramblokk....22
Figur 13 - Parameterisering i telegramblokken....23
Figur 14 - Telegramsett....24
Figur 15 - Parameterisering i telegramsettet....25
Figur 16 - Simulering....26
Figur 17 – Systemtopolgi....27
Figur 18 - Enkel flow av kommunikasjonen mellom RCU og RPi....29

Figur 19 - Sensorer med leg.....31
Figur 20 - Konstruksjon av Hexpod-rammen.....32
Figur 21 - Ferdig konstruert Hexapod 33
Figur 22 - Ferdigstilt Hexapod 37
Figur 23 - Arduino mega og Scc32U 39
Figur 24 - PS2 mottaker.....41
Figur 25 - led og Buzz-er 42
Figur 26 - Servo pinne konfigurasjon 43
Figur 27 - Reguleringssløyfe for servomotor.....45
Figur 28 - PWM.....46
Figur 29 - Ekstern kontroller.....47
Figur 30 - Kontroll funksjoner.....48
Figur 31 - Hexapod funksjonstabell.....63

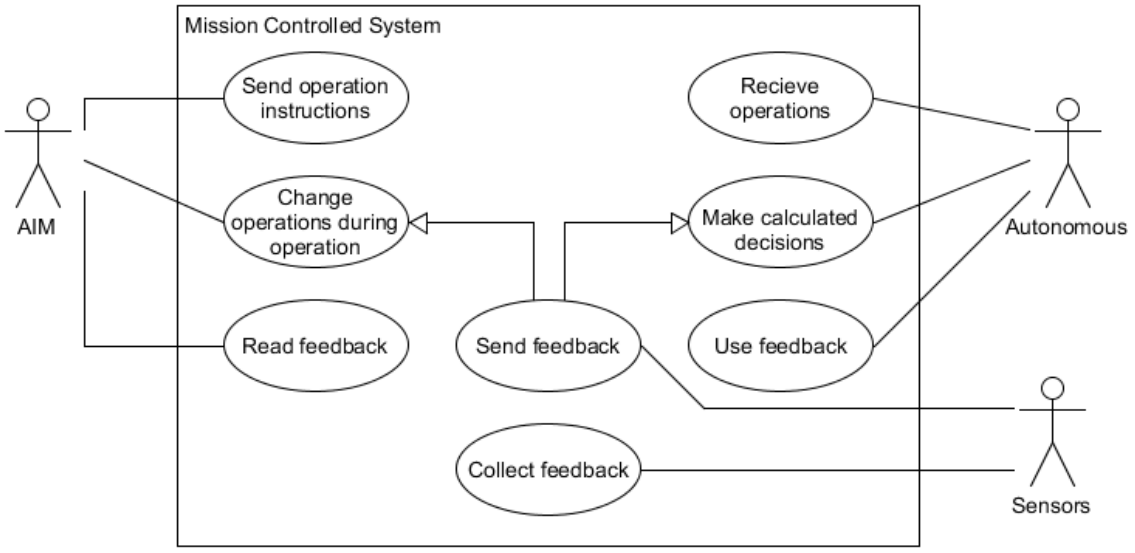
1 DOKUMENTHISTORIE

Tabell 1 - Dokumenthistorie

VERSJON NR:	DATO ENDRET	ENDRET AV	GODKJENT AV	BESKRIVELSE
0.1	25.02.2016	Tor Gunnar	Masoud	• Dokumentet ble opprettet
0.2	04.03.2016	Masoud	Aleks	• Oppdatert innholdet og lagt inn figurer.
0.3	08.03.2016	Aleks	Masoud	• Konstruksjon av robot, opplastning bilder, servoinformasjon og retting av div feil.
1.0	11.03.2016	Masoud	Aleks	• Første offisielle utgave
1.1	15.03.2016	Masoud	Hanna	• Endring av figurer og tekst i kapittel 4.3 (AIM2000)
1.2	18.05.2016	Hanna	Masoud	• Oppdatert innholdet i kapittel 4.3 (AIM2000)
1.3	20.05.2016	Aleksander	Masoud	• Endringer lagt til i kapittel 5 om fysisk prototyping
2.0	22.05.2016	Masoud	Aleksander	• Revidert oppsett • Dokumentet er godkjent og utgitt.

2 INNLEDNING

Dette dokumentet skal gi et innblikk i gruppen sin tankegang om valg av design og funksjonalitet i forhold til produktet som skal lages. Dokumentet tar for seg kravene for valg av nødvendig utstyr, oppbygging og utførelse, programmering og implementasjon. Det skal danne grunnlaget for videre arbeid med å få implementert ett system som tilfredsstiller oppdragsgiverens krav.



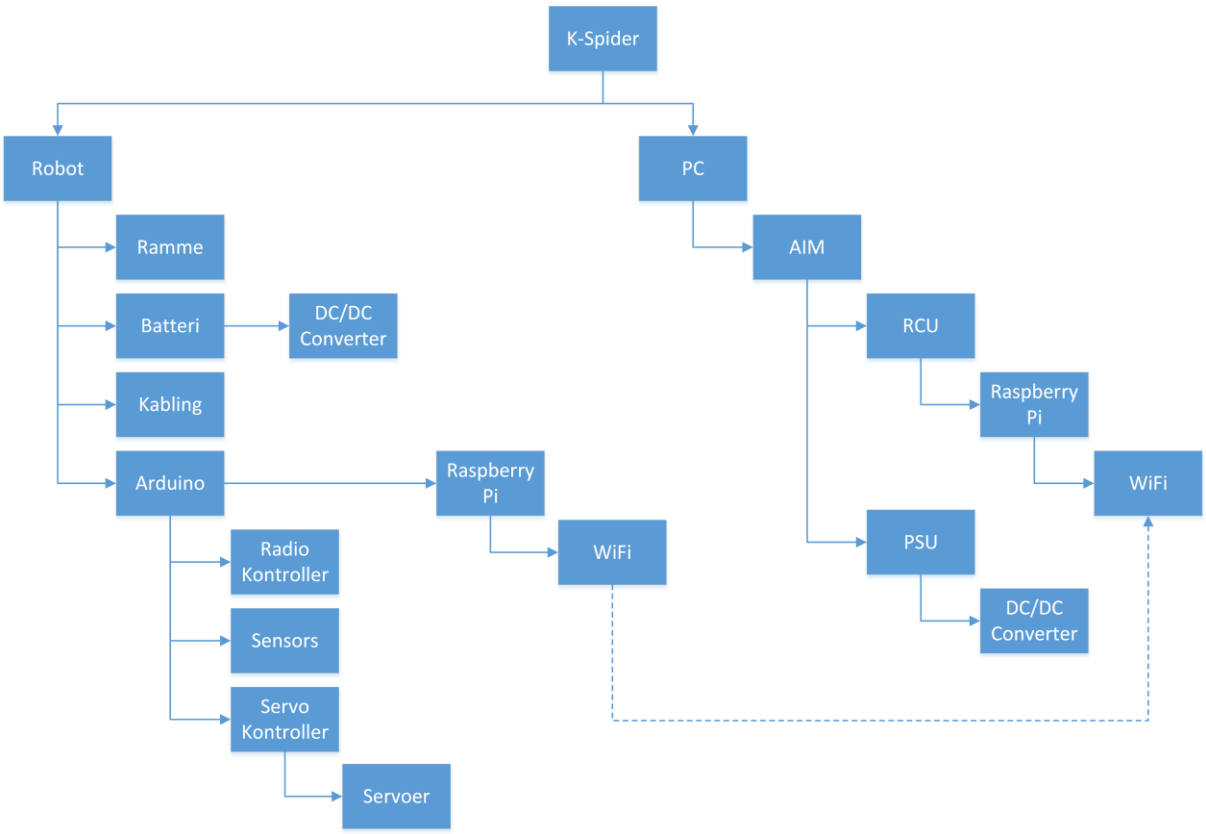
Figur 1 - Systemet som tilfredstiller kundens krav

3 UTSTYR

I dette prosjektet er det blitt utlevert en RCU kontrollenhet fra KM, sammen med en strømforsyning. Denne kontrolleren er essensiell for prosjektet, si den det var et av kravene fra KM at denne skulle brukes som kontrollerende organ for systemet.

Bortsett fra kontrolleren så har gruppen måtte ta noen valg angående de resterende delene:

- Mobil plattform som skal styres
- Bindeledd mellom RCU og plattform (enhet som skal tolke det kontrolleren sier og videreføre dette til plattformen).
- Eventuelle trådløse kommunikasjons løsninger
- Eventuelle sensorer og lignende.



3.1 RCU

Prosjektgruppa har fått utlevert en RCU510 av KM. Denne skal kobles med en tilhørende 24VDC spenningsforsyning. Denne spenningsforsyningen heter Quint power og er produsert av Phoenix Contact. Det er en 24VDC ± 1%, med utgangsstrøm på 10A. Kontrolleren krever 24 VDC ± 20% og en effekt på 20W. Dermed er det ikke noe problem ut ifra databladene å bruke denne forsyningen.

RCU er grunnsteinen i kommunikasjonen som skal oppnås med de andre delene vi skal se videre på. Kontrolleren har muligheter for kommunikasjon igjennom Ethernet, tradisjonell I/O[1] og profibus. Gruppa har valgt å ta i bruk en modbus kommunikasjon for prosjektet.

Figur 2 - RCU510

[1] Med tradisjonell I/O menes; Digitale signaler, samt 0-10V /4-20mA analoge signaler

3.2 Mobil plattform

Et krav fra oppdragsgiver var å velge ut en mobil plattform som skulle kontrolleres. Siden kommunikasjonen vi skulle produsere skulle være i stor grad uavhengig av plattformen, det vil si at det skal være minst mulig rekonfigurasjon nødvendig for å bruke en annen plattform ved en annen anledning. Dette gjorde at valget av plattform ble betraktelig enklere, så hovedfokuset rundt dette valget ble fokusert rundt pris, laste evne (når eller hvis det skulle bli aktuelt å implementere dette), showfaktor (hvis det skulle brukes til senere presentasjoner) og kanskje mest viktig, ekstern tilgang på tidligere erfaringer og support når det gjaldt denne plattformen.

3.3 Bindeledd mellom RCU og Plattform.

Vi trengte en måte å tolke informasjonen som RCU ga ut og gjøre dette om til noe plattformen kunne forstå, siden denne er basert på Arduino teknologi. Og eventuell mulighet for å gjøre RCU informasjonen om til et trådløst signal, da RCU ikke har noe direkte mulighet for å sende trådløst. Med dette i bakhodet under jakten på mulige løsninger, så falt vi på Raspberry pi(RP). Dette valget ble tatt på bakgrunn av RPi sin all bruks evne, der den kan jobbe med både Python og C-programmeringsspråk.

Siden det essensielt er en «liten» pc med eget Operativ system, prosessor, minne og god lagrings kapasitet, så er den en kraftig komponent for en lang rekke ulike kommunikasjons formål. Den er allerede blitt brukt som kontroll enhet i flere forskjellige andre industrier. Den har kommunikasjons muligheter igjennom USB, ethernet, samt seriell tilkobling, noe som var en attraktiv kvalitet vi lette etter. Men kanskje det sterkeste argumentet er at den er blitt koblet til Arduino teknologier utallige ganger før, slik at å finne informasjon ikke burde være så vanskelig. Både for modbus kommunikasjon og eventuelle trådløs overføringer senere i utviklingen.

3.4 Trådløse kommunikasjons muligheter

En trådløs kommunikasjons løsning er ikke akkurat et A krav, men det er noe gruppa har veldig lyst til å implementere og her er det igjen noen forskjellige muligheter å velge mellom.

Det er:

- IF, eller Infrarøde frekvenser
- WiFi
- RF, Radio frekvenser

Infrarød frekvens kommunikasjon er i og for seg en grei trådløs løsning i mange tilfeller, men den har en stor svakhet. Siden IF er en stråle basert overførings form, så krever den at en konstant «sikter» senderen på mottakeren, noe som innebærer at en må ha fri sikt til enhver tid. Denne svakheten var såpass stor at vi forkastet denne løsningen fort.

Wifi er en kommunikasjon form som også er kjent som **IEEE 802.11 standarden**. Den bruker en Radio frekvens signaler, men på en veldig spesifikk båndbredde. Wifi er en veldig utbredt kommunikasjon form når det kommer til å koble sammen data systemer, både i kommersielt bruk og på hobby basis. Det vil si at det er en lett tilgang på informasjon å finne om slike løsninger. Derfor valgte vi å benytte oss av denne løsningen.

Radio frekvens og wifi er i prinsippet akkurat det samme, hvor forskjellen er at RF er et mer generellt uttrykk. Hvor wifi opererer innen en spesifikk båndbredde, så er RF alle båndbreddene under et. Hvor bruksområdet i stor grad bestemmer hvilket båndbredde en vil bruke, da spesielt avstand kommunikasjonen skal foregå over er hoved faktoren og ta hensyn til. Dette er da også grunnen til at vi valgte å ikke benytte dette, siden innenfor rammene til oppgavene så var det ikke et behov for å kunne bruke systemet på lengre distanser. En gang lengre ute i visjonen av systemet som vil kanskje dette bli mer aktuelt.

4 SYSTEMDESIGN

Systemdesign skal vise hvordan gruppa har tenkt å gjennomføre løsningene våre og utstyret vi har tenkt å bruke for å oppnå dette.

Tabell 2 - Utstyrsliste for systemet

Område	Utstyrstype	Opprinnelse
Kontroller	RCU510	Kongsberg Maritime
Software	AIM2000	Kongsberg Maritime
	Python	
	C/C++	
	Arduino Software	
	SSC utility	
Strømforsyning	230VAC/ 24VDC	Kongsberg Maritime
Hardware	2xRaspberry Pi kort	Komplett.no
	Atmega2560	Egenbrakt komponent
	Servo-kontroller kort (SSC)	Robotshop.com
	HI-Tech Servo HS-645MG	Robotshop.com
	UBEC 5v/8A 6v/8A	Hobbyking.com
	Ultrasonic HC-SR04	Egenbrakt komponent
	Raspberry pi kamera v2	Qui-Huu Le-Viet
	PS2 controller&mottaker	Robotshop.com
	Lipo 11,1v & NI-MH batterier	Qui-Huu Le-Viet&Robotshop
Ramme/Leg	3DOF Aluminium leg og ramme	Robotshop

4.1 Utførelse og implementasjon

Planen er å ha en fysisk kommunikasjon mellom RCU kontrolleren og raspberry pi (RP) via kabel og modbus. Dette er fordi disse to komponentene uansett kommer til å holde seg samlet ved bruk, vi har også tenkt å finne en passende måte transporterer disse sammen på en enkel måte, ve d for eksempel en kasse eller en stor nok koffert.

Dette er den delen av prosjektet som er størst usikkerhet rundt. Det vil gå en god del mengde med timer her til å lære seg AIM, som er KM sin programvare og en kan dermed ikke tilegne seg informasjon fra andre kilder som internett. Det er blitt gitt grunnleggende opplæring i AIM fra KM, men utover dette så har de ikke mulighet til å gi annen hjelp enn via email eller over telefon. For å få RCU til å kunne styre systemet så er det essensielt at vi får til en god data overføring via modbus til RP, så dette kommer til å ta en god mengde tid.

Videre skal RP omgjøre signalene fra RCU og sende de videre til Arduino kortet på den mobile plattformen. RP som bruker Linux som Operativ system vil også ta en del tid, siden vi har ingen erfaring med Linux fra før så må vi sette oss inn i dette og finne ut hvordan vi bruker det. Når informasjonen er omgjort til noe Arduino kan jobbe med, så skal det overføres til plattformen. I første omgang kommer til å foregå ved en fysisk overgang, men senere så er ønsket at denne overføringen skal foregå ved trådløs kommunikasjon.

Når Arduino på plattformen har mottatt informasjonen, så skal den videre gi kommandoer til servo-kortet som styrer servoene hvis får plattformen til å bevege seg.

4.2 Modbus kommunikasjon

Vi har valgt å bruke modbus RTU-232 kommunikasjon, dette kommer av at:

- Siden det skal brukes digitale signaler, så passer dette. Informasjonen som sende relativt enkel.
- Det ønskes å tilbakemelding om utførte handlinger, dermed Rx og Tx serielle systemet i modbus fint.

RTU-232 er ikke spesielt godt skjermet for støy, men siden avstanden det er snakk er så liten, så vil ikke dette påvirke systemet i nevneverdig grad.

4.3 AIM2000

AIM2000 er et datamaskinbasert distribuert prosesstyrings- og overvåkingssystem levert av Kongsberg Maritime. Systemet brukes i et bredt spekter av programmer og typer fartøy for styring av:

- Strømstyringssystem
- Maskineri
- Propulsjon
- Ballast og cargo
- Utvidet alarmsystem

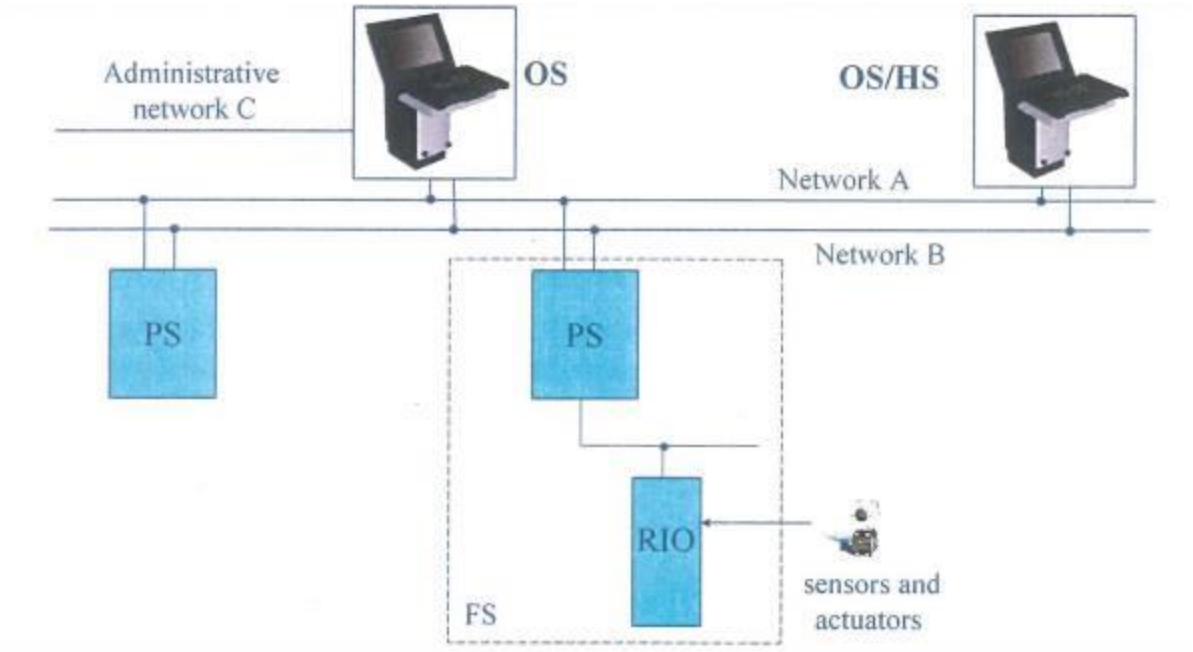
AIM200 systemet kan integreres med:

- Dynamisk posisjoneringssystemet
- Truster Styringssystemet
- Brann- og Gas systemet
- Nødavstengningssystemet

Hovedfunksjoner til systemet er å samle data, få nåverdi på skjermen, håndtere alarmer, manuell styring, automatisk kontroll, logging av data, rapportering, simulering og systemkonfigurasjon.

Systemet består av forskjellige typer stasjoner som kommuniserer med hverandre via et kommunikasjonsnettverk. Stasjonene som skal benyttes i denne oppgaven er:

- Operatørstasjonen (OS) – installert på en PC
- Prosesstasjonen (PS) – RCU510



Figur 3 - AIM200 Hardware component

For man kunne begynne å konfigurere inn kommandoene det vil være mulighet til å aktivere via RCU, som man lager i AIM.

- Må det kobles opp fysiske koblinger.
- Serielle portene settes opp mot hverandre og en informasjonsoverføring mellom portene må påvises.

4.3.1 Operatørstasjonen (OS)

Operatørstasjonen er et konfigurasjons- og operatørgrensesnitt som er installert på en PC. Operatørstasjonen (OS) er et utviklerens/operatørens vindu mot prosessen. Den brukes til overvåking, styring, konfigurering og simulering av prosessen. Den brukes også til nedlastning av program og konfigureringsdata til prosesstasjonen, til logging av data på masselager og til å utføre systemtesting.

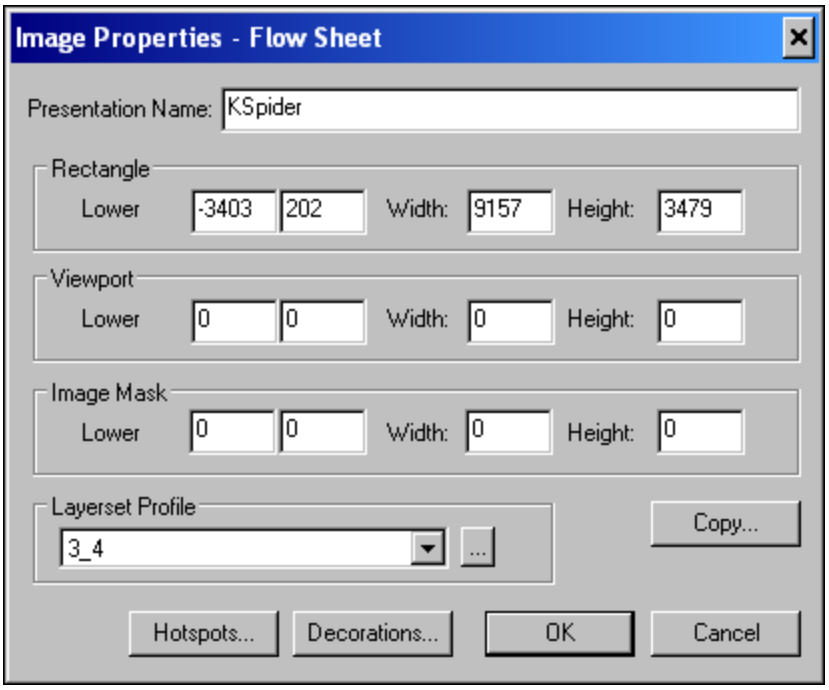
4.3.2 Prosess-stasjonen (PS)

Prosesstasjonen tar seg av det prosessnære arbeidet. Her avleses og settes inn- og utgangssignaler, og nødvendig signalbehandling og lagring utføres. Prosesstasjonen kommuniserer med andre stasjoner i systemet via kommunikasjonsnettverket. Prosesstasjonen består av datamaskinen RCU510.

4.3.3 Konfigurering i operatørstasjonen (OS)

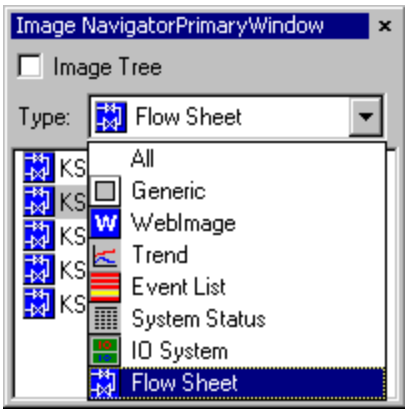
Konfigurering i OS beskriver hvordan tilstandene til systemet (nivåer, alarmer, logikk, settpunkter) skal presenteres til brukeren. Konfigurering i OS gjøres ved å koble sammen og parameterisere standardiserte funksjonsmoduler. I AIM2000 basis finnes alle de funksjonsmodulene gruppen måtte trenge for å kunne lage et PMS system.

Dette gjøres med såkalte "Images", eller bilder på norsk. Hvert bilde plasseres i "AIM world space". I det globale rommet befinner alle opprettede bilder seg. Hvert bilde "rammer" inn moduler, hotspots og decorations i det globale rommet.



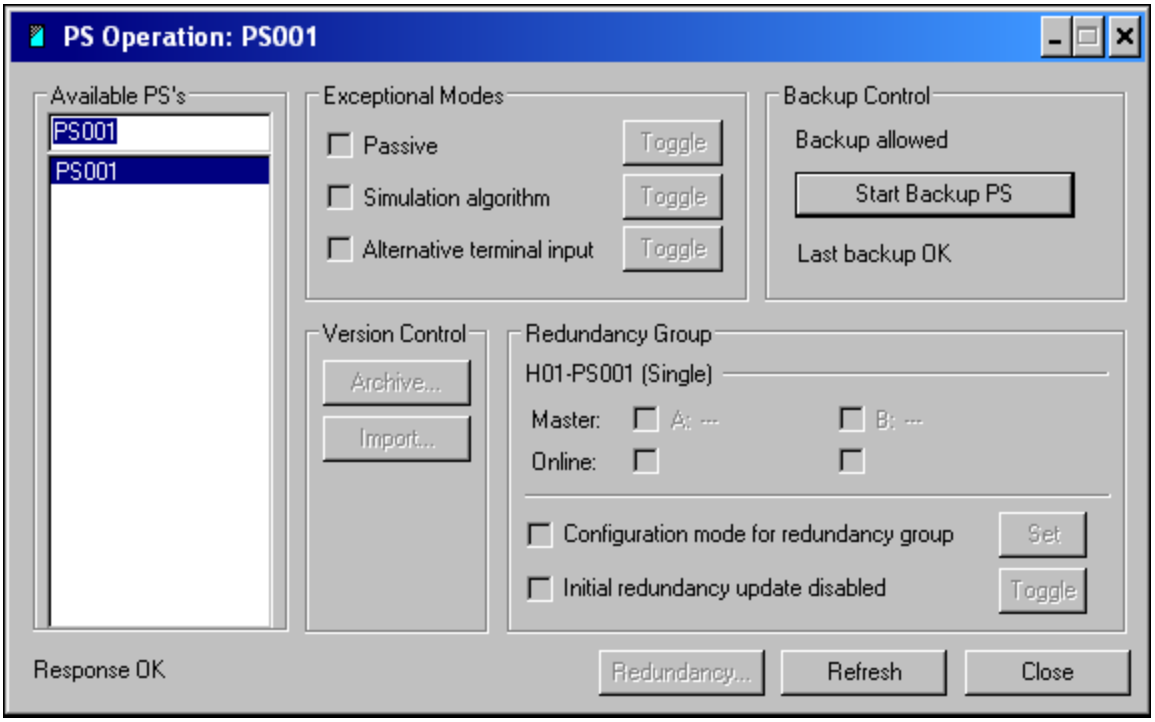
Figur 4 - Bildeparametrer i AIM world space

Det finnes flere forskjellige type bilder. De viktigste er så kalte Flow Sheet og Process images. Et Flow image er en grafisk representasjon av prosessen og signalene mellom funksjonsblokkene som beskriver den fysiske prosessen og kontroll av denne. Et Prosess bilde (Process image) er prinsipielt det samme som et Flytende bilde (Flow image), forskjellen er at et Prosess bilde har bedre og utvidede muligheter for grafisk presentasjon. Videre finnes det Trend Images for visning av tidsserier, WebImage for visning av websider og Event List image for visning av advarsler og alarmer. Det gjøres ved å velge Navigator på utstyrspanelet og velge den ønskelige typen av bildet.



Figur 5 - Typer av bilder i AIM

For å opprette et bilde må AIM-OS settes i OS configuration Mode. Dette gjøres ved å velge System -> OS configuration Mode. Alle endringene i bilder må lagres. Dette gjøres ved å velge File -> Save image. Oppsett av I/O moduler og koblinger mellom moduler lages i hardware, ikke på PC-en. Derfor er det veldig viktig å utføre Backup Control under konfigureringen i AIM. Dette gjøres ved å velge Operation->PS Operation, videre velges riktig Prosesstasjonen og velges Start Backup PS.



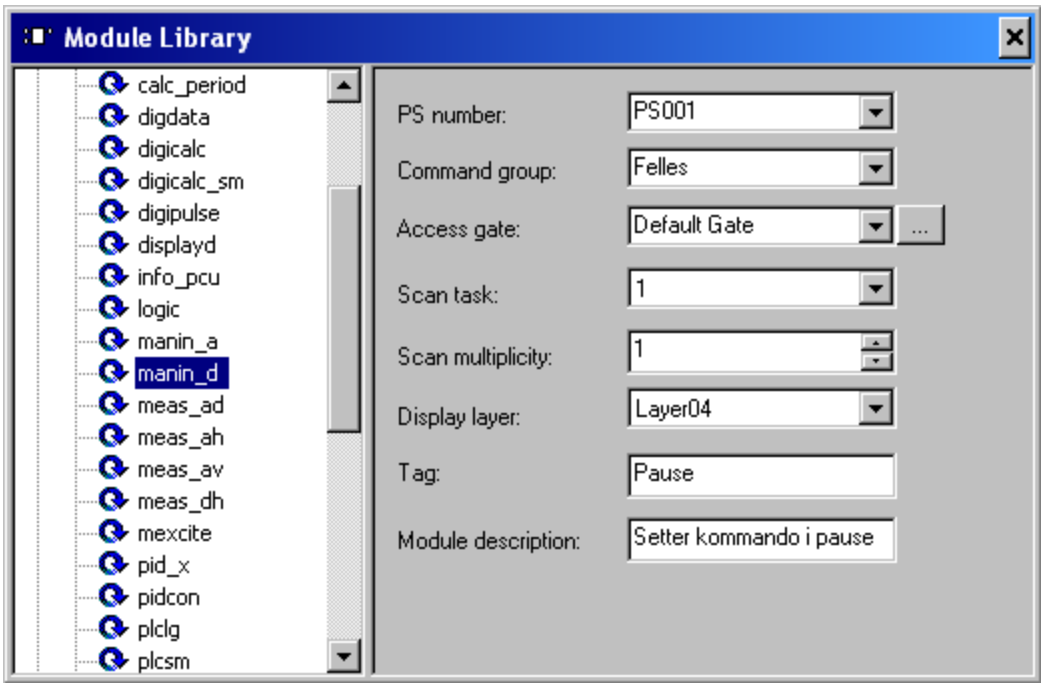
Figur 6 - Backup Control

Det første som må gjøres er å velge modulen fra modulbiblioteket, velge hvilken PS modulen skal ligge på (modulen skal ligge på samme PS som tilkoblede I/O). I denne oppgaven brukes kun én PS, nemlig PS001. I tillegg må hver modul ha et passende tag-navn som er unike. Dvs. programmet skal ikke kunne kjøres dersom to eller flere moduler har samme tag-navn.

I prosjektet skal brukes kun manin_a og manin_d moduler hvor en manin_a modul tilsvarer en bryter med analog utgang, mens en manin_d tilsvarer en bryter med digital utgang. Det skal sendes

kommandoer til roboten som et ferdig laget sett, hvor man kan enkelt bytte, slette, legge til kommandoer. De enkelte kommandoene skal opereres med manin_a moduler for å få de ønskelige verdiene for videre overføringen til sensor plattformen. Man kan velge hvilket sett (pakke med kommandoer) som skal sendes med å operere med manin_d moduler som har fått tag-navn Set 1, Set 2 og Set 3.

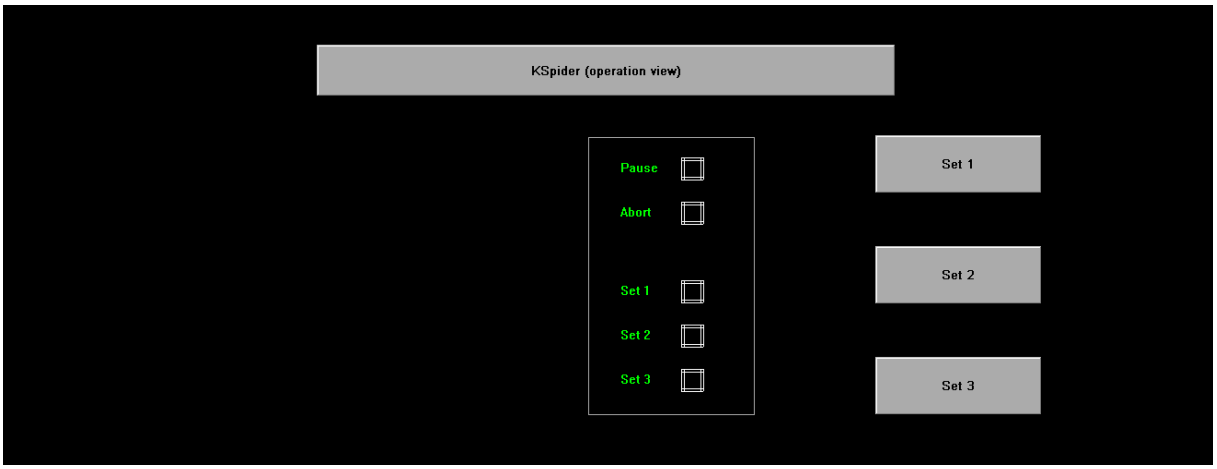
I tillegg skal det brukes to ekstra manin_d moduler for å sette settet i pause modus eller avbryte overføringen fullstendig.



Figur 7 - Modulbiblioteket

I dette eksemplet vises en manin_d modul med tag-navn Pause. Den modulen skal sette overføringen av kommandoer i pause når den skal være høy. Tilsvarende skal overføringen gå ut fra pause modusen når den skal være lav. Det hele bildet skal lages på lag04 (Layer04), derfor alle modulene må leges på lag04 ellers skal de være usynlige. Visnings området har blitt valgt 4_10, dvs. at alt som leges på lag mindre enn 4 eller større enn 10 skal være usynlig for operatøren. Derfor skal alle modulene lages på lag04, men tag-navnene er på lag11. Dette gjøres for å unngå overlapping av tag-navn og for å ha det ryddig og mer oversiktlig.

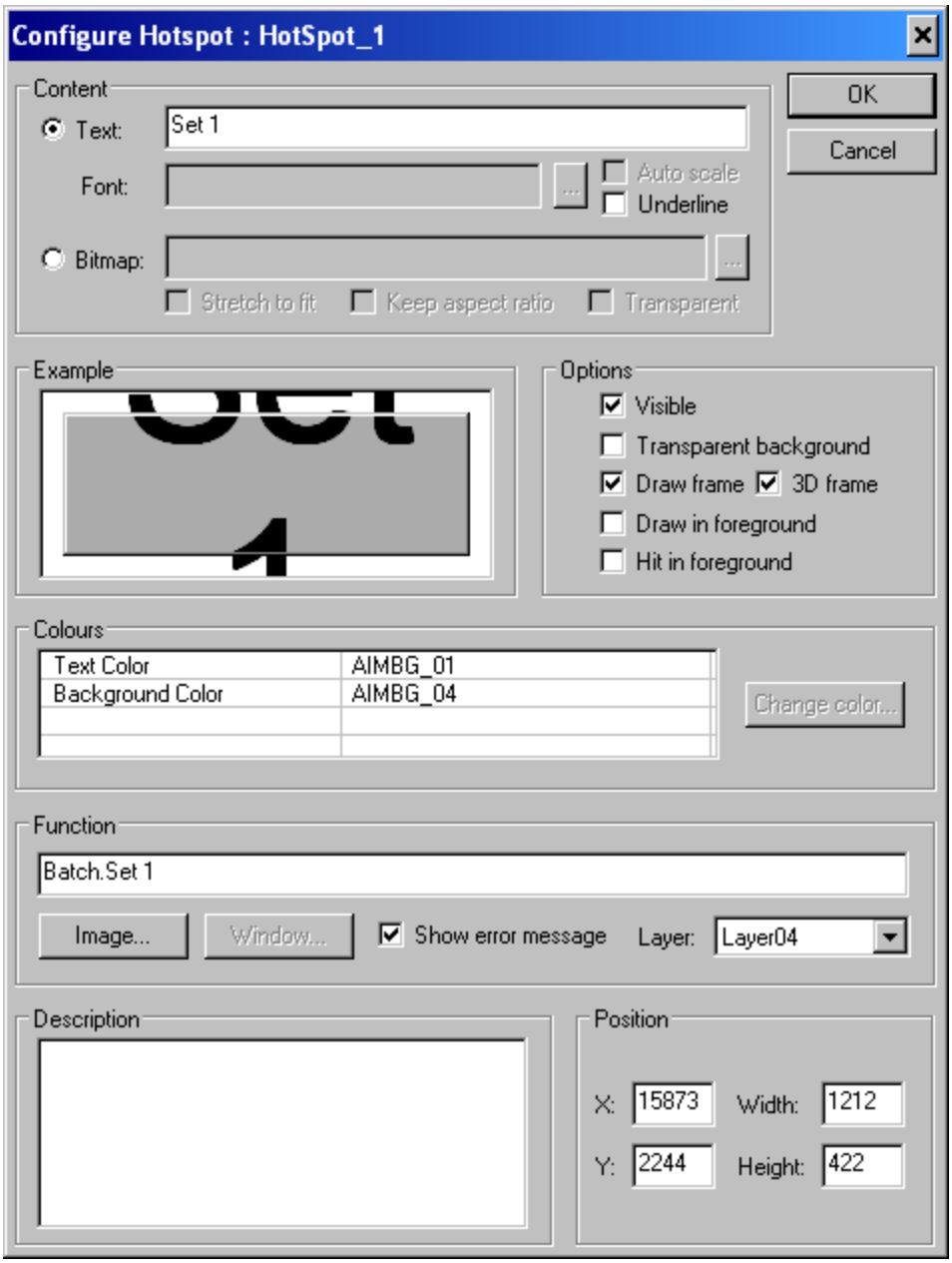
Etter å ha valgt modulen og gitt den et tagnavn settes den inn i flowview. Flowview er et visningslag i OS der alle kontrollmoduler ligger.



Figur 8 - Visning av kontrollmoduler i flowview

Etter modulen er blitt satt inn i flowview, må den kobles enten mot I/O, eller mot andre kontrollmodulene i systemet. Alle kontrollmodulene i prosjektet skal kobles mot I/O for å videre sende til Raspberry Pi via Modbus RS 232.

Hotspots er klikkbare områder (usynlig område over grafikk) som utfører funksjoner, typisk visning av et annet bilde. HotSpot-fuksjonen er blitt brukt i prosjektet for å operere med kommandopakker, dvs. som snarvei for å finne settet innholdet fra operatørens vinduet.



Figur 9 - HotSpot

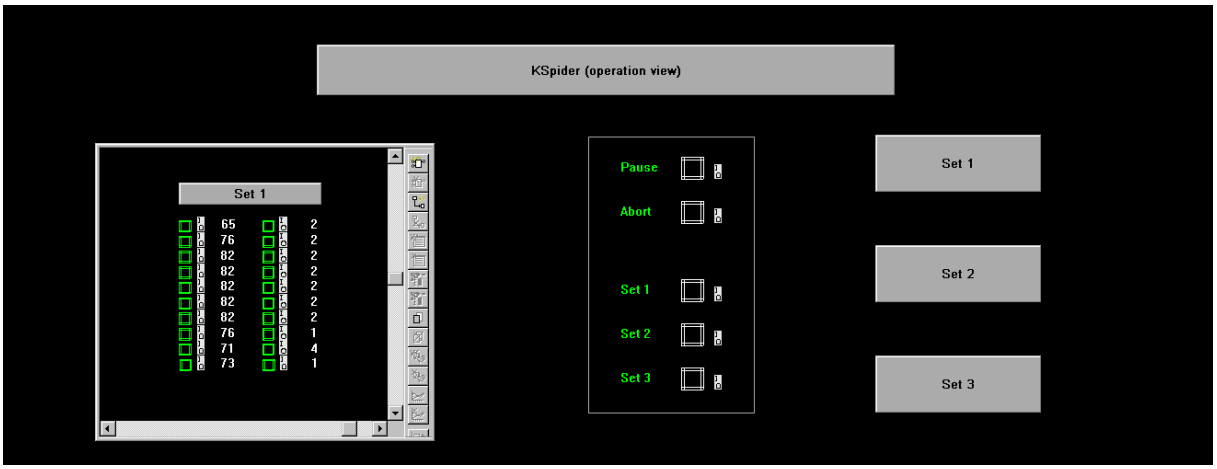
For at en HotSpot skal være en snarvei for et sett må det lages et PopUp vindu. Først må det defineres et bilde med de ønskelige funksjonene som skal tas opp etter behov. Og så lages et PopUp vindu. Dette gjøres ved å sette programmet i OS Configuration Mode og via OS Configuration velge Functions. Videre velges Batch som type og lages en batch funksjon med et passende tag-navn. I batch funksjonen legges to følgende strenger:

PopupManager.ShowPopup.<=CurrentImageWindowName>.ImageWindowPopup.KSpider_Set1. -Left=100 -Top=150 -Width=370 -Height=325 -NoNameWindow –TitleBar

ImageManager.ShowImage.<CurrentImageWindowName>.FlowImage.KSpider_Set1

De parameterne i PopUp manager beskriver hvordan PopUp vinduet skal vises i Flowview.

Figur 3 viser PopUp vinduet for Set 1 når man kliker på HotSpot med tag-navn Set 1.



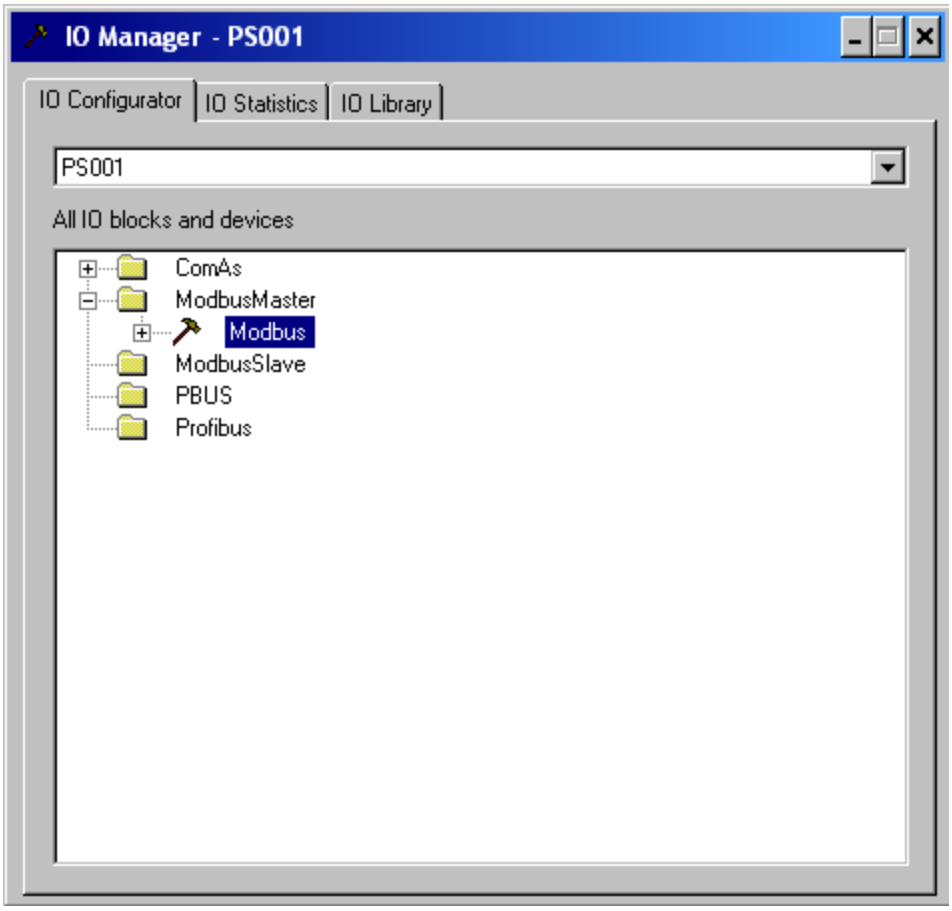
Figur 10 - PopUp vinduet i flowview

I tillegg ut fra figur 9 har det blitt laget to så kalte tomme HotSpot, den ene inneholder overskrift på visningsområdet, mens den andre omrammer de funksjonelle bryterne.

Decorations er HotSpot med grafikk som kan utføre funksjoner. Typiske dekorasjoner er ActiveX objekter, command buttons, andre flow og process images. Dekorasjoner har ikke blitt brukt i dette prosjektet.

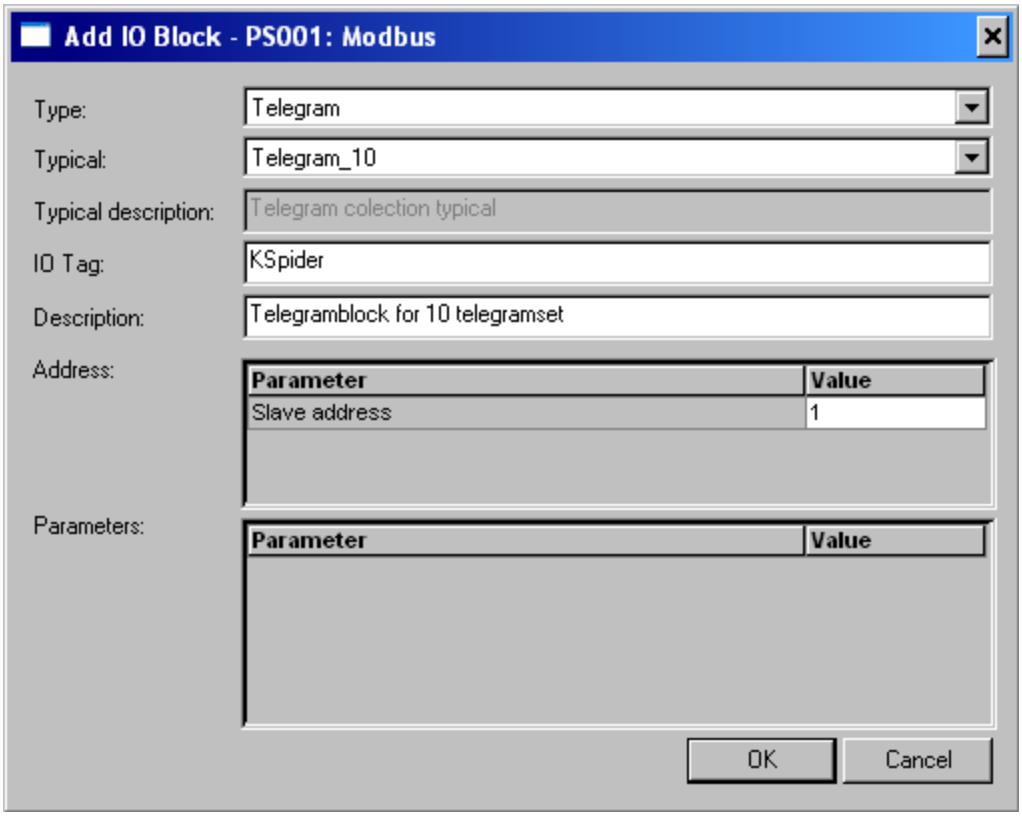
4.3.4 Telegrammer

Alle data fra AIM skal sendes til Raspberry Pi via Modbus RTU RS 232 som så kalte telegrammer. Dette gjøres ved å koble modulutgangene mot I/O i telegramblokken. I utstyrspanelet velges Tools -> IO Manager.. , vedere velges riktig PS i IO Cofiguration, nemlig PS001. Siden i denne konfigurasjonen Modbus RT 232 skal være en Master og Raspberry Pi er en Slave, velges ModbbusMaster -> Modbus.



Figur 11 - IO Manager

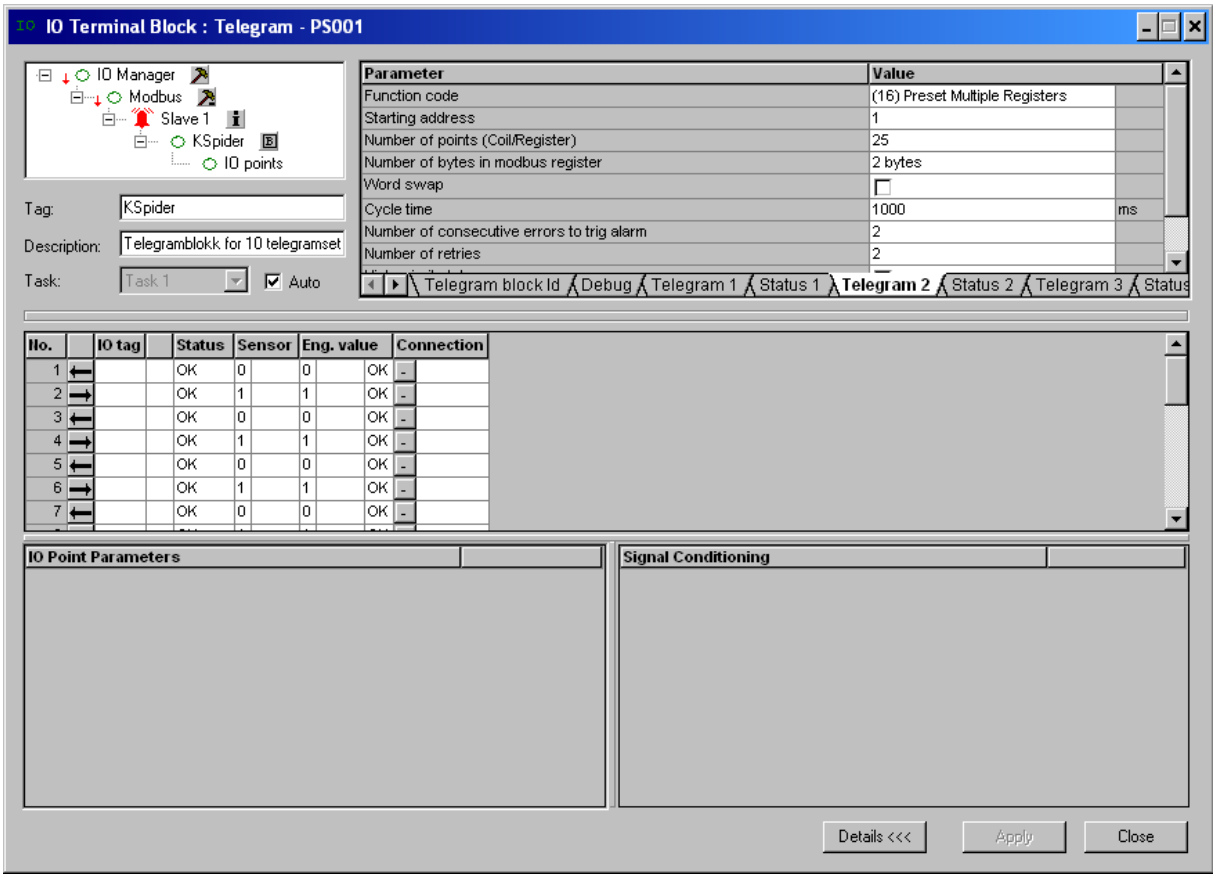
Videre fra Modbus velges Add IO Block, og leges til et telegramblokk med mulighet for 10 telegramsett som er den største telegramblokken.



Figur 12 - Telegramblokk

Etter det er blitt laget en telegramblokk må man definere og parameterisere telegramsett. Et telegramsett tilsvarer en Modbus funksjon, derfor skal data fra AIM sendes fra forskjellige telegramsett.

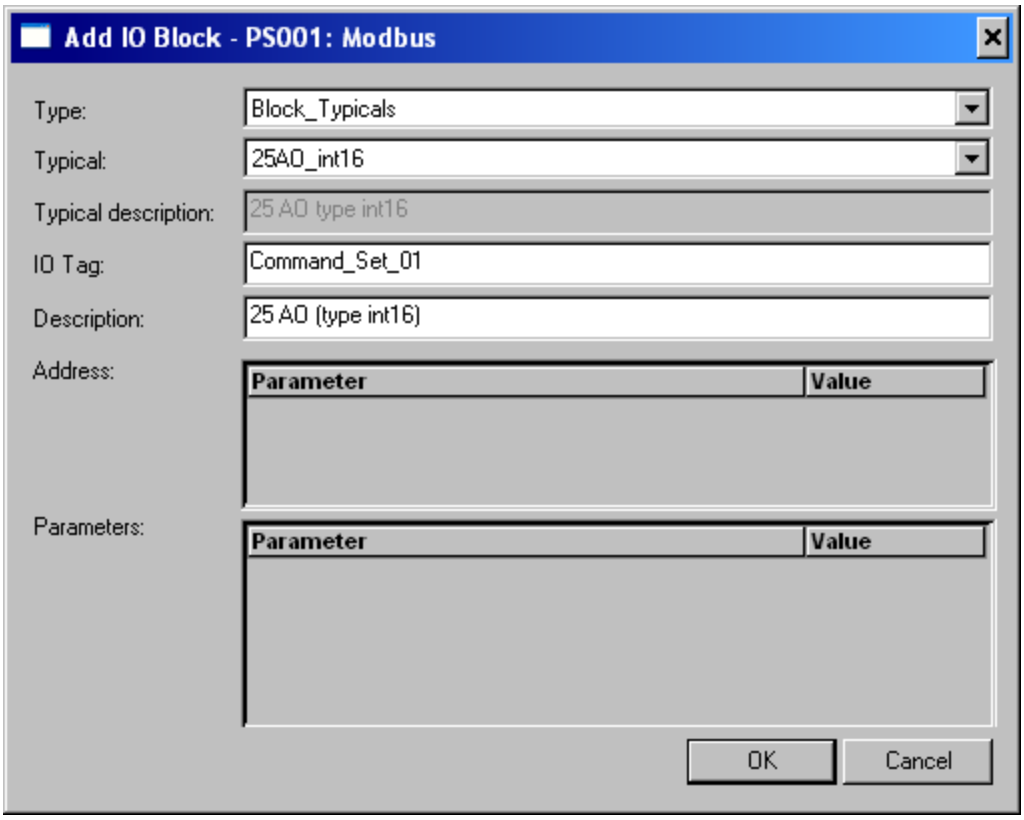
Fra telegramblokken definerer man hvilken Modbus funksjon skal brukes for hvert involvert telegramsett, start adressen, totalt antall enkle telegrammer i settet, antall bytes i Modbus registeret, syklustid, antall feil for å utløse alarm. I tillegg kan man gi et telegramsett høyt prioritet, debug.



Figur 13 - Parameterisering i telegramblokken

I dette eksempelet vises telegramsett 2 som skal sende telegrammer til Modbusen med Modbus funksjonskode (16) som skal kun skrive multiple registre. De multiple registrene trenges for å sende analoge data, men de digitale data skal sendes med Modbus funksjonskode (15) som skal skrive multiple coils.

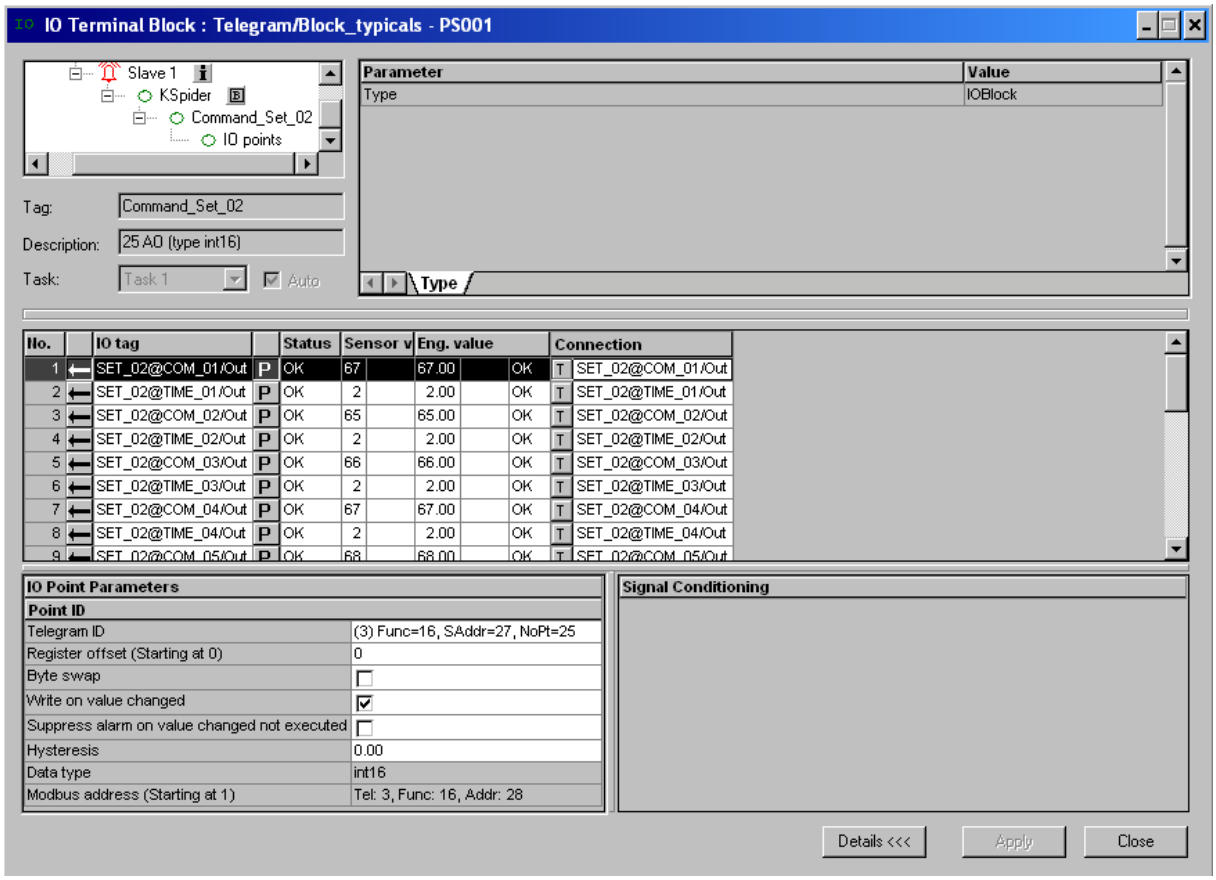
Syklustiden er på 1000 ms, dvs. dataene skal sendes hvert sekund. For å blokkere overføringen på en enkel måte kan man sette syklustid til 0. Antall punkteter (coils/register) er 25, dvs. dette telegramsettet kan sende 25 forskjellige data samtidig. Dette totale antallet tas ut fra hvert telegramsett som er visst på figur 10. I dette eksempelet er det blitt valgt et telegramsett med 25AO_int16 , dvs. 25 analoge utganger som skal sendes ut i int16 format (positive tall max 16^2=256).



Figur 14 - Telegramsett

Når man velger størrelse på et telegramsett lønner det seg å ta det settet med større totalt antall telegrammer enn det trengs i tilfelle man trenger å legge noe ekstra funksjoner til og å ha mulighet til å utvide systemet uten å innføre store forandringer i konseptet.

Etter det er blitt laget et telegramsett må man definere (koble modulutgangene) selve telegrammene.

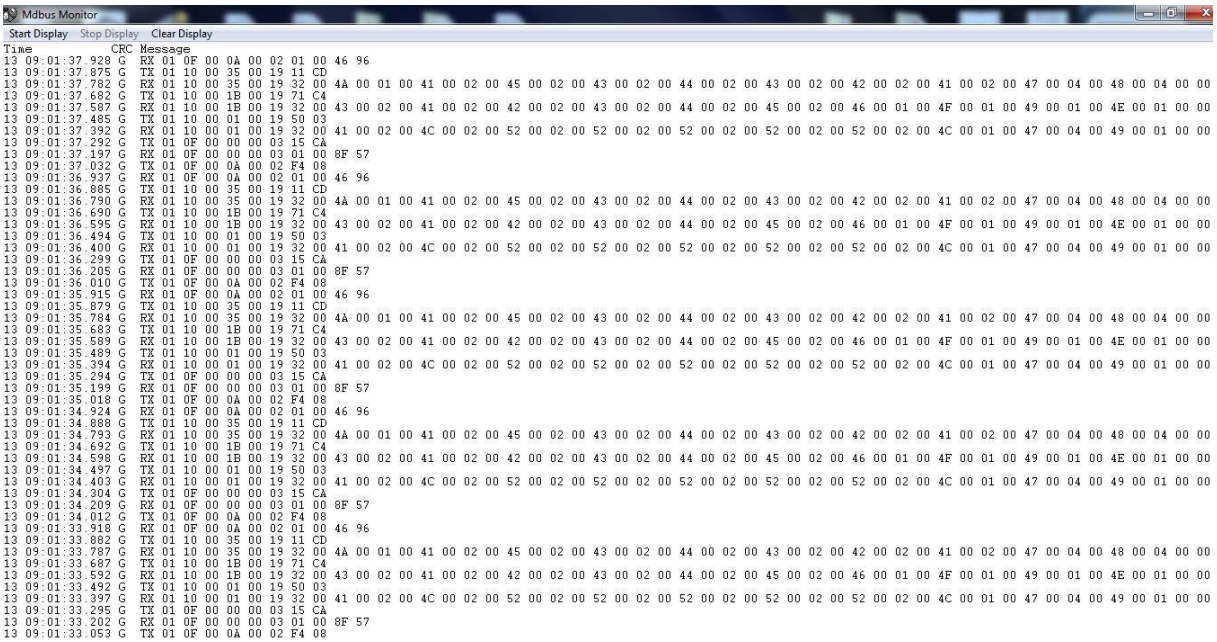


Figur 15 - Parameterisering i telegramsettet

Dette eksempelet viser alle utgangene til manin_a modulene som representerer Set2. Det er selve funksjonene til Hexapoden med tidslengde. I tillegg ser man at dette er telegram 3 med Modbus funksjonskode 16 (skal skrive multiple registre) med adressen 28. Adressen 28 viser på hvilken plass verdien på 67 skal vises ved simuleringen.

4.3.5 Simulering

For å teste om at det oppnås kommunikasjon og at alle data sendes med korrekte verdiene har det blitt utført simulering ved hjelp av en pc med installerte slave konfigurasjon på. Pc-en er koblet mot utgangen av Modbusen som overfører desimale verdiene som hex.

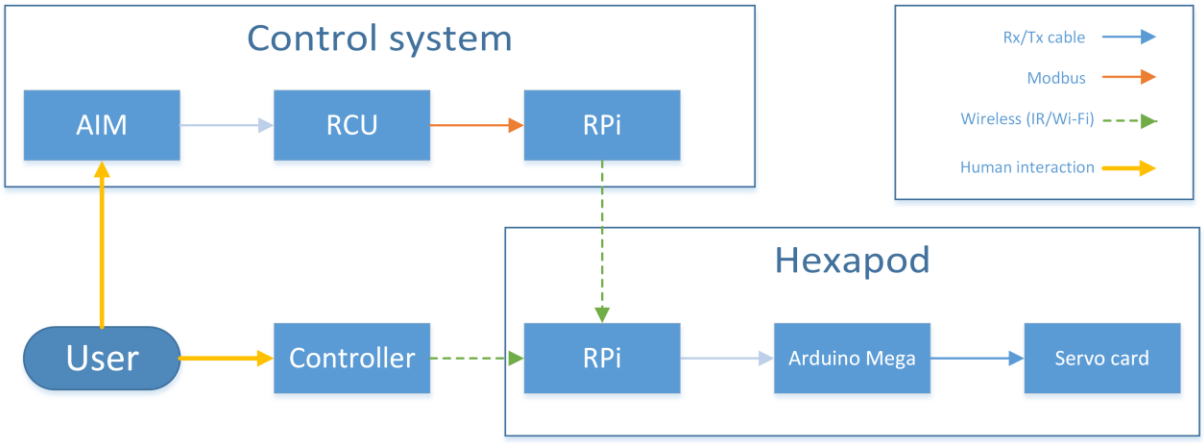


Figur 16 - Simulering

Figuren 13 viser overføring av data fra Set 1, Set 2 og Set 3 fordi alle de modulene er på, dvs. alle gir høye signaler, mens alle manin_d modulene er av, i så fall gir lave signaler.

4.5 Systemtopologi

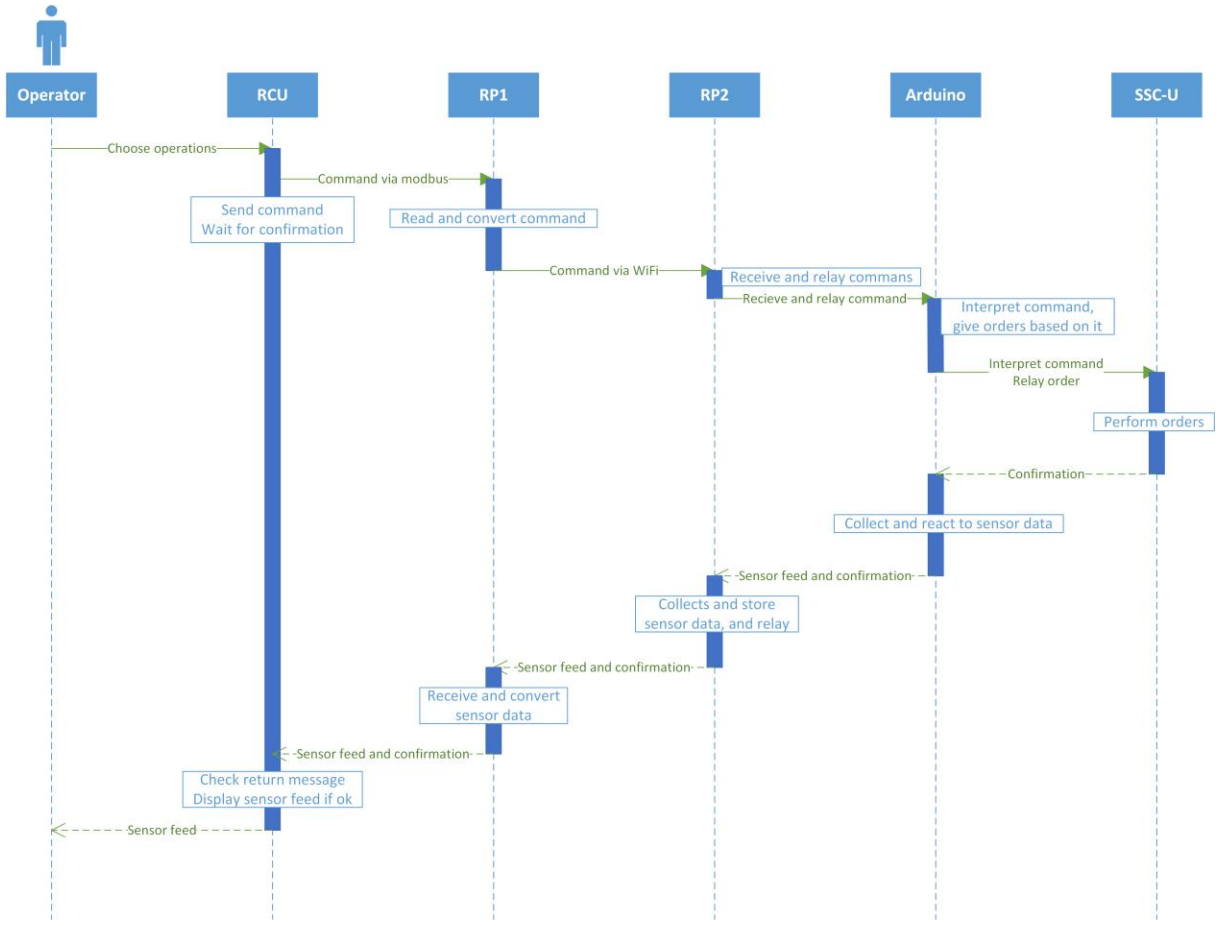
Kapittel 4.1 i dette dokumentet har tatt for seg hvordan oppbyggelsen og utførelsen av systemet er tenkt. Figur 2 presenterer en systemtopologi som enkelt viser hvordan kommunikasjon og oppbygging mellom de forskjellige komponentene er satt opp.



Figur 17 – Systemtopolgi

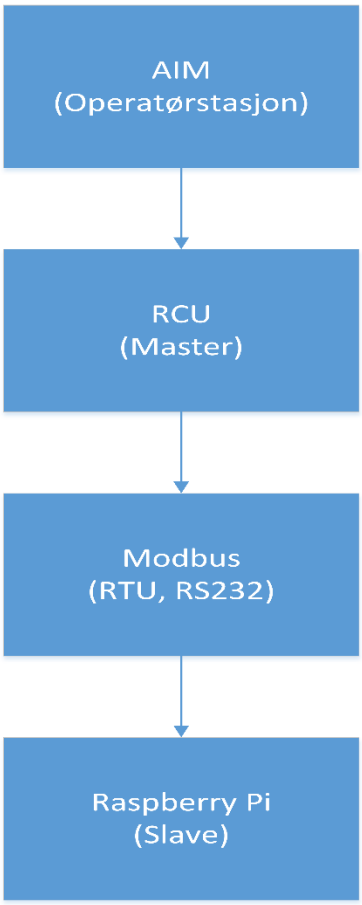
RCU 510 er som tidligere nevnt den sentrale komponenten i systemtopologien. Denne kontrollerer alle signaler, og kommuniserer med modbus ut til Hexapoden. Oppdragsgiveren har gitt uttrykk for at det burde gå greit å sette opp modbus-protokollene, men at det vil kunne dukke opp uforutsette hendelser som kan forsinke arbeidet.

Sekvensdiagram for dataflyt innad i systemet. Når systemet fungerer som det skal er dette hvordan signalene vil gå innad i systemet. Dette er spesielt viktig ettersom systemet har mange ledd som kommuniserer med hverandre, det er flere signaltyper der signalet kan stoppe eller ikke utføre de funksjonene som er tiltenkt.



4.6 RCU til Raspberry design

For å få en kommunikasjon mellom RCU og den mekaniske plattformen vår (hexapod), enten det blir gjennom kabel eller trådløst, så trenger vi en mellomkobling for å takle informasjonen som skal gå imellom disse enhetene. Denne mellomkoblingen har vi i utgangspunktet valgt til å være en Raspberry Pi. Figur 1 viser et overordnet flowchart av informasjonsflyten mellom AIM/RCU og Raspberry Pi via modbus.



Figur 18 - Enkel flow av kommunikasjonen mellom RCU og RPi

Tabell 3 - Utstyrsliste mellom RCU og RPi

RCU	Kontrolleren vi har fra Kongsberg Maritime
Raspbarry pi	Mini pc, med eget operativ system
Strømforsyning	24V 10A DC forsyning
Ethernet kabel	Standard Ethernetkabel
Com-port kabel	Seriell kabel med comport hode
Com/USB adapter	Adapter fra Com til USB
USB kabel	Standard USB kabel

5 Valg og utførelse av Mobil plattform

Sammen med prosjekt gruppa skulle vi bestemme oss for hvilken type Mobil Plattform som egnet seg for systemet vårt, som tidligere nevnt skulle det vare minst mulig rekonfigurasjon. Før vi valgte mobil plattformen var det viktig å se på hva kundens ønsker var. Kravspesifikasjonen som vi hadde jobbet sammen med oppdragsgiver var avgjørende for hvilken mobil plattform som skulle brukes i vårt oppdrag. En av de kravene fra oppdragsgiver var KR01(Rammekrav 01), der det står at, mobile plattformen skal operere i vanskelig terreng. Dermed ble alt som går på hjul var ute lukket for prosjektet vårt, ett annet mindre viktig rammekrav(KR05) sier at roboten skal utføre avanserte bevegelser. Etter mye søking på nettet har vi kommet frem til at en hexapod robot kan både operere i vanskelige terreng og at den kan utføre avanserte bevegelser, det viktigste av alt var at informasjonen om roboten var tilgjengelig. Sporing tilbake til kravspesifikasjon var avgjørende for tilfredsstille kundens ønsker, og dette har påvirket hele prosessen når det gjaldt valg av mobilt plattform.

I bestilling prosessen var det avgjørende å spore tilbake til risikoanalysen vi hadde jobbet med, de viktigste faktorene var om komponentene var defekte eller om vi brente komponenter under testing. Disse faktorene avgjorde hvor vi bestilte komponentene til prosjektet fra. Etter vurdering av risikoen av slike hendelser, var vi veldig nøye med hvor vi bestille delene fra. Prosjektgruppen undersøkte mulige løsninger og med tips fra ekstern veileder, har vi bestemt oss for å bruke en amerikansk nettside som heter robotshop.com, som hadde ekspress fraktløsninger og support.

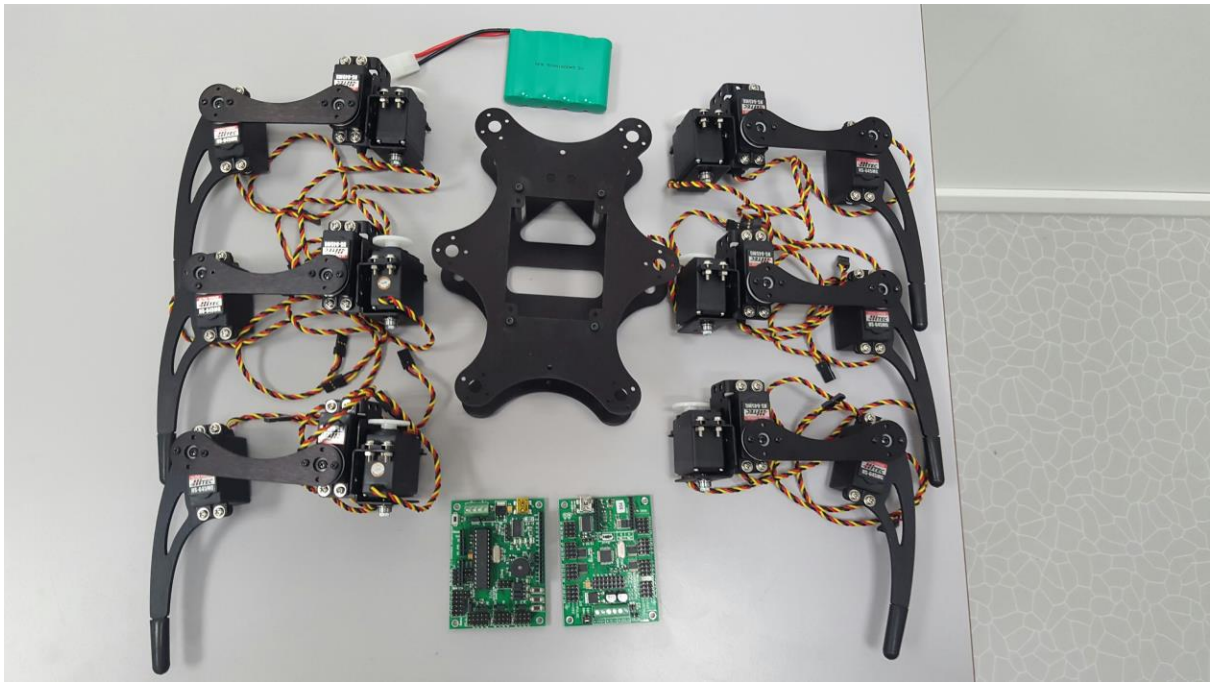
5.1 Konstruksjon av Mobilt Platform

Fra Robotshop.com bestilte prosjektgruppa ramme, batteri, mikrokontroller, servokontroller og servoer. Vi har begrunnet de mest kritiske valgene våre i teknologidokumentet.

Rammen som vi bestilte er en Hexapot 3DOF (3-degrees of freedom). Dette betyr at den har 3servoer per bein, altså 3frihets grader for bevegelse. Hexapot betyr at den har 6 bein og består da av 18 servoer som skal kjøre i samspill. Dette gjør at denne roboten kan gå i de fleste retninger, og utføre mer avanserte bevegelser. Dette kan spores til bake til rammekrav KR05 som sier roboten skal kunne utføre avansert bevegelse.

Figur 19 - Sensorer med leg

Ved konstruksjonen av beina måtte vi ta hensyn til senterposisjonen til servoene i hvert enkelt ledd og hvilken side beinet skulle bli montert på. Hvis det forekom avvik fra senterposisjon under montering så ville det kommet fram under kalibrering, avvik fra utgangsposisjon vil derfor bli korrigert.



Figur 20 - Konstruksjon av Hexpod-rammen

Figur(19) Viser Rammen rett før montering av de 6 beinene, samt Atmega2560, SSC, batteri og diverse kommunikasjonskabler mellom servokontroller og mikrokontroller.

Første vi gjorde var å skru beinene til rammen, for så å feste SSC kortet til rammen, deretter koblet vi pinnene til servoene inn i SSC kortet. Når SSC kortet og servoene var sammenkoblet opprettet vi kommunikasjon mellom SSC og «servo sequencer utility». I servo sequencer utility satte vi servoene til utgangsposisjonen, 0grader, og kunne dermed se feil ved montering eller om det trengtes å settes nye offsets (dette kalles også kalibrering). For mer info om hvordan servo posisjon og PWM kontroll fungerer, se 5.3.1.

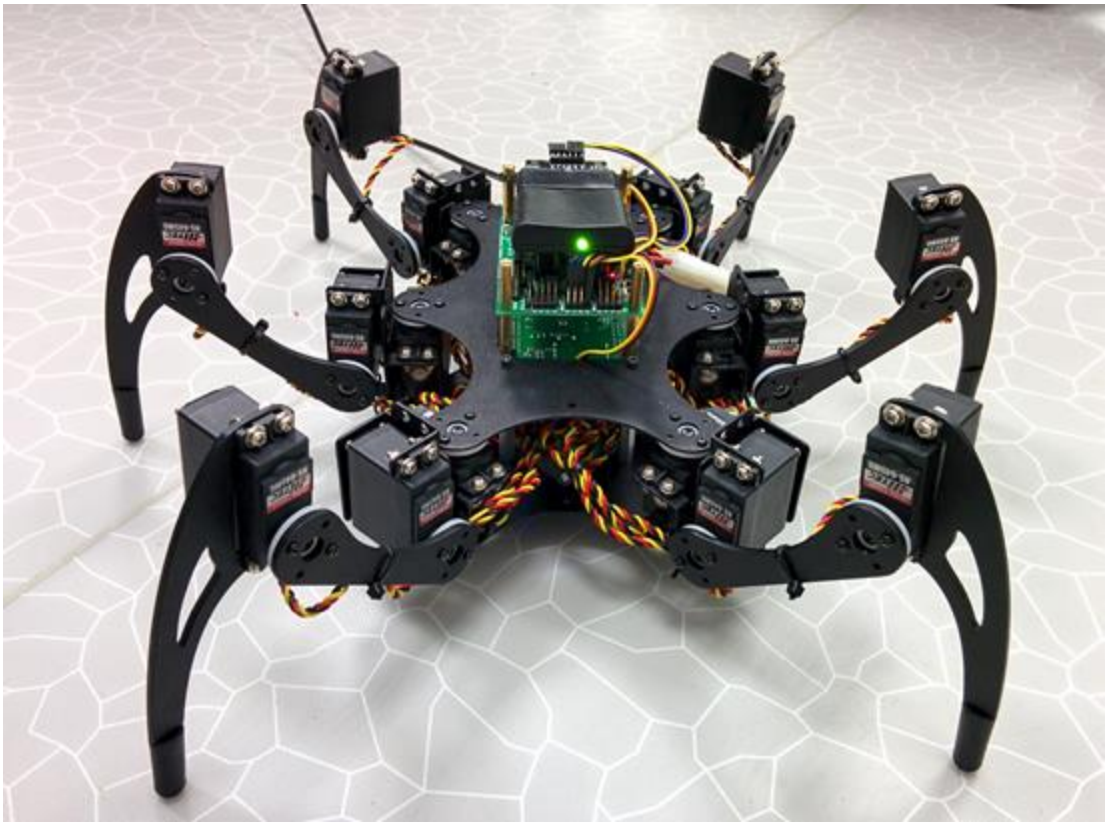
For å få kommunikasjon med SSC utility måtte vi sette kommunikasjonshastigheten til roboten og softwaren til 9600baud, på roboten gjør vi dette ved å holde baudknappen inne for å så repetitivt trykke til den lyser grønt. Roboten har forskjellige baud innstillinger «9600, 38400 og 115200»

De fleste utgangsposisjonene til servoene er -+5grader av den valgte 90grader posisjonen som vi har gitt servoene i Servo Sequencer Utility. Dette retter vi opp med å justere offsets til alle servoen e står i lik posisjon, denne posisjonsinformasjonen lagrer vi i SSC kortet. Dermed er roboten stabil og rett.

Deretter koblet vi på Atmega2560 og batteri, en switch til batteriet for å kunne kutte strømmen. Vi koblet strømmen til SSC og Atmega2560, og flyttet USB jumperen så Atmega2560 kunne få strøm fra batteri og ikke bare USB.

Når strømmen var i orden koblet vi på Tx/Rx kablene mellom SSC og Atmega2560 slik av vi fikk opprettet kommunikasjon mellom de to. Neste steg var å koble RF mottakeren på Atmega2560, det er denne mottakeren vi bruker til å få kommunikasjon med PS2 kontrolleren slik at roboten skal kunne kontrolleres.

Når alt det fysiske var i orden samlet vi bibloteker for arduino koden, endret koden til å spesifisere våre valgte pinner på SSC, og pinner på Atmega2560 brukt av RF mottakeren. Når dette var gjort kunne vi laste opp koden til arduino plattformen og koble den fra dataen. Vi satte roboten til å kommunisere på 38400baud, slik kontrolleren bruker, og dermed fikk vi kontakt med PS2 kontrolleren og vi kunne bevege hexapoden.

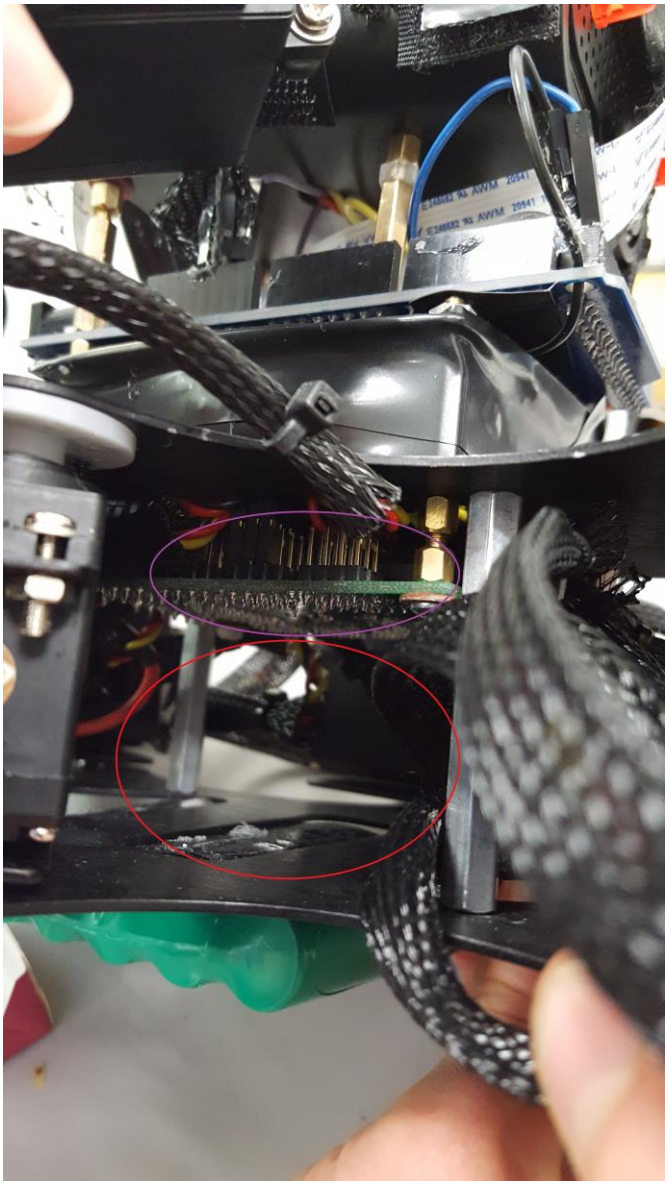


Figur 21 - Ferdig konstruert Hexapod

5.1.2 Rekonstruksjon av hexapod med Atmega2560

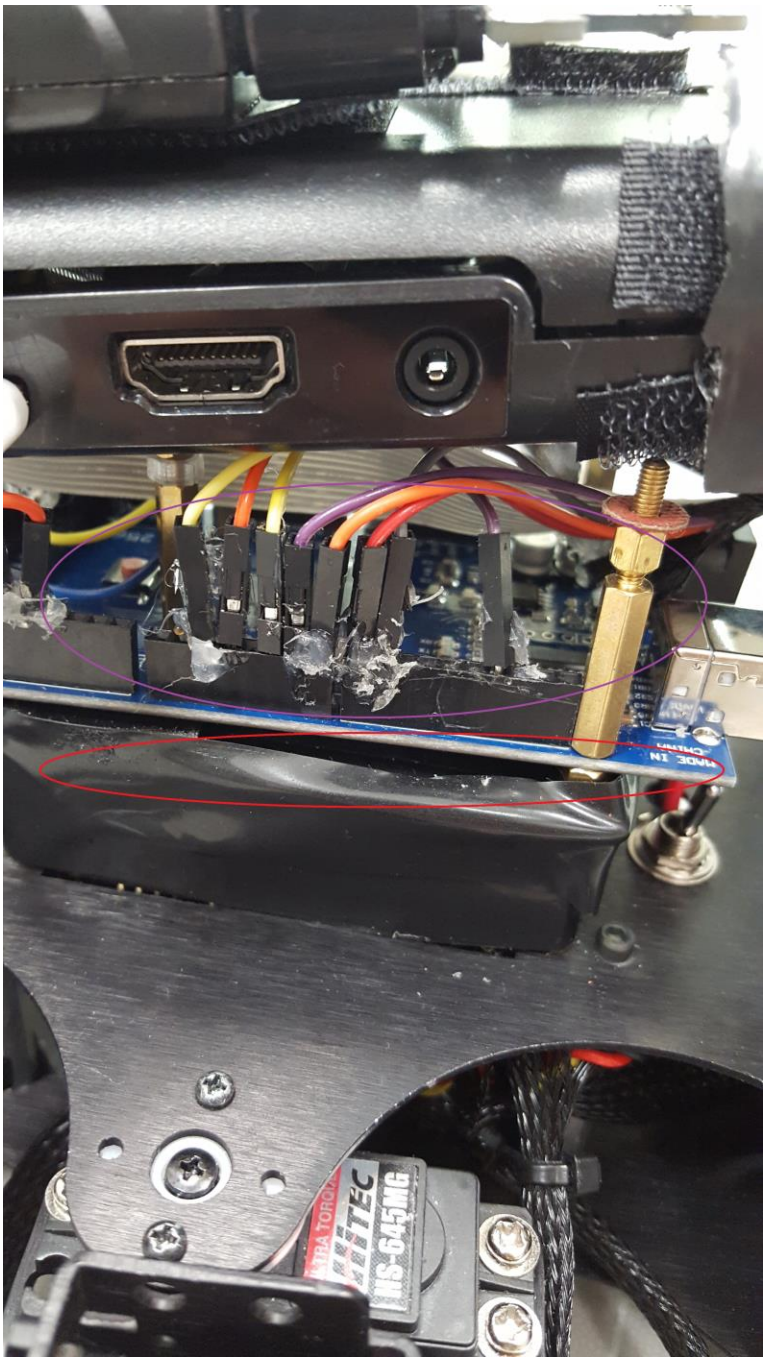
Vi har ombygd prototype plattformen vår slik at vi har fått mye kompakt plattform. Før vi koblet om var det ett høyt tårn. Endringer:

1) Vi har løftet SSC kortet opp, lilla sirkelen viser et montert rett under topp platen.
2) Den røde sirkelen viser all plassen vi har fått ved å flytte kortet opp, lipo batteriene skal plasseres her sammen med strømkilden til RPi
3) Man kan se de gamle grønne NI-HM batteriene montert under med borrelås.



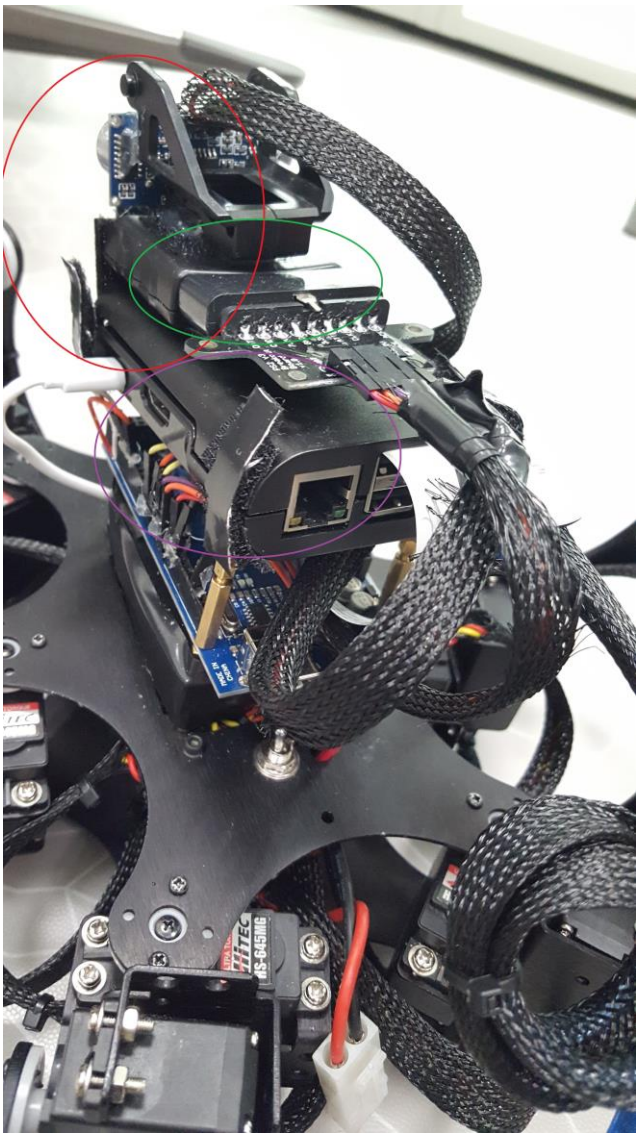
Andre endringer gjør det lett å komme til koblinger for arduino, SSC koblingene er tapet igjen da man ikke kommer til å tukle med dette med mindre hexapoden skal demonteres for godt.

1) Den røde sirkelen går rundt en stålplate vi har montert over SSC, denne låser kablene på plass og gir oss ett mye mer ryddig utseende enn tidligere. Serielle kablene Tx/Rx er dratt gjennom ett borehull i platen.
2) Lilla sirkelen er rundt de pinnene vi har byttet over til Megaen. Det er fortsatt høyt nok til å koble standard arduino pins. Dette er fordi det skal være lett å koble til nye sensorer og komponenter uten å måtte skru ting løs. Vi har unngått å lodde på grunn av tanke på videre utvikling.



Dette er ett bilde at topp tårnet på prototypen. Topptårnet består av RPi, wifimodul PS2 mottaker, Ultrasonisk sensor og RPi kamera modul.

1) Lilla sirkelen er kassen til RPi, her ligger kablingen som er dratt opp fra Atmega2560 skjult. Dette er hjernen og mottaker tårnet til hexapoden.
2) Grønne sirkelen er PS2 mottakeren
3) Rød sirkel er rundt Ultrasonisk og livefeed kameraet

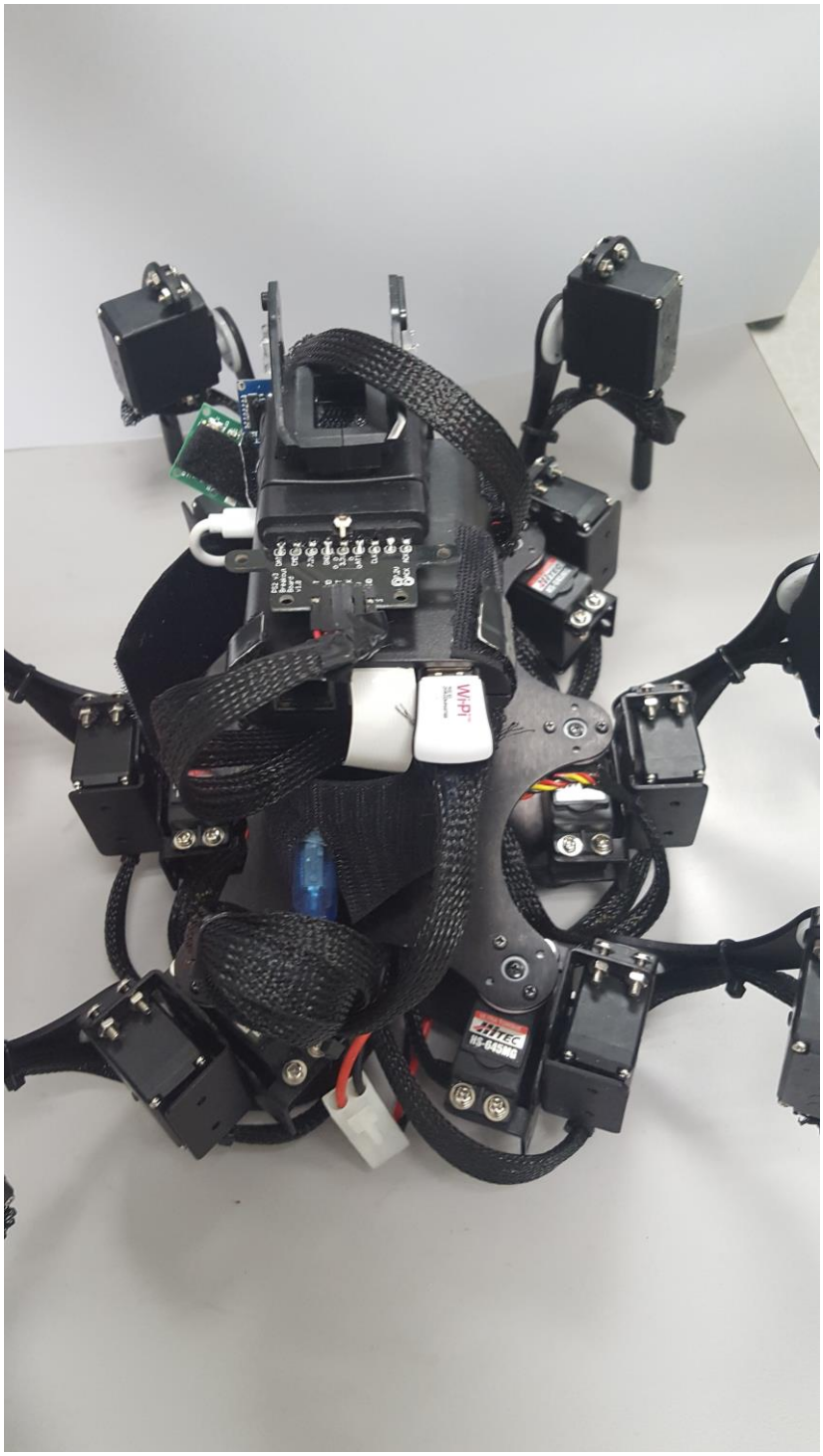


Dette er den fullstendige hexapod prototypen. Vi har dratt kablene gjennom kabelstrømper for å minske det fremhevede utseende av gule og røde kabler fra servoene. Vi har dratt en borrelås rundt for å skjule innmaten for estetiske grunner.

For å få tilgang til koblinger på Arduino drar man bare borrelåsen til siden og kobler. Om man skal koble på RPi er det bare å løfte av kassen RPi ligger i. For å programmere hexapoden plugger man bare USB inn i RPi eller Arduino som er åpent bak. I tillegg ligger batterier under i mellomrommet mellom de to platene som holder hexapoden sammen. Dette gjør hexapoden til en god prototype platform med lett tilgang til videre utvikling.

Figur 22 - Ferdigstilt Hexapod

Baksiden av hexapoden. Kan klart se at man har tilgang til programmeringsporter for RPi og Arduino.



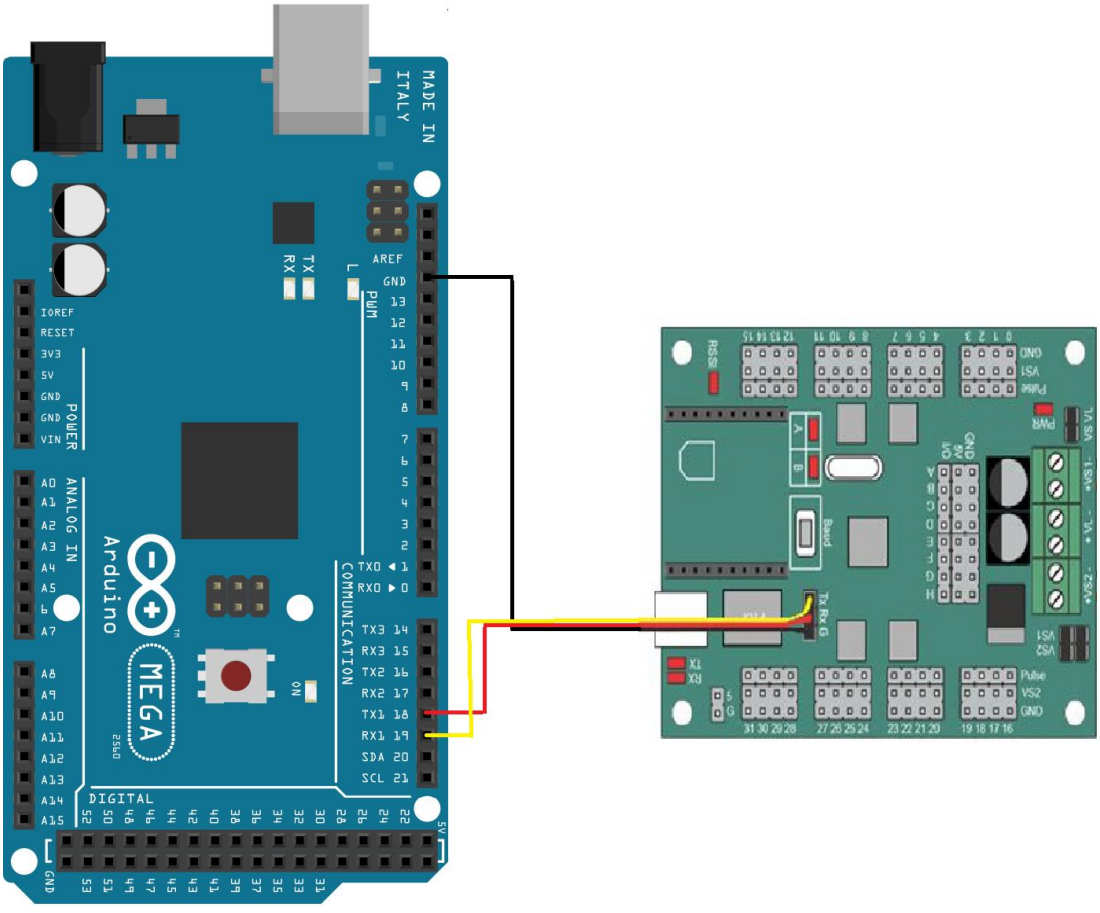
5.2.2 Kommunikasjon mellom Atmega2560 og SSC-32U

Kommandoer sendes fra SSC32U ved hjelp av Tx pinne, mens kommandoer som skal mottas av SSC32U gjøres via Rx pinne. Fra disse pinnene sendes kommandoer til servo kontrolleren fra Arduino Mega. For å gjøre dette, har vi koblet Tx pinne på Arduino til Rx pinne på

SSC32U, Rx pinne på Arduino til Tx pinne på SSC32 og GND til GND.

Tabell 4 - Arduino og SCC32U

Arduino Mega	SCC32U
TX	RX
RX	TX
GND	GND



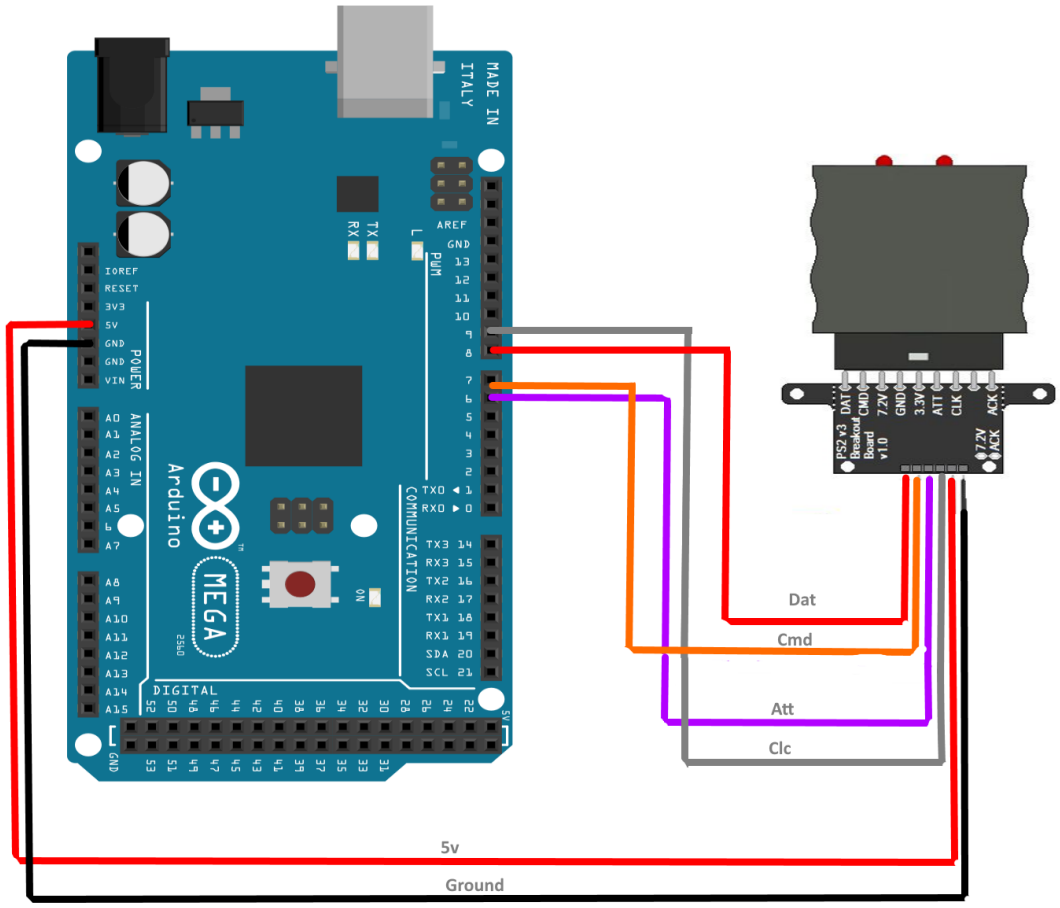
Figur 23 - Arduino mega og Scc32U

5.2.3 Koble sammen Arduino og PS2 mottaker sammen

Lynxmotion PS2 kontroller V4 er en 2,4 GHz trådløs Play Station 2 spillkontroller. Det inkluderer en liten mottaker modul som kobles til Arduino Mega

Tabell 5 - PS2 mottaker

Pinne konfigurasjon	PS2 mottaker
Pin 6	ATT
Pin 7	CMD
Pin 8	DAT
Pin 9	CLC
5v	+
GND	-



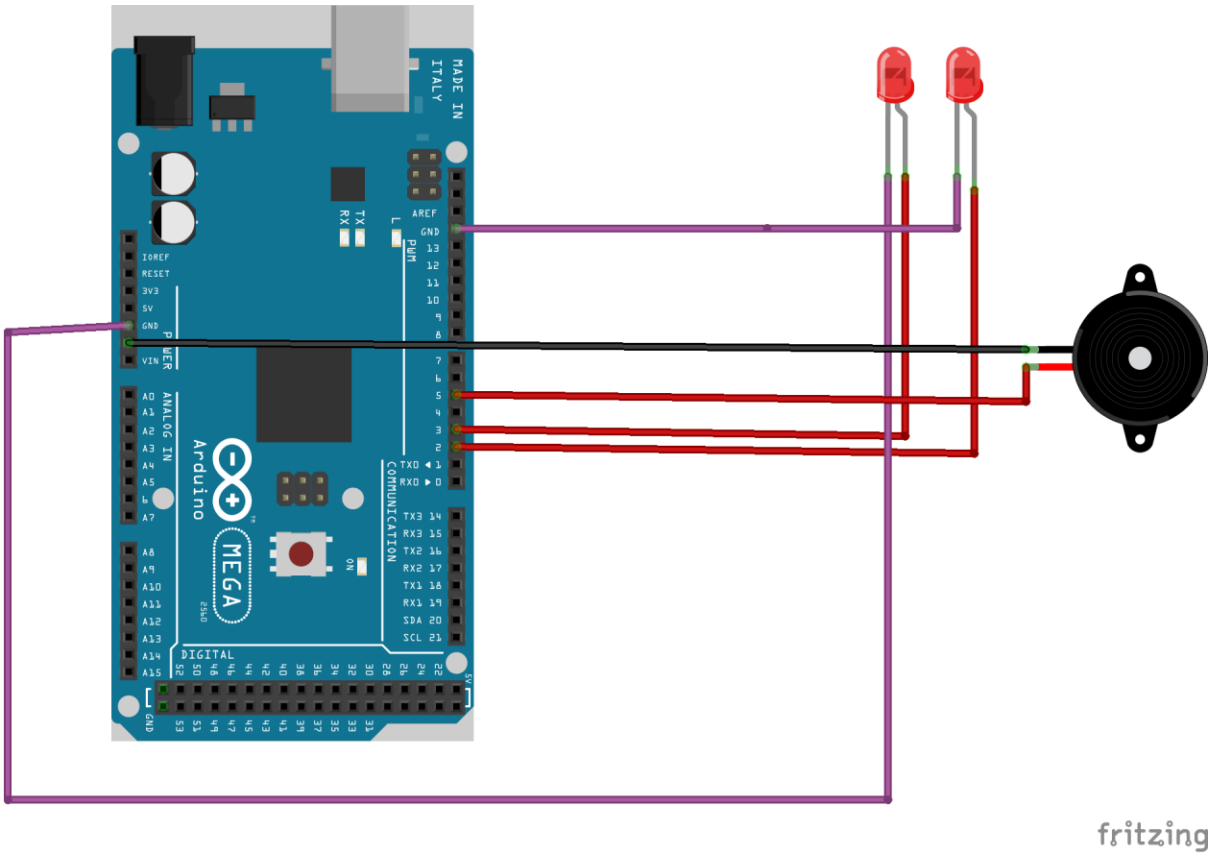
Figur 24 - PS2 mottaker

3.4 Komponenter

Vi har implementert 2 stk. led lys som forestiller robotens øye, og Buzz-er som gir beskjed når roboten er av og på.

Tabell 6 - Komponenter

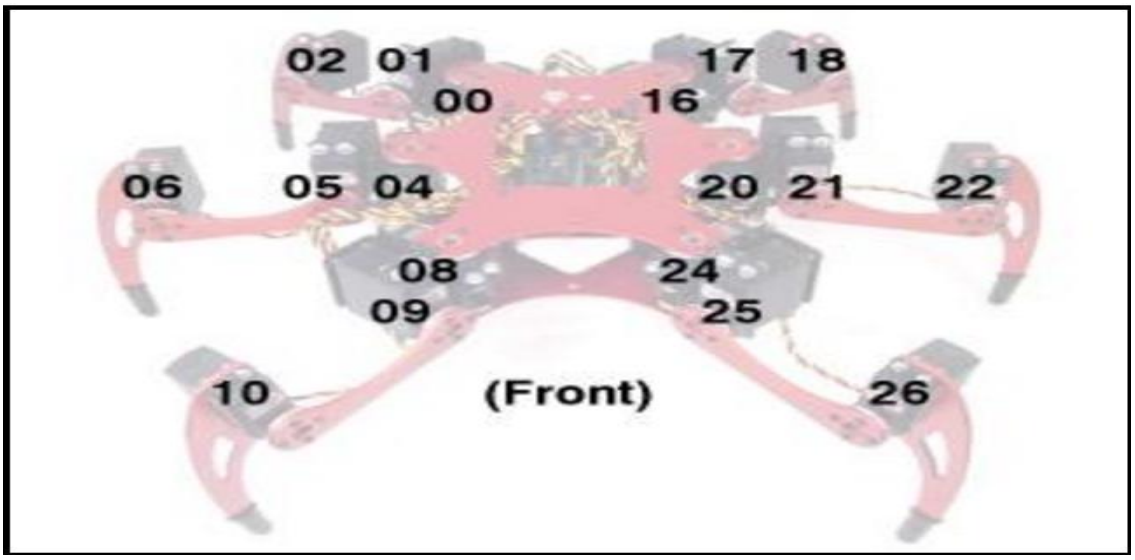
Pinne konfigurasjon	Komponent
Pin 2 / GND	Led1
Pin 3 / GND	Led2
Pin 5 / GND	Buzz-er

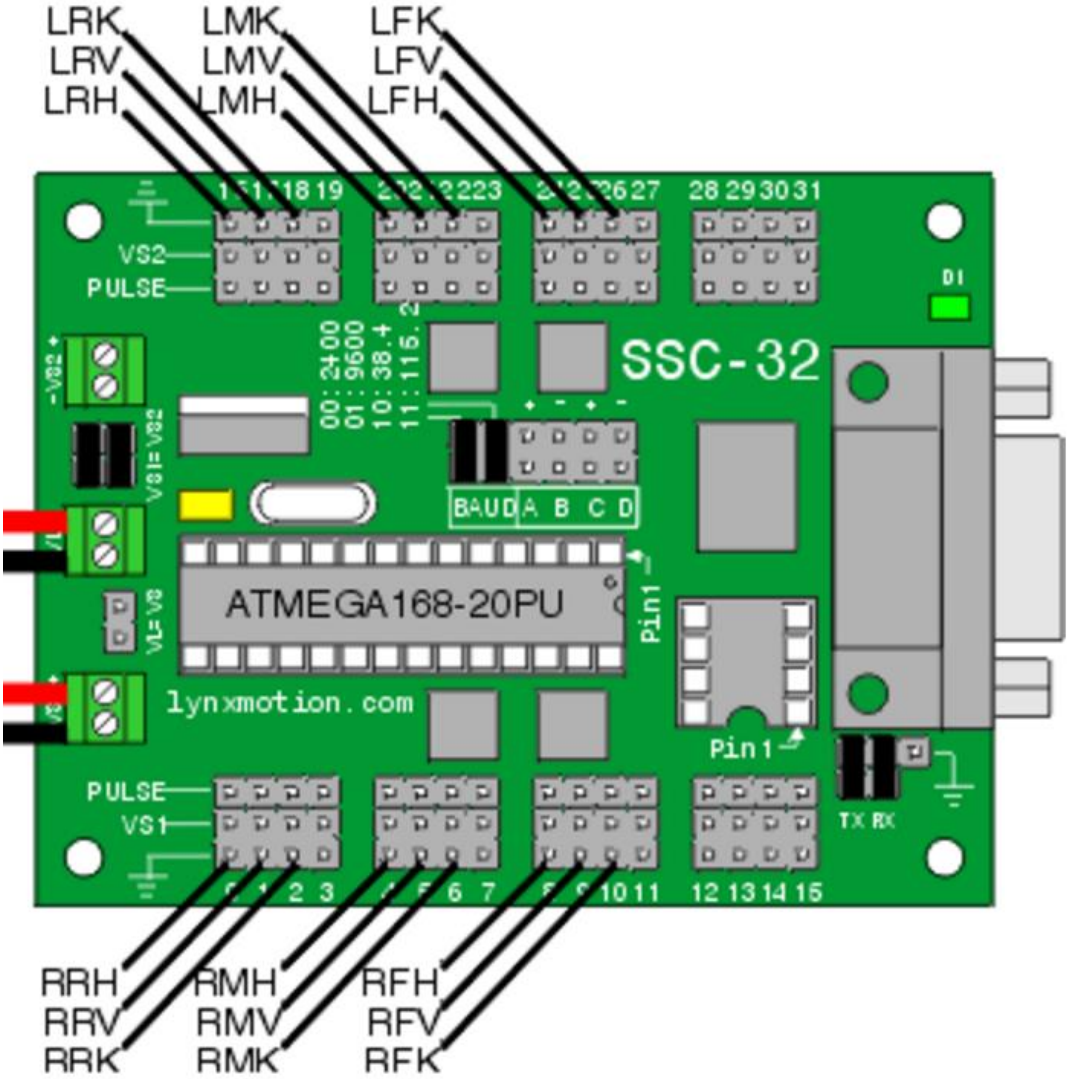


Figur 25 - led og Buzz-er

5.1.1 Kobling av servo til SCC32U

SSC32U er en servo kontroller som mottar og utfører kommandoer sendt til den fra Arduin o Mega. Ved hjelp av en servo kontroller frigjøres minne fra Arduino som ellers ville brukt til å oppdatere servo posisjonene.





Figur 26 - Servo pinne konfigurasjon

Tabell 7 - SCC32U VS1

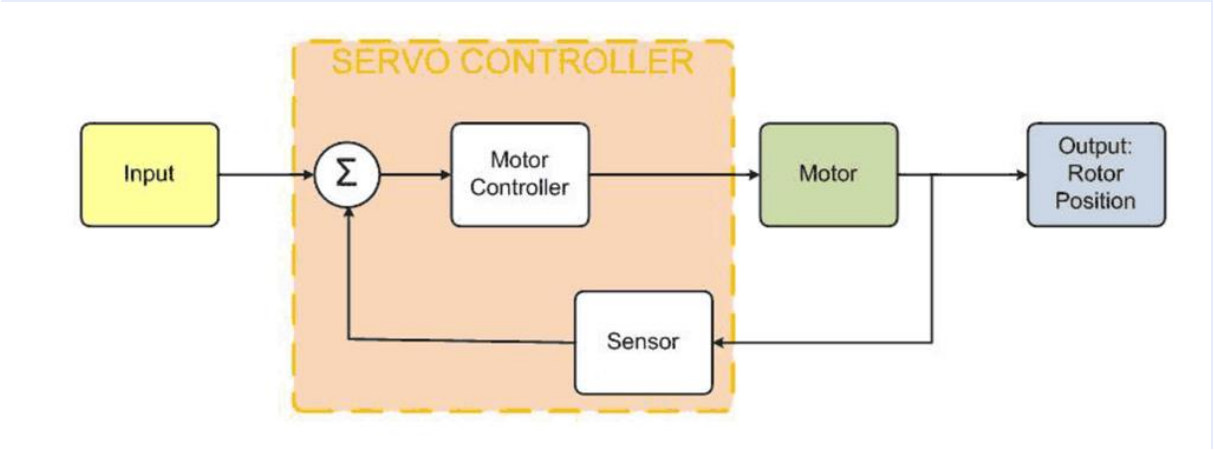
Leg	Servo	PULSE pinne	VS1	GND
Venstre bak	Innerst	0	+	-
Venstre bak	Midten	1	+	-
Venstre bak	Ytters	2	+	-
Venstre midte	Innerst	4	+	-
Venstre midte	Midten	5	+	-
Venstre midte	Ytters	6	+	-
Venstre foran	Innerst	8	+	-
Venstre foran	Midten	9	+	-
Venstre foran	Ytters	10	+	-

Tabell 8 - SCC32 VS2

Leg	Servo	Pulse pinne	VS2	GND
Høyre bak	Innerst	16	+	-
Høyre bak	Midten	17	+	-
Høyre bak	Ytters	18	+	-
Høyre midten	Innerst	20	+	-
Høyre midten	Midten	21	+	-
Høyre midten	Ytters	20	+	-
Høyre foran	Innerst	24	+	-
Høyre foran	Midten	25	+	-
Høyre foran	Ytters	26	+	-

5.3 Servomotor

En servomotor er en roterende aktuator som gir mulighet for nøyaktig kontroll av vinkel eller lineær posisjon, hastighet og akselerasjon. Den består av en passende motor som er koblet til en sensor for posisjonstilbakemelding. Det krever også en relativt sofistikert kontroller, i vårt tilfelle er SSC servokontroller vi har valgt å bruke, som er spesielt utviklet for bruk med servomotorer. Grunnen til vi valgte bruke servomotor, var at servomotor gir vinkel presisjon dvs. Det vil bare rotere så mye vi vil og deretter stoppe og vente på neste signal, det er en enkel DC motor som styrer vinkelrotasjon ved hjelp av lukket sløyfe feedback kontroll system. Servo motor er en spesiell typemotor automatisk opererer opptil viss posisjon for en gitt kommando ved hjelp av reguleringen sløyfen, som gir tilbake melding til den har nådd ønsket posisjon.



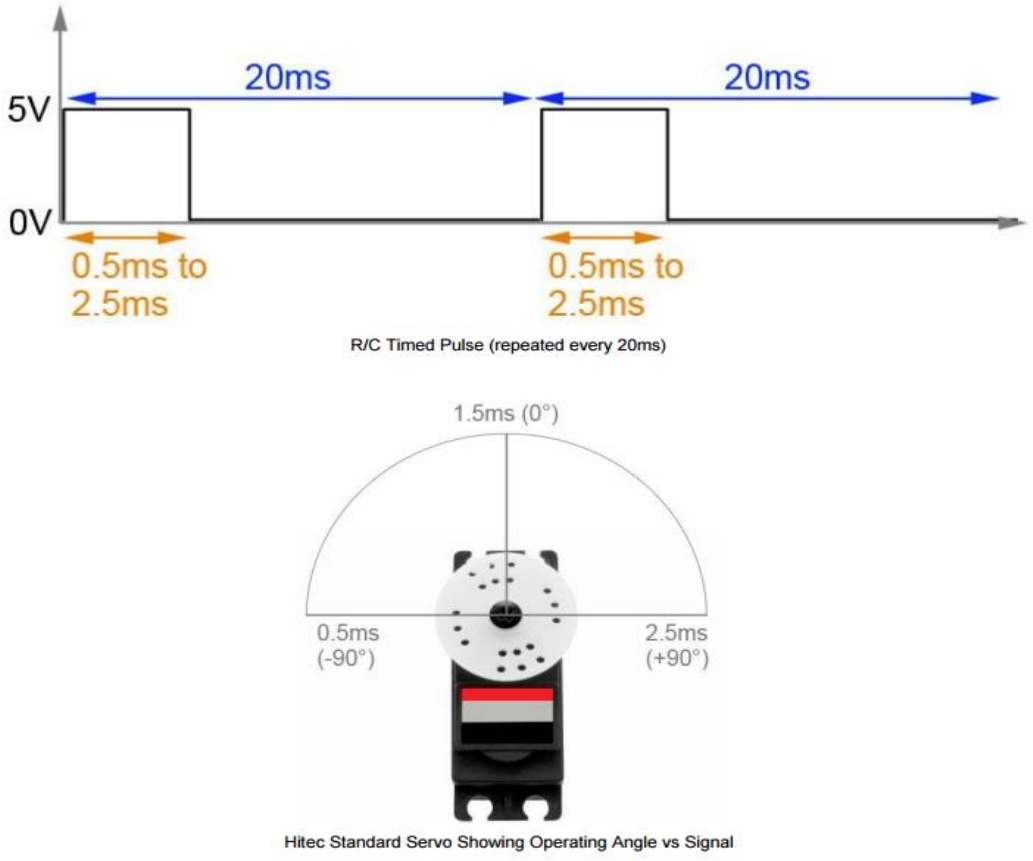
Figur 27 - Reguleringssløyfe for servomotor

5.3.1 Vår bruk av Servo

Servoer er kontrollert av pulsbreddemodulasjon(PWM), dette hvordan vi kontrollerer servoene våre.

Figur 11 viser ett generelt eksempel der en servo venter på ett gjentatt 5Volt signal, som blir timet for å se hvor lenge det forblir på. For de fleste servoer vil ett 0.5ms 5Volt signal gi en 90graders rotasjon av servo hodet. En 2,5ms lang 5V puls vil rotere servo hodet til +90grader, derfor vill en 1,5ms puls korrespondere med 0grader(sentrert). Pulsen må gjentas hvert 20ms for at den skal holde posisjonen.

Hadde servoene kunne flyttet seg 180grader ville de konstant kollidert. I Botboarduino koden definierer vi de mekaniske begrensningene til servoene ved å begrense PWM signalet slik at vi kan reduserer den fulle bevegelsen.



Figur 28 - PWM

5.3.2 Kalibrering av servomotor

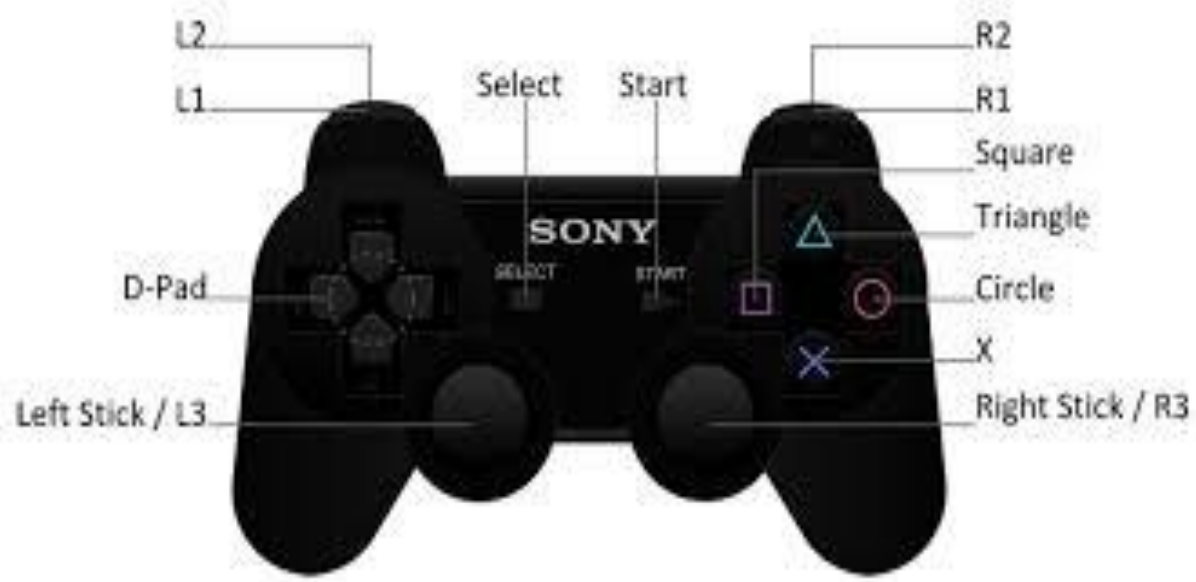
Servomotorer er aldri 100% sentrert, det ligger som oftest +- 5% feil inne som må kalibreres før servomotoren kan brukes i systemet. Kalibrering av servomotorer kan gjøres på mange forskjellige måter. Prosjektgruppen brukte SSC Servo Sequencer Utility Software. Denne programvaren var spesiellaget for servokontrolleren som gjorde det enkelt å kontrollere Lynxmotion SSC-32 Servo Controller til å eksperimentere og kalibrerer servoene. Med dette SSC Servo Sequencer Utility-program varen kunne vi enkelt flytte servomotorer, kalibrere sine posisjoner, lagre og spille av bevegelsessekvenser.

5.3.3 Ekstern kontroller

For å styre roboten valge vi en PlayStation 2(PS2) spillkontroll. PS2 kontrolleren er en 2,4 GHz trådløs kontroll med en rekkevidde på cirka 10 meter, to analoge styrespaker og 16 knapper som er tilgjengelige for robotstyring.

Figur 29 - Ekstern kontroller

5.3.4 Funksjoner til ekstern kontroller



Figur 30 - Kontroll funksjoner

Knapp	**Funksjoner**
Start	Slå av/på roboten
R3	Gå mode
L1	Slår på Shift modus
L2	Rotasjon/Danse modus
X	Slår på lagrde sekvenser
0 Sirkel	Slår på singel leg funksjon
[ ] Firekant	Øker hastigheten
Trekant	Setter roboten ned eller 35mm fra bakken
D-Pad opp	Body up 10mm
D-Pad ned	Body down 10mm
D-Pad venstre	Øker fart med 50 mS
D-Pad høyre	Synker speed 50mS

Gå modus kontroll	
Select	Endre gå modus
Venstre joystick	Gå rett frem bak over
Høyre joystick	Roterer samtidig som den går
R1	Gå hastighet
R2	Gå reiselengde
Shift Modus kontroll	
L1	Slå Shift-modus av
Venstre Joystick	Shift kroppen X/Z
Høyre Joystick	Shift og rotere kroppen Y
Rotasjon modus kontroll	
L2	Slå Rotasjon Modus av
Venstre Joystick	Roterer Kroppen X/Z
Høyre Joystick	Roterer Kroppen Y
Enkelt leg modus kontroll	
O Sirkel	Slå enkelt leg modus av
Select	Bytter ben
Venstre Joystick	Flytt ben X/Z
Høyre Joystick	Fytte ben Y
R2	Hold/sliper ben stilling
Sekvenser modus kontroller	
X	Slår av sekvens kontroll
Select	Bytter Sekvenser
R2	Starter sekvenser

5.4 Programmering av Hexapod

Valget av en hexapod kommer av at vi trengte en prototype plattform som har mulighet til å bevege seg i ulent terreng og holde diverse sensor utstyr for å ta eventuelle målinger. Hexapoden har 3grader av frihet med høy-moments hobby motorer per bein, HS-645MG. Dette gjør at den kan bevege seg i alle retninger med relativt komplekse bevegelses mønstre så lenge programmeringen er god nok.

Hexapoden er programmert i Arduino [3] programvaren, som er vårt valg av mikrokontroller, denne bruker C/C++. Sammen med en SSC(servokontroller) og har ferdig firmware for styring av servoer, SSC utfører ikke beregninger men holder «styr» på servoene til hexapoden. SSC-utility[4] programvare brukes til å kalibrere servoer.

Arduino

Mikrokontrolleren kan utføre sekvenser av instruksjoner og er hjernen som kalkulerer bevegelse eller servoposisjoner fra de verdiene hexapoden får. Arduino mikrokontrollere utfører beregninger, kommunikasjon og avgjørelser. Mikrokontrollere trenger ikke ekstra ram eller lagringskomponenter for å fungere, de er lette og billige til prototype plattformer, selv om de er svakere enn en PC.

Generelt om Hexapod koden:

Pheonix hexapod koden vi bruker er orginalt skrevet av Jeroen Janssen (Xan) for Lynxmotion Phoenix (Botboarduino) prosjektet. Som igjen er basert på den orginale Phoenix roboten av Kåre Halvorsen (Zenta). Koden er opprinnelig laget i Basic til å kjøre på Basic Atom Pro. Mye av original softwaren har vert basert av Zentas Excel Ark [1]. Kurt Eckhardt (KurtE) portet videre koden arduino (C/C++) programspråket med hjelp av Xan og Zenta, Kurts [2] Phoenix kode er en standard kjøreplattform for de fleste typer av robot «krabbere», koden er PhantomX. Og er laget for Botboarduino mikrokontroller platformen.

Linken til KurtEs github ligger i linken [1], PhantomX i linken [6], deler til forskjellige hexapod builds i Phoenix [7]. Alternativt kan basis plattform for hexapod anskaffes ved bruk av Powerpod programvaren, som auto genererer dette.

K-Spider kjører [5] CH3R_PS2 som er ett derivat av Kurtes deler. Spesielt laget til Lynxmotions Botboarduino, som er en spesiallaget hybrid arduino og lastes som «Arduino Duemilanove eller Diecimila At mega328». Vi bruker Atmega2560 som vår arduino plattform, Botboarduino er ikke lenger kompatibel med K-spider. Kspider koden ligger i vedlegg [1, 2, 3, 4, 5, 6, 7].

5.4.1 Fil oppsett

For at Hexapoden skal fungere må følgende filer være tilstede I folderet. Navnet på folderet må matche .ino filen for at koden skal kunne lastes sammen med bibliotekene.

Kode hierarki:

Kspider_CH3R_PS2_v1.ino

Hex_Cfg.h

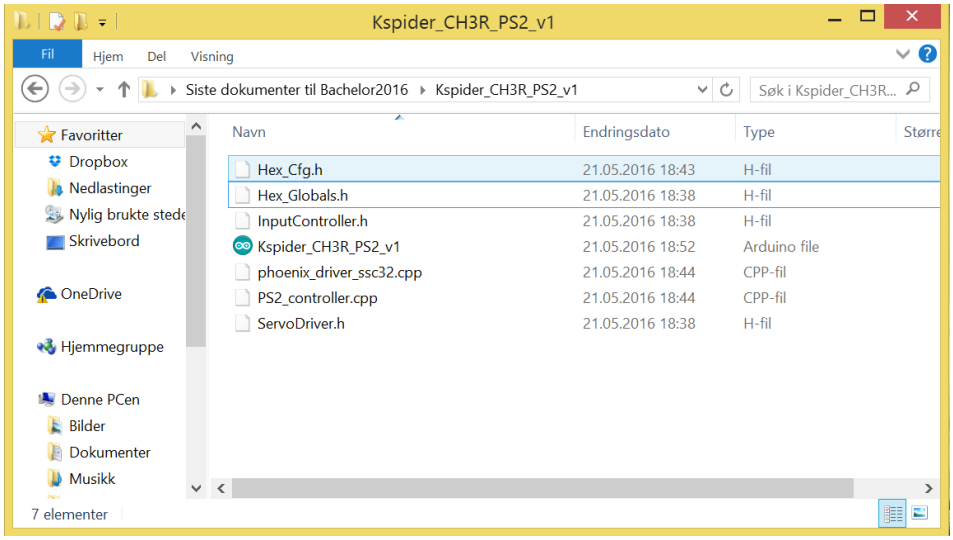
Hex_Globals.h

InputController.h

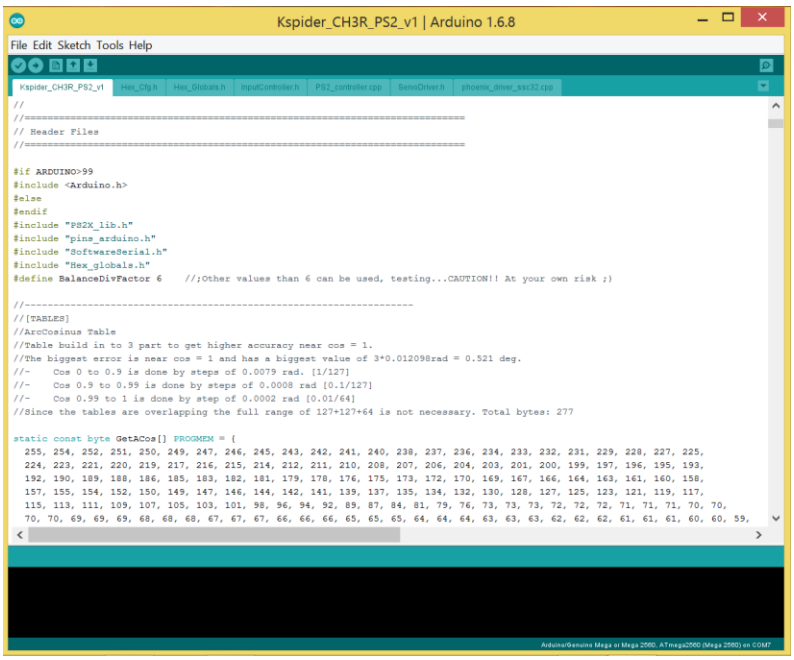
Phoenix_driver_ssc32.cpp

PS2_controller.cpp

ServoDriver.h



Oppsettet skal se slik ut inne i folderet før man åpner koden.



Dette blir resulterende utseende i arduino programmeringsverktøyet hvis du har gjort det riktig. Merk at at «Atmega2560» er valgt i Tools -> Processor. Bruk av seriell port(monitor) fra data krever at du har valgt Tools -> Port -> velg riktig USB seriell port

NB: merk at filen er .ino og IKKE .pde (Ikke bruk gamle bibloteker/filer da disse ofte ikke er støttet)

5.4.2 K-spider med Atmega2560:

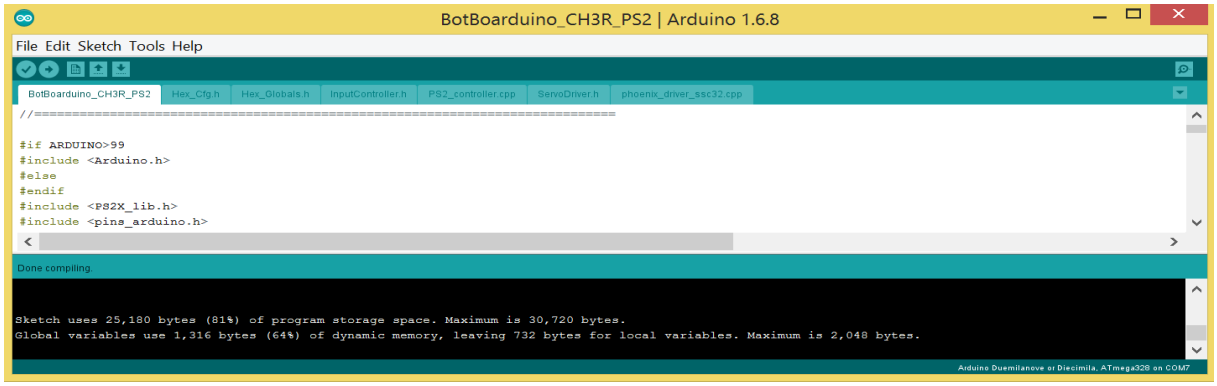
I begynnelsen av programmerings perioden begynnte vi med Botboarduino ettersom dette var den letteste plattformen og starte på, med gode pinner for strøm, koblinger og innebygd feilsøking.

Hoved grunnen til at vi ikke begynte med å lage egen kontroller var pro blemet med plass, uten endringer brukte basis koden til hexapoden 81% av minne, og 64% av Dynamisk. Dette ga altfor lite plass til en ekstra kontroller, vi brukte dermed tiden på å ta over kontrollen av den allerede anlagte PS2 kontrolleren.

Når vi endelig hadde ferdigprogrammert hexapoden til å kunne kontrolleres var minne kritisk lavt. Både dynamisk og fysisk minne var opptil 95% fullt, som fører til stabilitetsproblemer.

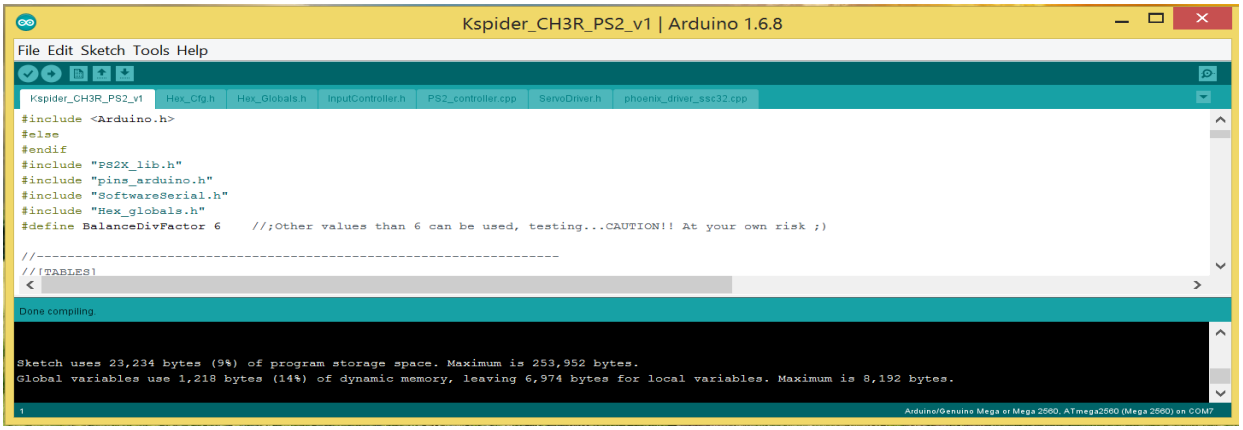
Det var heller ingen mulighet til å forbedre hexapodens funksjoner som den var. Vi valgte å bytte plattform til fordel for en kraftigere mikroprosessor. Dette vil gi mange flere muligheter på arduino siden av systemet i framtiden.

Figuren viser Botboarduino lastet uten enderinger i koden.

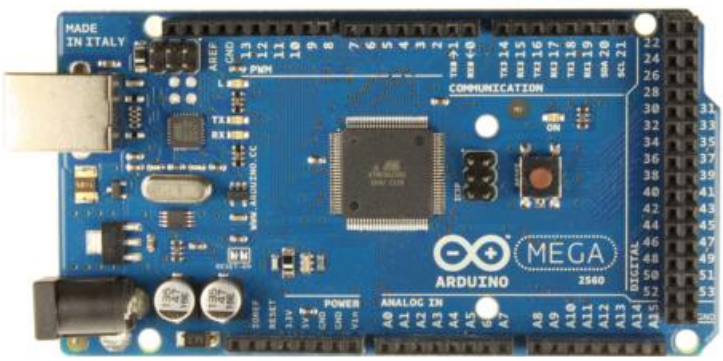


Vi valgte å porte hexapoden over til Atmega2560, dette innbærer endringer i bibloteker, pinn-numre, serielle koblinger og eventuelle funksjoner som ikke er kompatible med Atmega2560.

Figuren viser K-spider koden lastet med seriell kontroller. Prosessor kraften er 16Mhz (mot 8Mhz), bruker kun 9% fysisk minne og 14% av dynamisk minne.



Det er en hake ved bruk av Atmega2560, særlig i hexapods. Den har ett generelt dårligere interface for roboter enn Botboarduino, som nybegynnere var det greit at vi kunne få starte på en plattform som var tilpasset og ferdigprogrammert til testing. Men til syvende og sist velger vi funksjonalitet og ett bedre system. Bruk av SSC gjør dette mulig på grunn av mengden PWM signalpinner.



5.4.3 Bruken av PS2 kontroller (PS2_controller.cpp):

Potensiometrene i PS2 kontrolleren har en max verdi på 1023, når vi håndterer signalet fra PS2 kontrolleren har vi to valg:

1) Vi kan enten nedgradere oppløsningen fra 0 til 1023 til 0 til 255 isteden, som kan bli plassert i ett byte.
2) Alternativt kan vi behandle verdiene som bits, 1024 kan bli representert som 10bits (2^10). Som kan bli sendt som to pakker. Og når de ankommer klientsiden kan vi kombinere disse som ett tall.

Grunnen til dette er at de serielle pinnene til arduino kun kan sende 0-255(altså 8bits) hver gang vi bruker Serial.write() funksjonen, eller tilsvarende funksjoner til å sende eller mota data. Det å sende to bytes tar mer plass og tid enn ett byte, og vi trenger ikke nøyaktigheten til 0-1023(10bits), særlig når vi senere skal bruke dette i en seriell kobling mellom data enheter, vi foretrekker kjappere respons og bruker alternativ 1.

Vi har brukt PS2 kontrolleren til debugging og testing av systemet, den har vert en vital del som har passet på at vi har fått riktig kontroll og riktig bevegelsesmønster i hexapoden. Vi har brukt PS2_controller.cpp som utgangskontroll og koblet oss opp til funksjonene der istedenfor å lage en fullstendig ny Serial_controller.cpp, dette er igjen fordi vi ikke hadde nok plass på arduinoe n starten av prosjektet.

Oppretting av Seriell kontroll for hexapoden:

Vi har opprettet en seriell kobling med hastighet på 38400Baud som er vår valgte kommunikasjonshastighet mellom komponenter i systemet.

.begin oppretter seriell kommunikasjon med den valgte porten og hastigheten.

```
void setup() {

  int error;

  USBSERIAL.begin(38400);
  SSCSerial.begin(38400);
   // Pixy.begin(38400); Arduino /w pixy not present
  //Serial.println("38400 Baud Setup");
```

(Kspider_CH3R_PS2.ino)

```
// Which type of control(s) do you want to compile in
//#define DBGSerial         Serial
// #define Pixy              Serial2 //rx17 tx16
#define USBSERIAL         Serial // 1 0 USBport
#if defined(UBRR1H)
#define SSCSerial         Serial1 // rx19 tx18
#else
#endif

#define USEPS2

//____________________________________________________________________
```

(Hex_cfg.h)

Vi har definert USBSERIAL som Serial, dette er USB porten til Atmega2560. Mega har opptil 4 serielle porter.

Vi oppretter en int Rcu_Input som skal lese av USBSERIAL porten vi tidligere opprettet, med arduinos .read funksjon. Vi må bruke denne til å ta imot signaler fra systemet siden det vi skal hente informasjon fra ikke er analogt men digitalt.

```
void InputController::ControlInput(void)
{

    // Then try to receive a packet of information from the PS2.
    // Then try to receive a packet of information from Serial.
    ps2x.read_gamepad();          //read controller and set large motor to spin at '
    int Rcu_Input = USBSERIAL.read(); // Reads serial in ascii, you can deduct -'0'

    // Wish the library had a valid way to verify that the read_gamepad succeeded...
    if (((ps2x.Analog(1) & 0xf0) == 0x70)) { //7bit value
```

(PS2_controller.cpp)

5.4.4 Digitale hexapod funksjoner:

I og med at arduino monitor leser verdier separert, blir bruken av hex eller binære verdier ikke bare umulig uten annen seriell software, men også veldig upraktisk. For å få en enkel funksjon i arduino til å respondere på ett serielt signal må vi sende noe som vi kan teste med fra ett data interface, ascii er det greieste for dette.

Inne i koden må vi definere verdien vi skal motta som ett ascii tall, i dette tilfellet er Rcu_Input==82. Dette tilsvarer 52 i Hex, eller R som symbol. Vi tester funksjonen ved å sende R gje nnom serial monitoren til arduino softwaren og får aktivert funksjonen.

```
//[Walk functions]
if (ControlMode == WALKMODE) {
    //Switch gates
    if ((ps2x.ButtonPressed(PSB_SELECT)            // Select Button Test
            && abs(g_InControlState.TravelLength.x)<cTravelDeadZone //No movement
            && abs(g_InControlState.TravelLength.z)<cTravelDeadZone
            && abs(g_InControlState.TravelLength.y*2)<cTravelDeadZone  )|| Rcu_Input==82) {
              //Serial.println("Select WalkMode");
```

(PS2_controller.cpp)

Ascii [8] tabell. Vi bruker caps symboler fra A til Z i programmeringen, dette er lagt in som ascii i koden, fra 65 til 90.

ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
0	0	NUL	16	10	DLE	32	20	(space)	48	30	0
1	1	SOH	17	11	DC1	33	21	!	49	31	1
2	2	STX	18	12	DC2	34	22	"	50	32	2
3	3	ETX	19	13	DC3	35	23	#	51	33	3
4	4	EOT	20	14	DC4	36	24	$	52	34	4
5	5	ENQ	21	15	NAK	37	25	%	53	35	5
6	6	ACK	22	16	SYN	38	26	&	54	36	6
7	7	BEL	23	17	ETB	39	27	'	55	37	7
8	8	BS	24	18	CAN	40	28	(	56	38	8
9	9	TAB	25	19	EM	41	29	)	57	39	9
10	A	LF	26	1A	SUB	42	2A	*	58	3A	:
11	B	VT	27	1B	ESC	43	2B	+	59	3B	;
12	C	FF	28	1C	FS	44	2C	,	60	3C	<
13	D	CR	29	1D	GS	45	2D	-	61	3D	=
14	E	SO	30	1E	RS	46	2E	.	62	3E	>
15	F	SI	31	1F	US	47	2F	/	63	3F	?

ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
64	40	@	80	50	P	96	60	`	112	70	p
65	41	A	81	51	Q	97	61	a	113	71	q
66	42	B	82	52	R	98	62	b	114	72	r
67	43	C	83	53	S	99	63	c	115	73	s
68	44	D	84	54	T	100	64	d	116	74	t
69	45	E	85	55	U	101	65	e	117	75	u
70	46	F	86	56	V	102	66	f	118	76	v
71	47	G	87	57	W	103	67	g	119	77	w
72	48	H	88	58	X	104	68	h	120	78	x
73	49	I	89	59	Y	105	69	i	121	79	y
74	4A	J	90	5A	Z	106	6A	j	122	7A	z
75	4B	K	91	5B	[	107	6B	k	123	7B	{
76	4C	L	92	5C	\	108	6C	l	124	7C	|
77	4D	M	93	5D	]	109	6D	m	125	7D	}
78	4E	N	94	5E	^	110	6E	n	126	7E	~
79	4F	O	95	5F	_	111	6F	o	127	7F	

5.4.5 Analoge hexapod funksjoner:

PS2 kontrolleren vil gi signaler mellom 0-255 bits, verdien vi får inn til arduino er 128 ved utgangsposisjon (midt posisjon for potensiometeret), og teller opprinnelig opp til 255, og ned til 0.

Ved å trekke fra -128 i (ps2x.Analog(PSS_RY)-128), setter vi utgangsposisjon(midtstilling) til potensiometrene til 0. Dette er fordi verdiene vi får inn skal brukes til å beregne flere ting. Arduino formaterer altså de nye signalene til å være mellom -127, til 127. Der 0 er senter posisjon for de potensiometerene som er joystickene til PS2 kontrolleren, og vil null bevegelse.

X, Y, Z og Shift aksene til joysticken følger disse verdiene.

Vi lager nå «if» setninger som passer inn i de signalene basis koden er laget til å takle. Hvilk en tall systemet responderer på kommer ann på hvordan Cos, Sin og Tan beregnes inne i hovedkoden. Vi skal ikke jobbe å endre på basiskoden, men det er greit å vite. Hva som skjer er at systemet konstant bruker disse input verdiene til å beregne kroppens og beinenes posisjoner, spesielt invers kinematikk(IK). Det som er viktig er at vi bruker riktige verdier så vi forstyrrer minst mulig av den eksisterende plattformen. Vi bruker derfor de allerede beregnede max verdiene fra TravelLength.z/x/y som er forventet fra kontrolleren.

```
//Walking
if (WalkMethod)  //(Walk Methode)
    g_InControlState.TravelLength.z = (ps2x.Analog(PSS_RY)-128); //Right Stick Up/Down

else {
    g_InControlState.TravelLength.x = -(ps2x.Analog(PSS_LX) - 128);
    g_InControlState.TravelLength.z = (ps2x.Analog(PSS_LY) - 128);
}

if (!DoubleTravelOn) {  //(Double travel length)
    g_InControlState.TravelLength.x = g_InControlState.TravelLength.x/2;
    g_InControlState.TravelLength.z = g_InControlState.TravelLength.z/2;
}


              g_InControlState.TravelLength.y = -(ps2x.Analog(PSS_RX) - 128)/4;  //Right Stick Left/Right

if (Rcu_Input == 65) {g_InControlState.TravelLength.z = -64;}  // ascii A (Z - Left Stick Up) Forward
if (Rcu_Input == 66) {g_InControlState.TravelLength.z = 63;}   // ascii B (Z - Left Stick Down) Backward
if (Rcu_Input == 67) {g_InControlState.TravelLength.x = -63;}  // ascii C (X - Left Stick Left) Walk Left
if (Rcu_Input == 68) {g_InControlState.TravelLength.x = 64;}   // ascii D (X - Left Stick Right) Walk Right
if (Rcu_Input == 71) {g_InControlState.TravelLength.y = 32;}   // ascii G (Y - Right Stick Left) Rotate Left
if (Rcu_Input == 72) {g_InControlState.TravelLength.y = -32;}  // ascii H (Y - Right Stick Right) Rotate Right

 }
```

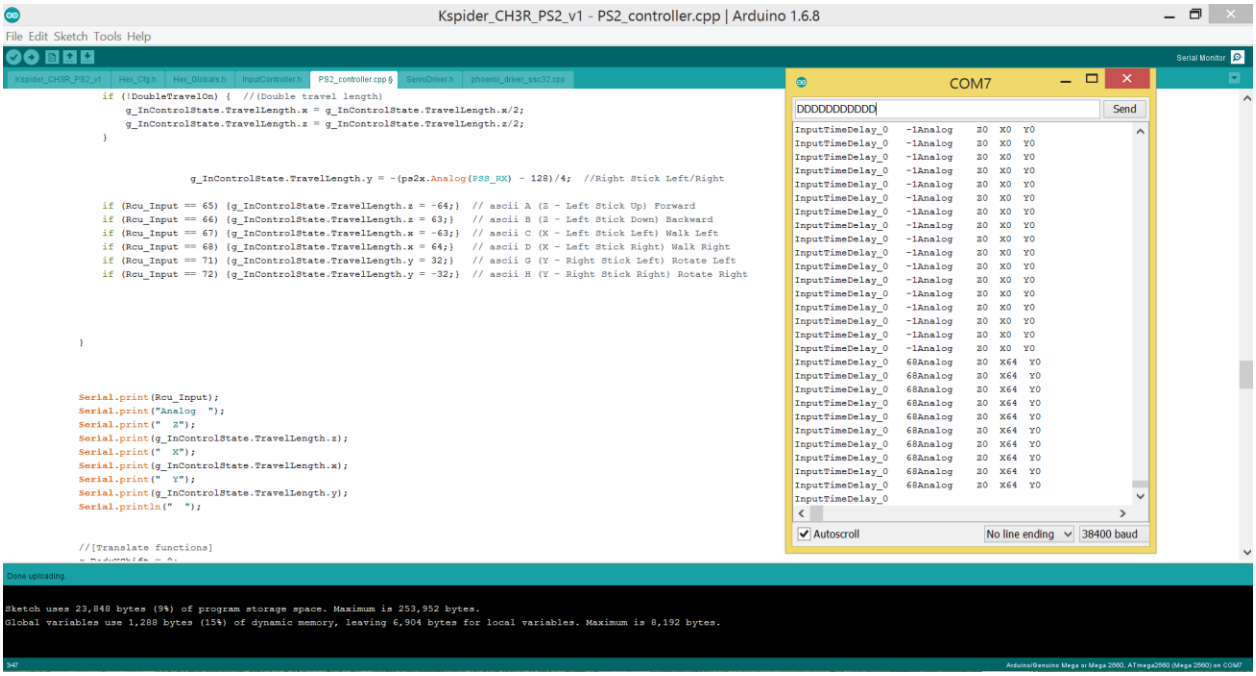
For Z, X og Y i dette tilfellet har vi funnet verdiene som tilsvarer max input bevegelse for en viss seriell input verdi, vi skriver verdien inn i funksjonen som brukes av hexapoden til å bevege seg når vi aktiverer funksjoner via A, B, C, D, G eller H. Hvilken verdi som må brukes kommer ann på hvordan denne bevegelsen blir beregnet innad i hexapoden.

Siden kontrollen skal være pakkestyrt ønsker vi ikke inkrementell økning i pådrag for hexapoden, vi vil kun ha en max kontinuerlig verdi som gir oss en konstant linær estimerbar bevegelse.

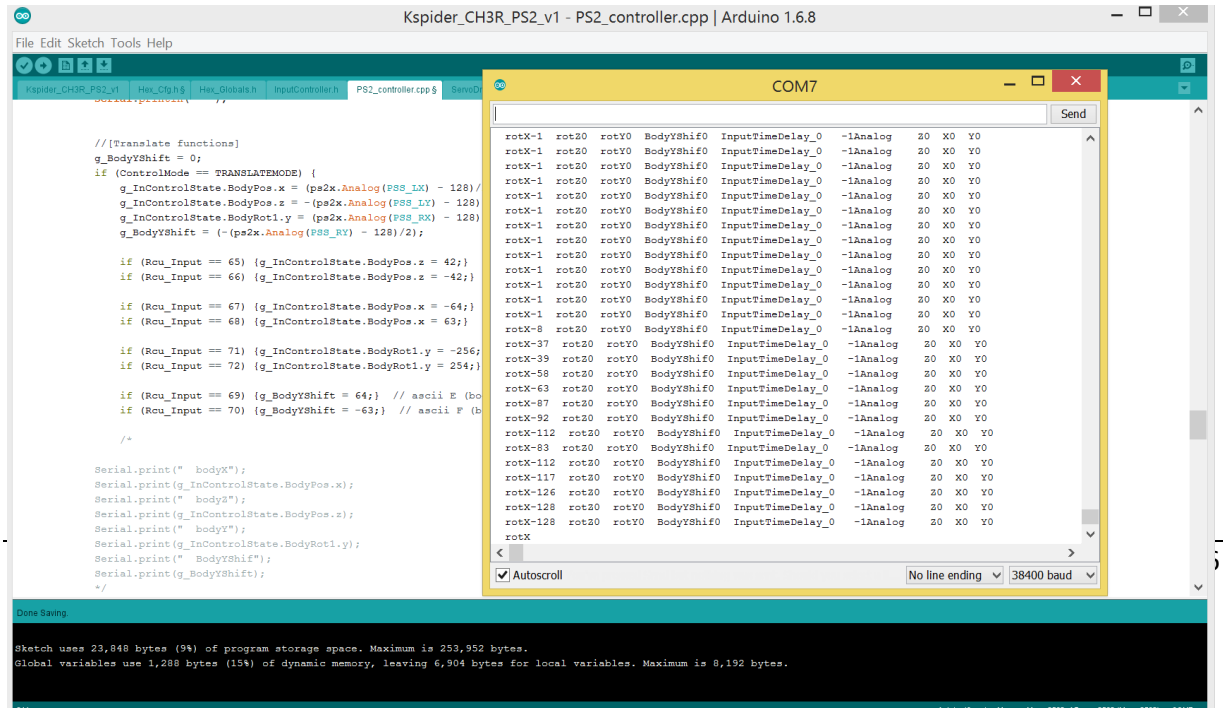
Siden vi skal styre hexapoden via datakommandoer er det best vi lager to funksjoner per bevegelses akse, dette gjør at vi ikke krever at kontrollen er analog på noe vis. Dette vil også gjøre at interfacet for hexapoden er standardisert, som betyr at ved å produsere ett lignende interface til en annen

enhet med samme ascii verdier for funksjoner, så kan vi ta kontroll over enheten relativt hurting. Hexapoden blir dermed utbyttbar.

Her er det mulig å se kontrollen der vi sender D. Arduino responderer med å skrive X til 64, og vi får side bevegelse til høyre. Starten og slutten på signalet avsluttes abrupt, bevegelsen er alltid konstant.



I tilfellet av denne analoge bevegelsen ser vi PS2 kontrolleren øker pådraget ved økende påtrykk på joysticken. Alternativet for oss serielt er å bruke W og V til å øke eller senke hastigheten til hexapoden. Hastighet kan også endres permanent i Kspider_CH3R_PS2_v1.ino: ServoMoveTime = 200 + g_InControlState.SpeedControl; (ved å gjøre 200 mindre øker hastigheten, og omvendt)



Seriell signalstyrke

Tallene som kommer opp I COM7 serial monitor vinduene er utskrifter fra kontroll verdiene inne i koden, vi har laget disse til å feilsøke problemer og finne riktige verdier. Noe som man bør legge merke til er at InputTimeDelay konstant står på 0.

```
//Calculate walking time delay
g_InControlState.InputTimeDelay = 128 - max(max(abs(ps2x.Analog(PSS_LX) - 128), abs(ps2x.Analog(PSS_LY) - 128)), abs(ps2x.Analog(PSS_RX) - 128));
g_InControlState.InputTimeDelay = (g_InControlState.TravelLength.z, g_InControlState.TravelLength.x, abs(g_InControlState.TravelLength.y)); //Setting Serial delaytime to 0


Serial.print("InputTimeDelay_");
Serial.print(g_InControlState.InputTimeDelay);
Serial.print("   ");
```

Walking time delay (input time delay) er praktisk for en håndkontroller fordi potensiometerene ikke vil stå 100% rett ved skader eller småfeil. Ett skjevt potensiometer fører til at man konstant får ett lavt signal som kan føre til at hexapod beveger seg uten input fra bruker. Denne var engang programmert som konstant 128 helt til man ga nok påtrykk på kontroller, som førte til at inputdelay minsket etterhvert som pådrag ble høyere. Dette fører til at bevegelses inputen til slutt overkommer inputdelayet eller «statisk programmert friksjon» som gjør at den beveger seg.

Siden vi ikke bruker PS2 kontrolleren (men funksjonene) så står denne delay typen i veien for oss. Våre signaler med denne typen beregning gir oss en sein og ubrukelig respons, enkelt og greit fordi vi ikke regner våre inputs inn i denne setningen. Vi lager vår egen til å overkomme inputdelay, vi trenger ikke denne funksjonen siden en kommando fra systeme t skal være absolutt.

g_InControlState.InputTimeDelay = (g_InControlState.TravelLength.z, g_InControlState.TravelLength.x, abs(g_InControlState.TravelLength.y));

Gjør at hexapoden reduserer input time delay til 0, dette fører til at vi via seriell kontroll får en jevn og god bevegelse uten forstyrrelser. Men som tradeoff kan det utsette PS2 kontrolleren for problemer ved unøyaktigheter i joystickene, vi er kun interessert i PS2 som testkontroller så dette går bra.

Dette fjerner den feilen PS2 kontrolleren har hatt fra start, som er at den overestimerer bevegelsesrommet til beinene under rotasjon. Hexapoden blir skadet, som vist til veildere; om PWM mønsteret er for stort i forhold til størrelsen på rammen, henger beina seg fast i hverandre og blir rykket og revet etter hvert som hexapoden beveger seg. Med inputdelay som 0, fungerer seriell kommunikasjon. Vi får en bevegelse som ikke skaper kollisjoner av legemer, d ette er løsningen for seriell kontroll.

Det er lagt til kontroll kommandoer for alle funksjoner, R2 double travellength er fjernet ettersom denne er destruktiv for systemet slik PS2 kontrollerens snufunksjon er. Andre endringer og tilpassninger som ikke inngår i selve kontroll Interface og dens funksjon er uteblitt.

5.5 Raspberry Pi 2

For å automatisk starte det som må startes når RP2 starter opp, bruker vi crontab (terminal: «sudo crontab -e») til å starte Desktop/RunOnStart/run.sh. Det går også an å bruke autostart i home/pi/.config/autostart. Terminal: «bash /Desktop/RunOnStart/run.sh», starter det også, hvis det må startes manuelt.

Scriptet kan ikke kjøres to ganger oppå hverandre, så hvis du vil kjøre det må du sørge for at det ikke kjører. Ellers får du en feilmelding når du prøver å starte det. Hvis det autostartet kjører det i bakgrunnen. Da kan du rename «run.sh» til noe annet («_run.sh» for eksempel), og så restarte RPien. Da vil den ikke kunne autostarte det, og du kan kjøre det manuelt, for debugging.

run.sh inkluderer også en terminalkommando som starter opp kamera-live-feeden, og en som starter TightVNC serveren, slik at den kan kobles til.

For å koble til RP2 TightVNC, kjør det på en PC og koble til «192.168.1.123:8554». Passord er «password». Kan kanskje være en god ide å endre dette.

For å koble til live feed, åpne VLC Player, åpne nettverk stream (ctrl+n), koble til «rtsp://192.168.1.123:8554»

Mer info om bruk av systemet finnes i brukermanual eller RP2:Desktop/RunOnStart/Info.txt.

5.5.1 Kommandosettprotokoll

Når RCUen har ett sett med kommandoer den sender til sensorplatformen, gjøres det via flere subsystemer som kommuniserer med hverandre.

RCU sender ett sett med kommandoer i form av en string basert på et array. Eksempel:

('x_1', 'x_2', 'x_3', …, 'x_n')

Når RP1 (Raspberry Pi 1, på kontrollsystemsiden) mottar dette, fjerner den unødvendige tegn, og lager det til en string på følgende møte:

x_1:x_2;x_3:x_4;…;x_n-1:x_n

Hvor hver kommandoene blir splittet på ';'. Slik at hver kommando blir slik:

x_1:x_2

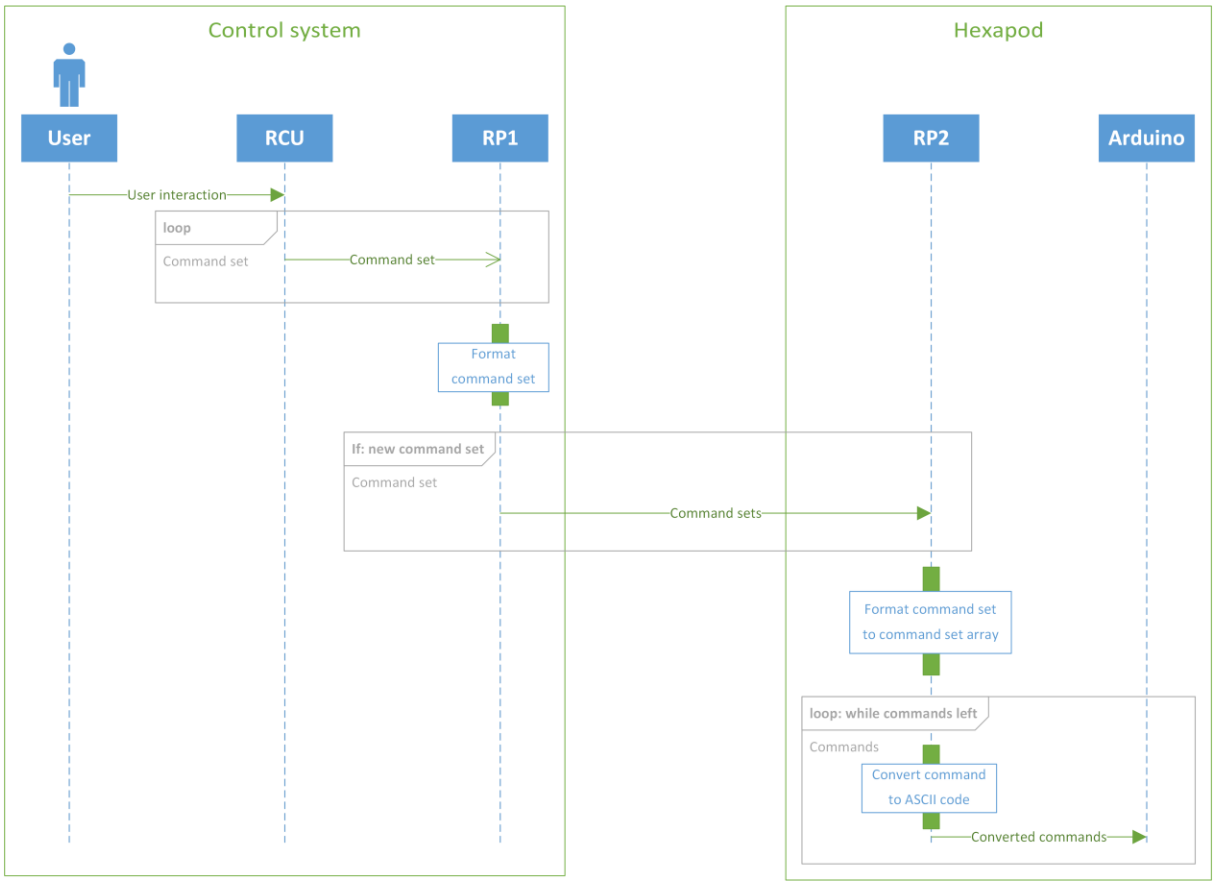
x_3:x_4, osv.

Det vil da si at annethvert element i kommandosettet blir før kolon, og annethvert etter kolon. Dette sendes så videre til RP2 (Raspberry Pi 2, på Hexapodsiden) via UDP-IP, gjennom routeren og det lokale nettverket.

RP2 mottar et kommandosett av gangen, og utfører kommandoene i settet en etter en. Den mottar i følgende format:

x_1:p_1;x_2:p_2;…;x_n:p_n

Hvor x-ene i hver kommando er funksjonskoder som representerer hva kommandoen gjør, og p-ene er eksekveringsparametere som forteller RP2 de skal gjøres.



Funksjonskodene, som finnes i «Hexapod Function Table», bestemmer hva slags kommando RP2 skal utføre. Dette er for eksempel «65» som betyr at hexapoden skal gå fremover, «80» som betyr at den skal reise seg opp 10mm, og «73» som betyr at den skal skru servoene av eller på.

Dec	ASCII	HEX	Button	Function description
65	A	41	L3 ↑	Move forward
66	B	42	L3 ↓	Move backward
67	C	43	L3 ←	Move left
68	D	44	L3 →	Move right
69	E	45	R3 ↑	Push up
70	F	46	R3 ↓	Push down
71	G	47	R3 ←	Rotate left
72	H	48	R3 →	Rotate right
73	I	49	Start	Start servos
74	J	4A	L1	Translate mode (dance)
75	K	4B	L2	Rotate mode (twist)
76	L	4C	O	Single leg mode
77	M	4D	X	GP-Player
78	N	4E	□	Balance mode
79	O	4F	△	Stand up / sit down
80	P	50	DPad ↑	+ 10mm height
81	Q	51	DPad ↓	- 10mm height
82	R	52	Select	Select modes
83	S	53	R1	Double lift height mode
84	T	54	R2	Double travel length mode (removed)
85	U	55	R3	Walkspeed mode
86	V	56	DPad →	Walk speed up
87	W	57	DPad ←	Walk speed down
88	X	58		
89	Y	59		
90	Z	5A		

Figur 31 - Hexapod funksjonstabell

I cmdReg.py (RP2:Desktop/RunOnStart/cmd_RP2/cmdReg.py) på RP2, som blir importert til RP2.py, ligger et register over kodene, som brukes til å konvertere funksjonskodene om fra RCU-RP1-RP2-format, til RP2-Arduino-format. Dette gjøres for å kunne sende enkeltkarakterer, da dette gjør kommunikasjonen med Arduinoen mye enklere. Her gjøres da «65» om til «A», og blir returnert sammen med en kode for funksjonstype. Dette lar RP2.py (RP2:Desktop/RunOnStart/cmd_RP2/RP2.py) vite hva slags funksjonskode det er. Så langt er det lagt til 3 typer, men flere kan legges til om nødvendig. Disse bestemmer hva RP2 skal gjøre med eksekveringsparameterne den mottar. De tre mulighetene er (hvor «x» er parameter):

«0» - send den konverterte funksjonskoden til Arduino èn gang, for så å vente x antall sekunder, hvor «x» er funksjonskoden sitt eksekveringsparameter. Eksempel: Starte hexapoden og vent 2 sekunder.

«1» - send den konverterte funksjonskoden til Arduino hele tiden, gjennom en viss tidsperiode («x» antall sekunder), et satt antall ganger i sekundet. Det er nå ca. 10 ganger per sekund, men dette kan endres, for å passe hexapoden. Eksempel: Hexapod gå frem i 15 sekunder.

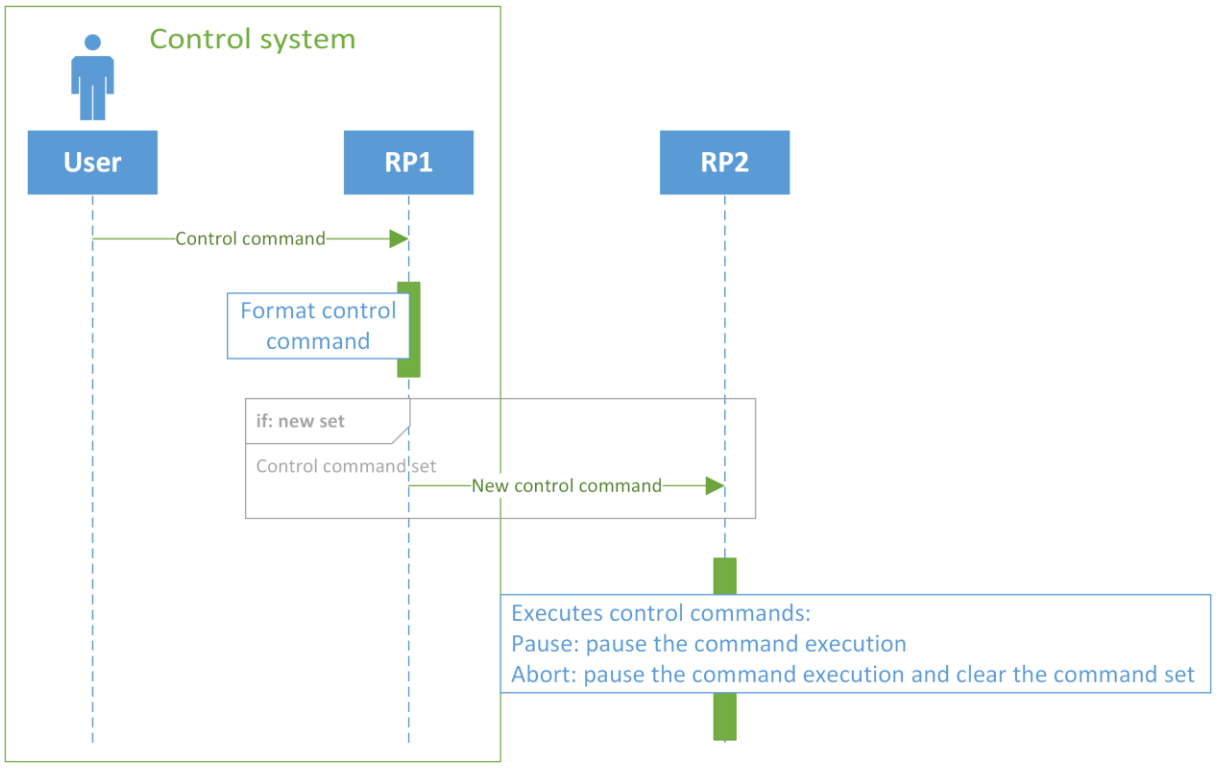
«2» - sende den konverterte funksjonskoden til Arduino «x» antall ganger. Eksempel: Hexapod reise seg opp 10 mm 3 ganger, i.e. 30 mm.

5.5.2 Kontrollkommandoer

Kontrollkommandoer er spesielle kommandoer som blir sendt parallelt og asynkront med de andre kommandoene. De er kommandoer til RP2 sin eksekvering. Kommandoer som «pause» og «abort» så langt.

Hvis RCUen sender setter «pause» til «1», sendes kommandoen til RP 1->RP2 å pauser eksekveringen av kommandosettene på RP2 inntil RCU setter pause til «0» igjen.

«Abort» gjør det samme som pause, men sletter også avbryter resten av kommandosettet. Dvs. at hvis du kjører abort pauser den til du skrur det av igjen, men fortsetter ikke med å utføre resten av kommandosettet som kjørte på RP2.



6 REFERANSER

[1] 21.05.2016, http://www.lynxmotion.com/images/html/proj098.htm

[2] 21.05.2016, https://github.com/KurtE

[3] 21.05.2016, https://www.arduino.cc/en/Main/Software

[4] 21.05.2016, http://www.lynxmotion.com/p-895-free-download-ssc-32-servo-sequencer-utility-created-using-flowbotics-studio.aspx 21.05.2016

[5] 21.05.2016, https://github.com/Lynxmotion/3DOF-4DOF-Hex/tree/master/BotBoarduino/3%20DoF%20Ready/BotBoarduino_CH3R_PS2

[6] 21.05.2016, https://github.com/KurtE/Phantom_Phoenix

[7] 21.05.2016, https://github.com/KurtE/Arduino_Phoenix_Parts/tree/master/Phoenix

[8] 21.05.2016, www.ascii.cl

[9] http://www.electrical4u.com/servo-motor-servo-mechanism-theory-and-working-principle/

[10] http://www.lynxmotion.com

[11] http://www.cvel.clemson.edu/auto/actuators/motors-servo.html

[12] http://www.robotshop.com/

[13] 12.02.2016, SSC servo utility, http://www.lynxmotion.com/p-895-free-download-ssc-32-servo-sequencerutility-created-using-flowbotics-studio.aspx

[14] Kurshåndbok fra Kongsberg Maritime

[15] Training Manual AIM2000

Høgskolen i Sørøst-Norge

KONGSBERG

TEKNOLOGIDOKUMENT

PROSJEKT	K-Spider		
OPDRAGSGIVER	Kongsberg Maritime AS		
UTFØRT VED	Høgskolen i Sørøst-Norge, avd. Kongsberg		
REVISJON	1.0		
MEDLEMMER	Aleksander Tokle Poverud, Masoud Shah Pasand, Tor Gunnar Finnerud, Hanna Kåsastul, Paul Knutson Sæther, Abdurahman Senkaya		
Dokumenthistorikk	**REVISJON**	**UTGITT**	**BESKRIVELSE**
	1.0	20.05.2016	Første utgave

INNHOLDSFORTEGNELSE

1 DOKUMENTHISTORIE 3

2 INNLEDNING 3

3 Valg av konseptet K-Spider 4
3.1 Valg av mobilt platform 4

3 Valg av kommunikasjon 5
3.1 Innledning 5
3.2 Direkte kablet kontroll 5
3.3 Kablet datakontroll 6
3.4 Trådløs 7
3.4.1 Infrarød(IR) 7
3.4.2 Radiofrekvens(RF) 8
3.4.3 Bluetooth 8
3.4.4 Wifi 9
3.4.5 GPRS/cellular 9
3.4.6 Autonomisk system 10
3.4.7 Konklusjon 10

4 Modbus 11

5 Arduino mot Raspberry Pi 13

6 Valg av servoer 14

7 Arduino Mega vs Botboarduino 15

8 Arduino og Raspberry Pi seriell eller I2C kommunikasjon 16

9 Valg av batteri til roboten 16

10 Valg av Modbus mottaker 19

11 REFERANSER 21

12 REFERANSER TIL BILDER 21

TABELLER

Tabell 1 - Dokumenthistorie 3
Tabell 2 Forskjellige spesifikasjoner 13
Tabell 3 - Pugh matrix for batterier 17

1 DOKUMENTHISTORIE

Tabell 1 - Dokumenthistorie

VERSJON NR:	DATO ENDRET	ENDRET AV	GODKJENT AV	BESKRIVELSE
0.1	06.04.2016	Masoud	Aleksander	• Dokumentet ble opprettet • Satt sammen alle teknologidokumenter til et dokument.
1.0	20.05.2016	Masoud	Tor Gunnar	• Lagt inn kap 7 og 8 • Dokumentet er godkjent og utgitt.

2 INNLEDNING

Dette dokumentet er begrunnelse og forklaring av de forskjellige teknologiene vi har valgt å bruke i prosjektet. Det inkluderer metodene vi har brukt for å bestemme hvilke komponenter som har egnet seg for systemet vårt.

3 Valg av konseptet K-Spider

Et tilstrekkelig grundig arbeid i konseptfasen er helt avgjørende for at prosjektgruppen skal gjøre de riktige prioriteringene. Valget av konkrete løsninger eller produkter for å realisere konseptet er ikke tema i konseptfasen.

På grunnlag av de analysene og vurderingene som er gjort i starten av prosjektet velges det konseptet som fremstår som det mest fordelaktige.

Alle alternativene skal spesielt ta hensyn til:

- Funksjonell og teknisk standardisering for systemet, konseptet skal utvikles til et faglig godt grunnlag som gir tilstrekkelig sikkerhet for valg av det alternativet som best oppfyller målene innenfor prosjektets rammer.

- Forslagene til alternative løsninger fra idéfasen utredes til et tilstrekkelig detaljert nivå som gir grunnlag for å velge ett alternativ med dokumentere konsekvenser av det valget som gjøres.

- Utredningene skal være likeverdige for alle alternativene. Det skal vises til hvordan alternativene kan innpasses i investeringsrammene og hvilken effekt gjennomføringen har på prosjektgruppens økonomisk bæreevne.

- Alternativene skal analyseres og vurderes i forhold til de oppsatte målene og et definert sett av kriterier.

De viktigste premissene for utredningsarbeidet i konseptfasen er å finne en løsning som ivaretar fremtidige behov for systemet, kapasitet og driftsformer, som er robust i forhold til endringer og som kan gjennomføres. Når prosjektgruppen blir enig om valgene er lønnsomt vil prosjektgruppe starte med planleggingsfase basert på dette konseptet. Begrunnelsen for anbefalingen og de forutsetningene som er lagt til grunn dokumenteres i dette dokumentet som vi har kalt Technology dokument.

3.1 Valg av mobilt platform

Vi kunne valgt hvilken som helst plattform til å være mobilt plattformen, men valg av hexapod gjør at vi kan heise, senke, klatre trapper og gjøre mer kompliserte bevegelser enn de fleste andre plattformer. Et av kravene våre var at den skulle kunne bevege seg i ujevnt terreng.

På grunn av valget av en solid ramme av aluminium med sterke servoer kan roboten utvides til å holde flere sensorer og komponenter eller større batteri. Dette gir flere forskjellige sensormuligheter i fremtiden, som å legge på gyro for å få den til å holde seg stabil også lignende.
I tillegg har hexapoten mer showfactor og er en mer utfordrende robot å jobbe med, det er også mulighet for å jobbe med PWM og lage spesielle bevegelser, eller sekvenser til roboten.

3 Valg av kommunikasjon

3.1 Innledning

En robot krever å få data fra omgivelsene sine for å gjøre avgjørelser, eller samle data. Dette excluderer ikke roboten fra å kunne være semi autonomisk (har aspekter som er kontrollert av ett menneske og andre som den gjør selv). Ett eksempel er en drone som styres av ett menneske via kontroller, men har innebyggd gyro og motor reguleringsystem som holder den stabil, med kamera som gir feedback video til fører. Sensorer tracker tempratur, vindmotsand, avstand fra nærmeste objekt også videre. Hvis roboten mister kommunikasjon med fører må ett autonomisk program trygt lande eller returnere dronen. For å kunne bestemme hvordan vi tar imot og sender informasjon er det viktig å fastsette hvor autonomt sluttsystemet skal være og om vi ønsker kablet, tråløst eller fullt autonomisksystem.

3.2 Direkte kablet kontroll

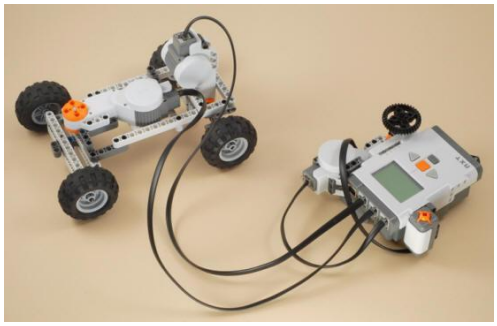
Enkleste måten å kontroller en robot på er en håndholdt fysisk kontroller som er koblet til bilen via en kabel. Dette er ved bruk av knapper, joysticks, switcher osv. For å la brukeren kontrollere roboten uten komplex type elektronikk.

Pros:

- Aldri tap av signal og lite påvirket av støy
- Minimelt med komplex elektronikk
- Roboten kan bli fysisk hentet om noe går galt (typisk undervannsroboter)

Cons:

- Kabelen kan sette seg fast, eller bli kuttet.
- Rekkeviden er minsket pga lengden på kabelen
- Økt friksjon pga kabelen



3.3 Kablet datakontroll

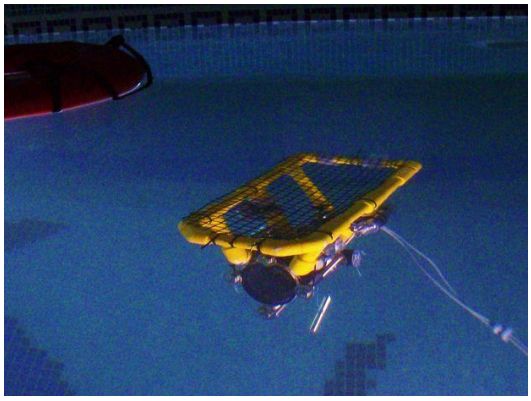
Ett nytt steg er å putte inn en mikrokontroller i systemet og forsette og bruke kabel. Mikrokontrolleren kobler man sammen med I/O portene, eller USB og gir mulighet til å kontrollere den videre gjennom et tastatur eller andre kontrollere. Ved å bruke mikrokontroller må man programmere roboten til å reagere ved input.

Pros:

- Samme pros som kablet
- Mer avangserte bevegelser kan mappes til knapper eller commandoer
- Flere kontroll muligheter, data, joystick etc.
- Sensorer og programmert «intellegens» gjør at den kan ta noen typer avgjøresler selv.

Cons:

- Koster mer en kablet pga ekstra elektronikk
- Samme cons som kablet



3.4 Trådløs

3.4.1 Infrarød(IR)

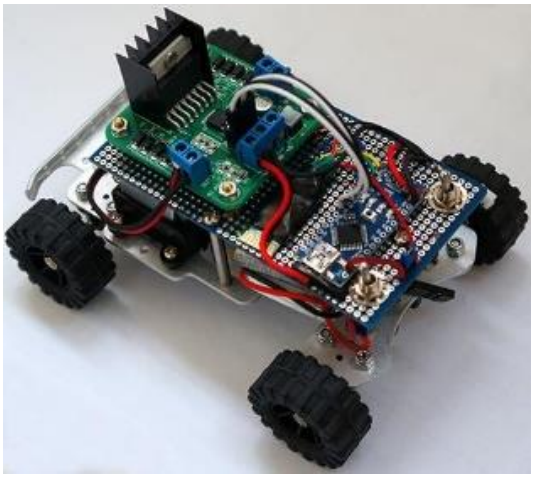
Infrarøde sendere og mottakere gjør at vi ikke trenger kabler. Ett problem med disse er at motakeren må kunne se eller ha mest mulig fri sikt heletiden for å kunne ta imot signaler. Infrarøde kontrollere (tv kontroller etc) er brukt til å sende kommandoer til en mikrokontroller som tolker og kontrollerer roboten.

Pros:

- Billig
- Enkle tv kontrollere kan brukes som kontrollere

Cons:

- Kort distanse
- Trenger fri sikt til mottaker



3.4.2 Radiofrekvens(RF)

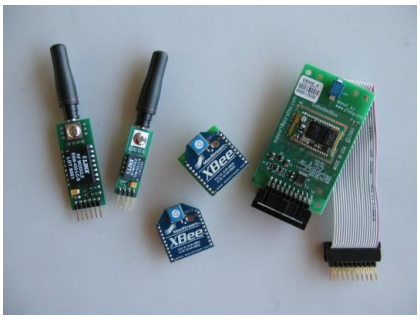
Bruker som regel mikrokontrollere til å sende, mota og tolke data sendt via RF. Mottaker bokser er som regel kretskort(PCB) some er mottaker enhet og f.eks servo motor kontroller som styres. Trenger ikke fri sikt til mottaker og kan øke kommunikasjons rekkeviden til robot med opptil flere kilometer.

Pros:

- Enkelt oppsett
- Kommuniksjon over lange distanser

Cons:

- Lav datarate, kun enkle kommandoer
- Frekvenser kan lett bli tatt opp eller sendt av andre



3.4.3 Bluetooth

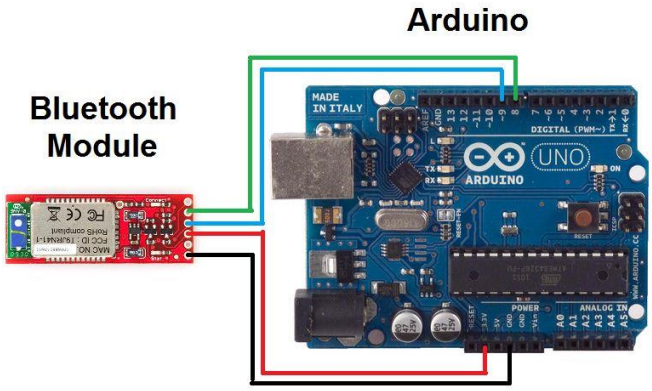
En type RF som føller spesifikke protokoller for sending og mottak av data. Normal bluetooth rekkevide er ca 10meter, men gjør at man kan kontrollere roboter via enheter som støtter bluetooth, som mobiltelefoner og laptops. Bluetooth gir mulighet til toveis kommunikasjon, som RF.

Pros:

- Høyere data overføringskapasitet
- Kontrollerbart fra alle bluetooth enheter, dater, smarttelefoner, desktops etc.

Cons:

- Ca 10m rekkevide med frisikt
- Må «paires» med enheten den kobler til



3.4.4 Wifi

Wifi er en mulighet for roboter. Å kunne kontrollere roboter via internet gir endel større fordeler (og noen ulemper) i forhold til trådløs kontrollering. For å sette opp trådløs kommunikasjon må man ha en trådløs router koblet til nettet, og en wifi modul på roboten som skal brukes.

Pros;

- Store mengder data kan overføres
- Kontrollerbart fra hvor enn i verden sålenge roboten er innen rekkevidde til trådløs router

Cons:

- Programmering trengs
- Max rekkevide kommer ann på kvalitet av router

3.4.5 GPRS/cellular

En annen trådløs teknologi er mobilnettverk, orginalt er systemet laget for kommunikasjon mellom mennesker, men det kan også brukes til å kontrollere roboter. Vanlige telefonfrekvenser er regulert, det å lage mobiltelefon moduler på en robot krever mye programmering og mye forståelse for telefonnettverket, og regulasjoner.

Pros:

- Direkte satelitt kobling mulig
- Roboten kan kontrolleres hvor enn den har mobilsignaler

Cons:

- Setup og konfigurasjon er veldig komplex
- Hvert nettverk har forskjellige restriksjoner
- Mobilnettverk er ikke gratis
- Slike systemer er enda ikke satt opp for hobbybruk generelt

3.4.6 Autonomisk system

Nestesteg er å bruke en mikrokontroller i roboten til å kunne ha en platfrom som fullt ut reagerer på inputen fra sensorer den har tilkoblet. Autonomisk kontroll kan være før programmert uten feedback, med litt feedback eller fullt ut hel autonomiskt med komplex sensor feedback. Et ekte autonomisk kontrollsystem bruker ett spekter av sensorer til å ta de beste mulige valgene til en gitt situasjon på egenhånd.
Pros:

- Kan være så enkelt som unngå objekter med distanse sensorer, til å lande ett romskip.

Cons:

- Kun så bra som moduler og programmereren kan få den til å bli.
- Komplext

3.4.7 Konklusjon

Vi ser at det mest praktiske for oss er å bruke forskjellige typer kommunikasjon, og etterhvert oppgradere om vi får kontroll over roboten.

Wifi mellom Raspberry pi og Botboarduino er mest praktisk siden vi kan sende en større mengde data fram og tilbake mellom enhetene. Det gir oss også muligheten til å sende over lengre distanse som vi kommer til å trenge i fremtiden når roboten blir videre utviklet til ett mer autonomisk systm. RCU'en som gir kommandoer til raspberry pi oppererer også ikke altfor fort, så det å sende pakker med informasjon til systemet som systemet selv prossesserer vil være det beste.

Eventuelt vil det bli ett fokus på å autonomisere systemet, da er denne koblingen mest praktisk, og kan overføre større menger sensor data som fører kan bruke. Siden en del av prosjektet er å utvikle en sensor platform må vi ha kapasitet til å eventuelt overføre data.
Før vi implementerer wifi er vi siktet inn på å bruke kablet datakontroll mellom raspberry pi og botboarduino. Dette er den tryggeste første løsningen som vil gi oss ett fungerende system med relativ komplexitet.

4 Modbus

Modbus er en seriell kommunikasjons protokoll som er veldig mye brukt i industrien i samarbeid med PLC 'er. Den er blitt så mye i bruk siden er simple og robust å bruke. Denne protokollen gjør det mulig å at flere enheten sender data til en hovedenhet som så behandler denne informasjonen. Det er forskjellige variasjoner av denne protokollen for eksempel: Rs232, Rs485 og Tcp/Ip.

Rs-232 Er den protokollen som kanskje er mest brukt i industrien. Det er også den eldste protokollen og er i mange tilfeller begynt å bli byttet ut med nyere og mer effektive alternativer.

Styrkene til 232 er:

- Enkel implementering.
- Veldig utbredt.
- Lave kostnader.
- Fleste microkontrollere har muligheter for denne protokollen.

Svakhetene er:

- Forskjellige konfigurasjoner mellom flere forskjellige enheter.
- Veldig begrenset kommunikasjons distanse.
- Veldig mottagelig for støy.
- Veldig mange forskjellige software protokoller.

Rs-485: Er betraktelig mindre utbredt enn det 232 er, den er betydelig mye raskere da den kan håndtere større datamengder. Konfigurasjonen dens tillater bruk av flere drivere og flere mottakere.
Styrkene til 485:

- Lave kostnader.
- Immunitet mot støy.
- Konfigurasjon som tillater bruk av flere drivere og mottakere.

Svakhetene er:

- Veldig lite utbredt. Vanskeligere å finne informasjon og eksempler.
- Færre enheter med standardiserte implementasjons muligheter.

Tcp/Ip: Er egentlig det samme som Modbus RTU, forskjellen er at den egentlig tar RTU pakkene og pakker inn disse i TCP/IP pakker.

Strykene er i hovedsak:

- Enkel å implementere
- Lav kostnad

Svakhetene er:

- Noe tregere enn andre Ethernet baserte industri protokoller
- Siden det er enda et lag som dataene blir lagt i, så blir feilsøkingen noe vanskeligere enn ved vanlig modbus RTU.

Konklusjon

Med hensyn til denne informasjonen så falt vi på valget RS232, spesielt med tanke på hvor utbredt denne protokollen er og tilgjengelig informasjon. Tcp/Ip var også en stor mulighet, men siden den og rs232 er veldig like bortsett fra at feilsøking på denne kan være vanskeligere på dette, så droppet vi dette av praktiske hensyn.

5 Arduino mot Raspberry Pi

Her er det tenkt at vi tar en kikk på forholdene mellom Arduino og Raspberry Pi, og hvilken vi burde velge. Begge er tenkt å operere i samme funksjoner og miljø, og det eneste vi skal se på er hvordan de kan takle dette og hva prisene er i forhold til hverandre

Tabell 2 Forskjellige spesifikasjoner

Enhet	Arduino Uno	Raspberry pi 2
Operasjons spenning	5v	3.3v
Input spenning	7-12v	5v
Digitale I/O pinner	14	40
PWM I/O pinner	6	-
Analoge I/O pinner	6	-
Flash minne	32KB	SD Kort (15Gb)
Klokke hastighet	16Mhz	900Mhz
Operativ system	Ingen	Ja
IDE	Arduino	Flere forksjellige
Flere operasjoner	Nei	Ja
On Board netverk	Nei	Ja, via Ethernet
Ram minne	2KB	1Gb
Annet	-	4 USB porter og HDMI inngang
Pris	21$(Dollar)	35$(Dollar)

Konklusjon:

Siden pris kostnadene ved de 2 mulighetene er så like, så valgte vi å gå for Raspberry pi(RPi) da den har betraktelig bedre hardware kapasitet og tilkoblingsmuligheter. Den har også en mye mer allsidige bruksområdet da den bruker et operativsystem som gir muligheten til å gjøre mye forskjellige slags oppgaver, samt at den støtter flere programmeringsspråk. Den mest åpenbare svakheten til raspberry pi(RPi) og styrken til Arduino, er analoge inputs. Raspberry har ingen standard mulighet for dette (kan implementeres ved hjelp av ekstra moduler), mens hos Arduino så er dette med som standard. Med dette i bakhodet så gir RPi oss gode muligheter for senere utvidelser, enten ved denne gruppen eller senere grupper.

6 Valg av servoer

For å foreta et valg mellom servorer, som vi skulle ha 18 av, 3 til hvert bein på hexapoden. Dette valget måtte vi ta på bakgrunn av flere faktorer. Disse faktorene var miljøet den kanskje skulle operere i, som terreng. Og hvor mye utstyr, for eksempel vi skulle ha systemet til å bære, dermed løfte styrken til servoene.

Som en metode for å få et bedre overblikk over styrkene og svakhetene til de aktuelle servotypene, så valgte å benytte oss av pugh-matrix metoden. Pugh-matrix fungerer ved at vi setter servoene opp mot hverandre i en matrise, hvor vi lister opp kravene vi setter til komponentene og gir en poeng sum til hver komponent ut i fra hvor tilfredsstillende de er på de individuelle kravene.

Choise of servo			Score			Weighted score			Numbers			
	Criteria	**Weight**	**A**	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**	**Units**
	Specs											
	Speed	**2**	5	3	1	10	6	2	0,15	0,2	0,23	*sec/60*
	Torque	**4**	1	4	5	4	16	20	3,7	9,6	13,2	*kg/cm*
	Weight	**3**	5	4	1	15	12	3	43	55,2	110	*g*
	Size	**4**	4	4	2	16	16	8	40*20*36.5	41*20*38	59*29*50	*mm*
	Other											
	Cost	**2**	3	3	4	6	6	8	9,89	26,99	26,29	*USD*
Short	**Full name**				∑	51	56	41				
A	*HS-322HD*											
B	*HS-645MG*											
C	*HS-755HB*											

Konklusjon:

Etter å ha benyttet pugh-matrix så kom vi frem til at HS-645MG var den servoen som passer vårt prosjekt best. Selv om HS-322HD hadde veldig lik score, så var Torque såpass lav at vi kom frem til at den ikke er så aktuell likevel på grunn av dette punktet alene, selv om prisen var en god del mindre. Så denne metoden ga en god oversikt over de forskjellige komponentene og forskjellene mellom de.

7 Arduino Mega vs Botboarduino

I dette kapittelet skal vi se på forskjellene mellom Arduino mega og botboarduino. Dette skal gjøres fordi det ble spørsmål hvorvidt botboarduino var godt nok for arbeids oppgavene vi hadde tiltenkt det, og samtidig se om Arduino mega er et godt nok til at det faktisk er ønskelig å bytte de ut med hverandre.

Grunnen til at en utskiftning ble foreslått er at prosjektet var kommet så langt at ekstra funksjoner til systemet var ønskelig og botboarduino sin minne kapasitet og prosessor var i stor grad oppbrukt. Da vi hadde planlagt å implementere ytterligere funksjoner til systemet, så kunne det komme til å bli et problem med kapasiteten.

Når en skal ta å sammenligne disse enhetene, så er det noen egenskaper som i hovedsak vil bli fokusert på, disse egenskapene er blant annet lagrings kapasitet, minne og prosessorkraft, samt GPIO pinner og den fysiske størrelsen på enheten.

Arduino-mega vs Botboarduino

Egenskap	**ATmega2560**	**Botboarduino**
Op voltage	5v	5v
Input voltage	7-12v	7-12v
Dig I/O pinner	54(15 PWM)	14(6 PWM)
Analog input pin	16	6
Flash minne	256KB	32 KB
SRAM	8KB	2 KB
EEPROM	4KB	1 KB
Klokkehastighet	16 MHz	16 MHz
Lengde	101.5 mm	70 mm
Bredde	53.3 mm	53 mm

Konklusjon:

Som vi kan så er begge disse kortene like på mange måter, men ATmega2560 har betraktelig større lagrings plass. Siden dette var en av egenskapene vi ville se etter, så er dette en ønskelig egenskap. Som nevnt tidligere så trengte systemet økt lagring, for å kunne håndtere utvidelser. Selv om ATmega2560 sine dimensjoner er noe større en botboarduino, så veier den økte kapasiteten opp dette.

8 Arduino og Raspberry Pi seriell eller I2C kommunikasjon

Spesifikasjoner om seriell (USB) og I2C kommunikasjon.

Seriell (USB):

- Asynkront,
- God for kommunikasjon mellom 2 enheter (opp til 127)
- Lite rot, siden det er snakk om USB
- Enkel å bruke
- Max distanse 5m (kommer ikke til å bli nødvendig i systemet)
- Hastigheter fra 1.5 Mbps til 4.8Mbps
- 5v/1A

I2C:

- Synkront
- 7 bits og 10 bits adresser
- 10kbps til 5Mbps hastigheter
- Multi-master bus
- Tx – Rx
- Kan være litt vrient å implementere

Konklusjon:

Som vi kan se så er begge protokollene like på mange måter der det er relevant for prosjektet. Overførings hastighetene er ganske like, USB er litt tregere. De like go de på avstand, men det som kanskje er det viktigste er at USB er noe enklere å få til. Og at USB har den evnen at det kan overføre 5V/1A, i tillegg kan også Arduino programmeres fra Raspberry pi skrivebord.

9 Valg av batteri til roboten

Batterier er essensielle komponenter av en robot, og uten disse, kan en robot ikke være funksjonell. Før vi velger en passende batteri, har vi analysert komponentene og deres energiforbruk. Noe som gir et bredt utvalg av batterier som er tilgjengelig på markedet, å velge et batteri kan være en kompleks prosess.

I prosessen med valg av batteri, ble det tatt hensyn til forskjellige punkter, inkludert vekten av roboten, servomotorenes strømforbruk osv. Disse faktorene er avgjørende for tiden roboten kan operere med et fulladet batteri.

Her er noen fordeler og ulemper med hver.

Tabell 3 - Pugh matrix for batterier

Batteritype				
Kriterier	LiPo	Alkaline	NI-CD	Ni-MH
Sikkerhet	-	-	0	-
Kostnad	-	-	-	+
Memory effekt	+	0	-	0
Livssyklus	+	-	+	+
Effekt	+	+	-	+
Størrelse	+	+	+	+
Miljøvennlighet	0	-	-	-
Utladning	+	-	+	+
Sum+	5	2	3	4
Sum-	-2	-5	-4	-2
Sum0	0	0	0	0
Nett	3	-3	-1	2
Valg	JA	NEI	NEI	JA

LiPo:

- Fordeler:
    - Kan lades.
    - Inneholder ikke miljøgifter.
    - Kan kastes i restavfall.
    - Har lav vekt i forhold til energien det kan lagre.
    - Har lavt volum i forhold til energien det kan lagre.
    - Kan lages i firkantet form (ypperlig for RC).
    - Lettvekt

- Ulemper:
    - Må lades opp før det kan tas i bruk.
    - Bør "hvile" 15 min etter lading før det tas i bruk.
    - Bør "hvile" 15 min etter at det har blitt tappet ut før det settes på lading.
    - Kan eksplodere og ta fyr ved feil bruk.
    - Trenger regulator

NI-MH:

- Fordeler:
  - Ingen minneeffekt
  - Lett vekt
  - Inneholder ikke miljøgiften kadium
  - Råvaren Nikkel kan gjenvinnes og brukes i nye batterier
  - Kan lades
  - Lettvekt
- Ulemper:
  - Inneholder miljøgiften Nikkel.
  - Skal behandles som spesialavfall når de kastes.
  - Må lades opp før det kan tas i bruk.
  - Har høy selvutlading.

NI-CD:

- Fordeler:
  - Kan lades.
  - Kan ta imot lading raskt og kan gi fra seg strøm raskt.
  - Råvarene nikkel og kadmium kan gjenvinnes og brukes i nye batterier.
- Ulemper:
  - Inneholder miljøgiftene nikkel og kadmium.
  - Skal behandles som spesialavfall når de kastes.
  - Må lades opp før det kan tas i bruk.
  - Påstås å ha en minneeffekt, denne kan "nøytraliseres" ved å tømme batteriet helt av og til.

Konklusjon:

Basert på konseptvalg matrisen, har vi begrenset våre valg til enten LiPo, eller et NI-MH. Vi har valgt NI-MH batteriene som ekstra løsning, Siden dette er vanligste typen av batterier som brukes i mobile robot plattformer, mens LiPo batterier vil gi lengre levetid til systemet og dermed valgt som hoved batterikilde til systemet.

10 Valg av Modbus mottaker

Valg av mottaker til Modbus er kanskje den vanskeligste og viktigste biten i prosjektet, her er det avgjørende med tid, lagring av informasjon, enkelhet og kunnskap i feltet. Informasjonen som kommer fra RCU 510 må tolkes og det fins et hav av muligheter. Etter gjennom gang av oppgaven har vi sortert og vurdert noen av de forskjellige løsninger.

Her er ulempene og fordelene med løsningene våre:

Lage Kretskort:

- Fordeler:
  - Mindre kabler
  - Ingen programmering
  - Lettvekt
  - Størrelse

- Ulemper:
  - Ingen wifi løsning
  - Trenger mikroprosessor for videre føring av signalet
  - Tar lang tid å bestille

Arduino Modbus løsning:

- Fordeler:
  - C/C++ programmering
  - Kunnskap fra området
  - Har tilgang til flere Arduino
  - Wifi løsning

- Ulemper:
  - Mindre kapasitet
  - Må ha ekstra enhet for wifi løsning eller xbee wifi løsning
  - Kan ikke ha kamera live feed
  - Mindre prosessorkraft
  - Mindre lagring plass

Raspberry pi:

- Fordeler:
  - Stort minne
  - wifi løsning
  - Kraftig prosessor
  - Kamera live feed

- Ulemper:

- Takler ikke mer 3.3V GPIO pinnene
- Ingen kunnskap fra området
- Python programmering (ingen er kjent med dette språket)

Konklusjon:

På grunn av kompleksiteten til systemet og fordi vi satt fast enstund på dette vanskelige området så kjørte vi arbeid parallelt med arduino og raspberri pi siden disse begge var godt brukbare løsninger for oss. Vi droppet kretskort, til tross for at flere i gruppen er gode med PCB editor og kretskortdesign. Evt forsinkelser i levering eller feil ved bestilling være for stor risikofaktor. Samtidig som at funksjonene vi får ut av denne løsningen ikke passer oss.

11 REFERANSER

[1] Electronic Design, 18.03.2016, http://electronicdesign.com/what-s-difference-between/what-s-difference-between-rs-232-and-rs-485-serial-interfaces

[2] Modbus, 18.03.2016, https://en.wikipedia.org/wiki/Modbus

[3] Oscarliang, 08.04.2016, https://oscarliang.com/connect-raspberry-pi-and-arduino-usb-cable/

[4] Oscarliang, 08.04.2016, https://oscarliang.com/raspberry-pi-arduino-connected-i2c/

[5] Arduino, 16.05.2016, https://www.arduino.cc/en/Main/ArduinoBoardDuemilanove

[6] Arduino, 16.05.2016, https://www.arduino.cc/en/Main/arduinoBoardMega2560

[7] Sparkfun, 18.05.2016, https://learn.sparkfun.com/tutorials/i2c

[8] Academia, 19.05.2016,
http://www.academia.edu/4771771/Understanding_I2C_Firewire_and_USB_protocol

12 REFERANSER TIL BILDER

[1] ROV, 25.02.2016
http://cdn.instructables.com/FRH/F91Q/FQ6DWWV4/FRHF91QFQ6DWWV4.MEDIUM.jpg

[2] Direkte kontroll, 25.02.2016, http://troop503.info/robotics/wp-content/uploads/2011/07/Remote-Control-NXT-Rover.jpg

[3] Remote controll 25.02.2016,
https://cdn.shopify.com/s/files/1/0174/1800/products/Remote_Control_2_of_3_1024x1024.jpg?v=374658030

[4] RF, 25.02.2016, http://slab.concordia.ca/wp-content/uploads/2008/11/img_0454- 537x402.jpg RF 25.02 14.55

[5] Bluetooth, 25.02.2016, http://whiz-3d.com/images/arduinoBT.jpg

[6] Wifi, 25.02.2016, http://stutt.no/51d065

[7] GPRS 25.02.2016, http://www.techkrazy.com/wp-content/uploads/2015/02/Mobile-Phone-Mobile-Phone-Market-Driving-Force.png

[8] Wifi Router, 25.02.2016, http://www.bandwidthplace.com/wp/wp-content/uploads/2014/11/wifi_router.png

[9] Batteri, 23.02.2016, http://www.robotshop.com/blog/en/how-do-i-choose-a-battery-8-3585

[10] Batteri, 23.02.2016, http://www.rc-leker.no/index.php?option=com_content&view=article&id=14&Itemid=28

Høgskolen i Sørøst-Norge

KONGSBERG

TESTPLAN

PROSJEKT	K-Spider		
OPDRAGSGIVER	Kongsberg Maritime AS		
UTFØRT VED	Høgskolen i Sørøst-Norge, avd. Kongsberg		
REVISJON	3.0		
MEDLEMMER	Aleksander Tokle Poverud, Masoud Shah Pasand, Tor Gunnar Finnerud, Hanna Kåsastul, Paul Knutson Sæther, Abdurahman Senkaya		
Dokumenthistorikk	**REVISJON**	**UTGITT**	**BESKRIVELSE**
	1.0	01.01.2016	Første utgave
	2.0	08.03.2016	Andre utgave
	3.0	20.05.2016	Tredje utgave

INNHOLDFORTEGNELSE

1 DOKUMENTHISTORIE 3

2 INNLEDNING 3

3 FREMGANGSMÅTE 4

3.1 TESTFASER 5

3.1.1 Pre-alfa Test 5

3.1.2 Alfa Test 5

3.1.3 Beta Test 6

3.1.4 Internal Acceptane Test 6

3.1.5 Factory Acceptance Test 6

3.1.6 Utgivelse av produktet 6

4 TESTSPESIFIKASJON 7

4.1 Testing ved hjelp av inspeksjon 7

4.2 Testing ved hjelp av funksjonstest 8

5 TEST AV RAMMEKRAV 10

6 TEST AV FUNKSJONELLE KRAV 11

7 TEST AV MASKIN-/PROGRAMVARE KRAV 12

8 GODKJENNINGSKRITERIER 13

8.1 Kriterier til start, avbrudd og gjenopptakelse av test 13

9 RESULTATDOKUMENTER 14

10 VEDLEGG 14

TABELLER

Tabell 1 - Dokumenthistorie 3

Tabell 2 - Klassifikasjon av feil 4

Tabell 3 - Test av rammekrav ved hjelp av inspeksjon 7

Tabell 4 - Test av maskinkrav ved hjelp av inspeksjon 7

Tabell 5 - Test av rammekrav ved hjelp av funksjonstest 8

Tabell 6 - Test av funksjonelle krav ved hjelp av funksjonstest 8

Tabell 7 - Test av maskin- og programvarekrav ved hjelp av funksjonstest 8

Tabell 8 - Testplan 9

Tabell 9 - Test av rammekrav 10

Tabell 10 - Test av funksjonelle krav 11

Tabell 11 - Test av maskin- og Programvarekrav 12

Tabell 12 - Vedlegg 14

FIGURER

Figur 1 - Testfaser 5

1 DOKUMENTHISTORIE

Tabell 1 - Dokumenthistorie

VERSJON NR:	DATO ENDRET	ENDRET AV	GODKJENT AV	BESKRIVELSE
0.1	15.01.2016	Hanna	Masoud	• Dokumentet ble opprettet
1.0	02.02.2016	Hanna	Masoud	• Godkjent og utgitt
1.1	02.03.2016	Masoud	Aleksander	• Endret navn på dokumentet fra Testspesifikasjon til Testplan. • Oppdatert innholdet
1.2	04.03.2016	Masoud	Tor Gunnar	• Lagt inn fremgangsmåte av testingen og testfaser. • Lagt inn vedlegg 1 (Testrapport)
2.0	08.03.2016	Masoud	Hanna	• Godkjent og utgitt
2.1	10.05.2016	Masoud	Aleksander	• Endret på versjon nr. og datoer som ikke var innført.
2.2	19.05.2016	Masoud	Aleksader	• Lagt inn vedlegg 2 • Rettet på godkjenningskriterium Krav ID KF04
3.0	20.05.2016	Masoud	Aleksander	• Lagt inn resultater av testene • Dokumentet er godkjent og utgitt

2 INNLEDNING

Dette dokumentet inneholder informasjon om hvordan testplan/testspesifikasjon skal bidra til at kravene i kravspesifikasjonen skal være oppfylt. Testspesifikasjonen beskriver testmetodene som skal utføres for å teste kravene i kravspesifikasjon. Hensikten med testspesifikasjon er å verifisere krav og at kravene blir oppfylt.

3 FREMGANGSMÅTE

Test-arbeidet vil i prinsippet være manuelle tester via inspeksjon for å se at kravene er oppfylt og at dette fungerer på en god måte. Resultatene vil sjekkes opp mot testspesifikasjonen som er skrevet i kapittel 4. Alle testresultater vil noteres, og til slutt bli samlet i en testrapport.

Tabell 2 viser kategori rangering som vi skal bruke ved notering av eventuelle feil i vedlegg 1 - testrapport. Prioriteringen er klassifisert fra A-D i synkende rekkefølge:

Tabell 2 - Klassifikasjon av feil

Kategori A:	Feil i en eller flere funksjoner som forårsaker stopp, for eksempel kommunikasjon, system, maskin eller funksjon.
Kategori B:	Feil har alvorlige konsekvenser i systemet eller komponenter, for eksempel feil i databaser, programmering og kalibrering.
Kategori C:	Feil med mindre alvorlige konsekvenser, for eksempel mindre grad av ustabilitet, små funksjonelle mangler.
Kategori D:	Feil uten alvorlige konsekvenser, for eksempel unøyaktig kalibrering, dårlig estetikk, rotete layout.

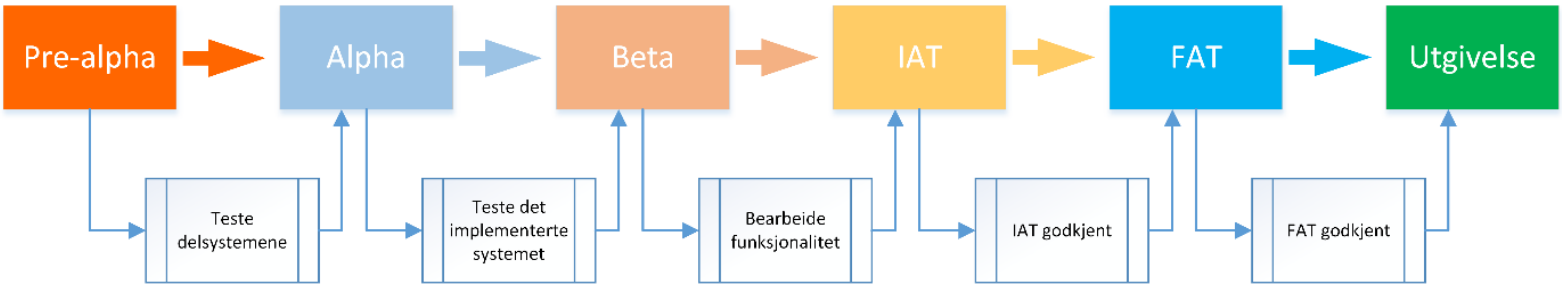
Vi har lagt vekt på det mest grunnleggende av funksjonaliteten til de forskjellige delsystemene nå i starten. Dette er for å kunne teste at systemet fungerer som enkle delsystemer før vi skal integrere det. Dette kalles for en «Alfa» test. Hver iterasjon vil brukes til å utvikle og bygge produktet opp ved å øke funksjonaliteten i inkrementelle deler. Når alle delsystemene er ferdigstilt begynner vi på integrere systemet, det er her vi må jobbe mest med programmering for å få signalene til å gå gjennom systemet til roboten som skal styres. For å kvalitetssikre arbeidet vårt har vi brukt flere verktøy. Vi har delt opp testingen i flere faser, slik det er beskrevet under i dette kapittelet.

En «Beta» test blir utført for å avdekke feil og verifisere at programvaren er stabil og at produktet har riktig funksjonalitet.

«Internal Acceptance Test» (IAT) blir utført internt i prosjektgruppen for å verifisere og avdekke feil i kommunikasjonsnettverket. Dette gjøres for å kvalitetssikre produktet på en god måte før oppdragsgiver blir invitert til en test av produktet.

Når alle delsystemene er på plass og alt blir satt sammen til et komplett system skal det kjøres en "Factory Acceptance Test" (FAT) på hele systemet for å verifisere at alt fungerer sammen slik det skal. Denne testprosedyren er beskrevet i kapittel 4 og utover til kapittel 7.

3.1 TESTFASER



Figur 1 - Testfaser

3.1.1 Pre-alfa Test

Pre-alfa refererer til alle aktiviteter som utføres før testingen finner sted. Disse aktivitetene kan omfatte analyser av krav, design, implementasjon, testing osv. I tillegg har vi opprettet en milepælsoversikt, slik at prosjektgruppen til enhver tid vet når en spesifikk funksjon skal være ferdigstilt. I denne fasen vil vi teste hver enkel komponent for å verifisere at de fungerer. Her vil vi få muligheten til å rette på enkle feil og bytte ut komponenter om det skulle være nødvendig. I tillegg får man et godt overblikk om hvordan hver komponent fungerer i systemet.

3.1.2 Alfa Test

Alfa-testen går videre på å ta alle de komponentene vi testet i pre-alfa og implementere disse sammen til systemet. Planen for å gjennomføre testen vil være å teste RCU til RPi systemet og at Hexapoden fungere som det skal. Det vil si at det er kommunikasjon mellom alle delsystemene og at vi får en indikasjon på hvordan alle delsystemene vil fungere sammen som et system. Etter denne testen vil vi kunne vite hva vi må endre på av «hardware» og «software» før vi går videre. Prosjektgruppen er enige om at denne testen ikke er godkjent før alle delsystemene fungerer som de skal.

3.1.3 Beta Test

Betatesting utføres etter alfa-testingen, og før produktet slippes ut til oppdragsgiveren for offentlig testing. Dette er vanligvis ment som en offentlig utgivelse, men ofte begrenset for å få tilbakemeldinger på produktet. Beta-testingen vil avdekke og redusere feil, og danne et høyere potensial for fremstillingen av bedre kvalitet på sluttproduktet.[1]

I denne fasen blir funksjonaliteten av systemet bearbeidet, basert på resultatene fra alfa-testingen. Viktig å merke seg at i denne fasen vil produktet vårt allerede ha en stabil arkitektur og kan utgis som et ferdig produkt. Men mest sannsynlig vil det ha funksjonaliteter som ikke har blitt bearbeidet nok og om det endelige produktet vil ha de samme funksjonalitetene som betaversjonen avhenger av testresultater og tilbakemeldinger fra oppdragsgiveren. Vi vil først og fremst teste om alle funksjonalitetene i delsystemene fungerer og at det er stabil kommunikasjon mellom dem.

3.1.4 Internal Acceptane Test

IAT (Internal Acceptance Test) kommer etter betatestingen er en test som blir gjennomført for å avgjøre om kravene eller spesifikasjonen I kontrakten er oppfylt. Dette er en komplett test av '' hardware'' på det endelige produktet. Her følges prosedyren i kapittelet 5, 6 og 7 for å verifisere at systemet oppfyller kravspesifikasjonen. Hvis alt er ok, vil vi vurdere ut ifra punktene som er satt opp i godkjenningskriteriene i kapittel 8. Om testen er bestått vet vi at produktet fungerer og vi kan gjøre oss klar til neste som er FAT.

3.1.5 Factory Acceptance Test

FAT (Factory Acceptance Test) er den siste store testen for å verifisere at produktet har møtt kravene spesifisert i kravspesifikasjon, og for å verifisere at vi nå er klare for å overlevere systemet til oppdragsgiveren. I FAT følger vi samme prosedyren i kapittelet 5, 6 og 7 som vi gjorde i likhet med IAT. Vi bruker FAT-sjekklisten for å verifisere at kravene til alle rammene er nådd. Når FAT er komplett og godkjent er vi klare for å overleve produktet til oppdragsgiveren.

3.1.6 Utgivelse av produktet

Etter at alle testene er utført og alle godkjenningskriteriene i kapittel 8 er nådd og godkjent er vi klare til å offisielt gi ut produktet og overrekke dette til oppdragsgiveren.

[1] http://www.guru99.com/alpha-beta-testing-demystified.html

4 TESTSPESIFIKASJON

Testspesifikasjonen skal være et kontrollorgan som sikrer at kravene skal være oppfylt. Den er bygd slik at hver test refererer til samsvarende krav og forteller hva som testes, hvordan testen utføres, hvilket resultat den får og på hvilket tidspunkt testen utføres. Det er også et dynamisk dokument før endelige godkjenninger er på plass og dokumentet anses ferdig i henhold til produktet. Testene settes opp etter modellen beskrevet i tabell 8.

Det skal være utført flere tester under prosjekt prosessen. Noen av dem skal utføres før hele prosjektet er ferdigstilt. For å gi bedre oversikt over testene er alle kravene blitt fordelt mellom visuell (inspeksjons-) testing og funksjonstesting med prosjekt lokasjon, dvs. hvilke konstruksjons deler er involverte i hvert av kravene. I tillegg er alle kravene blitt satt i rekkefølge etter prioritet slik at de skal testes rett etter de skal være tilgjengelige for testing.

4.1 Testing ved hjelp av inspeksjon

Tabell 3 - Test av rammekrav ved hjelp av inspeksjon

KRAV ID	PRIORITET	SYSTEMDEL/PROSJEKT LOKASJON
KR01	A	Plattformen (robot)
KR02	A	Plattformen (robot)
KR03	A	Plattformen (robot)

Tabell 4 - Test av maskinkrav ved hjelp av inspeksjon

KRAV ID	PRIORITET	SYSTEMDEL/PROSJEKT LOKASJON
KM01	A	Plattformen, AIM7, RCU510
KM02	A	Plattformen, AIM7, RCU510
KM03	A	Plattformen, AIM7, RCU510
KM04	C	Plattformen, AIM7, RCU510, feedback sensorer
KM05	C	Plattformen, AIM7, RCU510, misjon sensorer

4.2 Testing ved hjelp av funksjonstest

Tabell 5 - Test av rammekrav ved hjelp av funksjonstest

KRAV ID	PRIORITET	SYSTEMDEL/PROSJEKT LOKASJON
KR02	A	Plattformen (robot)
KR03	A	Plattformen (robot)
KR04	A	Plattformen (robot), AIM7, RCU510
KR05	C	Plattformen (robot), AIM7, RCU510, misjon sensorer

Tabell 6 - Test av funksjonelle krav ved hjelp av funksjonstest

KRAV ID	PRIORITET	SYSTEMDEL/PROSJEKT LOKASJON
KF01	A	Plattformen (robot)
KF02	A	Plattformen (robot), AIM7, RCU510
KF03	B	Plattformen (robot), fjernkontroll
KF04	C	Plattformen (robot), AIM7, RCU510, feedback sensorer
KF05	C	Plattformen (robot), AIM7, RCU510, nettverket

Tabell 7 - Test av maskin- og programvarekrav ved hjelp av funksjonstest

KRAV ID	PRIORITET	SYSTEMDEL/PROSJEKT LOKASJON
KM04	C	Plattformen (robot), AIM7, RCU510, feedback sensorer
KM05	C	Plattformen (robot), AIM7, RCU510, misjon sensorer

Under ligger den konkrete fremgangsmetoden for å teste at kravene er oppfylt.

Tabell 8 - Testplan

KATEGORIER	BESKRIVELSE
TEST ID	Testens identitet mot treffende krav.
KRAV ID	Kravets identitet linkes mot test identitet.
TEST TYPE	Hvilken test type benyttes.
METODE	På hvilken måte test utføres.
DOGKJENNINGSKRITERIUM	Det som gir testen godkjent.
RESULTAT	Hvilket resultat testen har gitt.

Når testene er utført trekkes det en konklusjon av resultat og det avgjøres da om det er nødvendig med tiltak og eventuell endring av kravet.

5 TEST AV RAMMEKRAV

Tabell 9 - Test av rammekrav

TEST ID	KRAV ID	TEST TYPE	METODE	GODKJENNINGSKRITERIUM	RESULTAT
TR01	**KR01**	Inspeksjonstest	Å koble komponentene til roboten sammen og observere visuelt et fysisk legeme.	Testen er godkjent ved inspeksjon at det er observert et fysisk legeme, og at bevegelige elementene er inkludert.	Godkjent
TR02	**KR02**	Funksjonstest Inspeksjonstest	Å tilkoble roboten til et styresystem og observere robotens bevegelser.	Testen er godkjent når robotens lemmer beveger seg som forventet.	Godkjent
TR03	**KR03**	Funksjonstest Inspeksjonstest	Å teste funksjonene og observere utseende.	Testen er godkjent når funksjonene er verifiserte og utseende ser presentabelt ut.	Godkjent
TR04	**KR04**	Funksjonstest	Å påse at det er RCU510 som utfører hoved sekvensene på roboten.	Testen er godkjent når kontrollen av roboten skjer via RCU510 fra AIM7.	Godkjent
TR05	**KR05**	Funksjonstest	Å verifisere funksjonene hver for seg og samhandling mellom hverandre.	Testen er godkjent når roboten reagerer på inputen fra sensorer som forventet.	Godkjent

6 TEST AV FUNKSJONELLE KRAV

Tabell 10 - Test av funksjonelle krav

TEST ID	KRAV ID	TEST TYPE	METODE	GODKJENNINGSKRITERIUM	RESULTAT
TF01	**KF01**	Funksjonstest	Å utføre en praktisk test, og observere resultatet	Testen er godkjent når roboten utfører sammensatte bevegelser.	Godkjent
TF02	**KF02**	Funksjonstest	Å koble systemet sammen og overføre data fra kontrollsystemet AIM7 via RCU510 til roboten.	Testen er godkjent når roboten responderer på kommandoer fra kontrollsystemet AIM7 via RCU510.	Godkjent
TF03	**KF03**	Funksjonstest	Å koble roboten og en ekstern fjernkontroll sammen og sende kommandoer fra fjernkontrollen til roboten.	Testen er godkjent når roboten responderer på kommandoer fra en ekstern fjernkontroll.	Godkjent
TF04	**KF04**	Funksjonstest	Å koble AIM7, RCU510 og roboten sammen og overføre måle-data fra robotens sensorer til operatørsystemet.	Testen er godkjent når det skal være mottatt informasjon fra roboten til Operatør via kommunikasjonsnettverket.	Ikke implementert
TF05	**KF05**	Funksjonstest	Å sende kommandoer fra AIM7 trådløst via WiFi til roboten.	Testen er godkjent når roboten responderer på kommandoer AIM7 via WiFi.	Godkjent

7 TEST AV MASKIN-/PROGRAMVARE KRAV

Tabell 11 - Test av maskin- og Programvarekrav

TEST ID	KRAV ID	TEST TYPE	METODE	GODKJENNINGSKRITERIUM	RESULTAT
TM01	**KM01**	Inspeksjonstest	Å observere en operatørstasjon	Testen er godkjent når det skal være observert at operatørstasjonen er tilkoblet til systemet og begge to er operative.	Godkjent
TM02	**KM02**	Inspeksjonstest	Å verifisere kontroll systemet.	Testen er godkjent når RCU510 er tilkoblet og utfører styring av roboten.	Godkjent
TM03	**KM03**	Inspeksjonstest	Å verifisere kontroll systemet.	Testen er godkjent når AIM7 styrer roboten.	Godkjent
TM04	**KM04**	Funksjonstest Inspeksjonstest	Å skape en situasjon hvor installerte sensorene ville reagere.	Testen er godkjent når installerte sensorene er observert og måledata skal være avlest av roboten og sendt til operatørstasjon.	Godkjent
TM05	**KM05**	Funksjonstest Inspeksjonstest	Å skape en situasjon hvor installerte sensorene ville reagere.	Testen er godkjent når installerte måledata er observert og sensorene skal være sendt til operatørstasjon fra roboten.	Ikke implementert

8 GODKJENNINGSKRITERIER

For at en test skal regnes som vellykket eller feilet må det verifiseres av den ansvarlige.
Det stilles en rekke kriterier til systemet:

- Alle delsystemer skal fungere i henhold til kravene som er satt.
  - Et delsystem regnes som funksjonelt når det ikke finnes feil av Kategori A og B samt at det har oppfylt kravet spesifisert i kapitlene 5, 6 og 7.
- Antall feil som avdekkes skal være synkende.
- Alle avdekkede feil av Kategori A skal være rettet i hele systemet.
- Alle avdekkede feil av kategori B skal være rettet i hele systemet.

Det er også en rekke kriterier for at systemet skal kunne være komplett:

- Alle nådde krav er testet og verifisert.
- Systemet skal ha passert FAT beskrevet i delkapittelet 3.1.5

8.1 Kriterier til start, avbrudd og gjenopptakelse av test

Det stilles en rekke kriterier til start, avbrudd og gjenopptakelse av test. Disse er:

- Tor Gunnar som har hovedansvar for testen må konsultere før en vurdering tas for start og gjenopptakelse av test.
- Kategori A og B vil være grunnlag for å avbryte testing og starte feilretting.
- Feil av Kategori C må vurderes for å avbryte testing og starte feilretting.
- Feil av Kategori D danner grunnlag for en kommentar.

9 RESULTATDOKUMENTER

Siden det er behov for omkjøring av testene er det viktig at disse dokumenteres, og alt av resultater skal kunne refereres til:

- Testplan
- Testspesifikasjon i kapittel 4 (i dette dokumentet)
- Rammekrav, funksjonelle krav og maskin-/programvare krav i kapittel 5,6 og 7.
- Sluttrapporter for test
- FAT sjekkliste

10 VEDLEGG

Tabell 12 - Vedlegg

Vedlegg	Beskrivelse
1	Mal - Testrapport
2	Mal - Godkjenning av FAT-test

NB!
Utfylt vedleggsskjemaer og resultater av tester finnes i Testrapport-dokumentet.

Vedlegg 1 – TESTRAPPORT

TESTRAPPORT

Testrapportkode:	
Testnavn:	
SPORBARHET	
Krav ID:	
Test ID:	
INFORMASJON	
Testansvarlig:	
Deltakere:	
Dato:	
Utstyr:	
Forventet resultat:	
Test nummer:	
Faktisk resultat:	
FEILRAPPORT	
Feil kategori:	
Feil beskrivelse:	
Utbedring:	
Konklusjon:	
Resultat:	

Vedlegg 2 – Godkjenning av FAT-test

I dette dokumentet finnes underskrift fra Kongsberg Maritimes representant og prosjektgruppens deltakere. Dokumentet bekrefter resultatet av Factory Acceptance Test (FAT) gjennomført 20.05.2016 på Kongsberg Maritime møterom West Mystic.

GODKJENNING

Ved signering bekreftes det at FAT er gjennomført og godkjent.

Dokumentet er lest og godkjent av følgende representanter:

Dato: 20. Mai 2016

_____
Qui-Huu Le-Viet
Ekstern Veileder

_____
Aleksander Tokle Poverud

_____
Masoud Shah Pasand

_____
Tor Gunnar Finnerud

_____
Hanna Kåsastul

_____
Paul Knutson Sæther

_____
Abdurahman Senkaya

TESTRAPPORT

PROSJEKT	K-Spider		
OPDRAGSGIVER	Kongsberg Maritime AS		
UTFØRT VED	Høgskolen i Sørøst-Norge, avd. Kongsberg		
REVISJON	1.0		
MEDLEMMER	Aleksander Tokle Poverud, Masoud Shah Pasand, Tor Gunnar Finnerud, Hanna Kåsastul, Paul Knutson Sæther, Abdurahman Senkaya		
Dokumenthistorikk	**REVISJON**	**UTGITT**	**BESKRIVELSE**
	1.0	20.05.2016	Første utgave

INNHOLDSFORTEGNELSE

1 DOKUMENTHISTORIE 3

2 INNLEDNING 3

3 SAMMENDRAG OG KONKLUSJON 4

3.1 Sammendrag 4

3.2 Konklusjon 4

4 TESTFASER OG TESTRAPPORTER 5

4.1 Pre-Alfa-Test 5

4.2 ALFA-TEST 6

4.3 BETA-TEST 12

4.4 IAT-TEST 17

4.5 FAT-TEST 19

5 VEDLEGG 23

6 REFERANSER 23

TABELLER

Tabell 1 - Dokumenthistorie 3

Tabell 2 - Resultater fra Pre-Alfa-test 5

Tabell 3 - Trap1 (Styring av Hexapoden med ekstern kontroller) 6

Tabell 4 - Trap2 (Botfunksjon og oppkobling) 7

Tabell 5 - Trap3 (RCU til Raspberry Pi 1 via Modbus RS 232), testnummer 1 7

Tabell 6 - Trap3 (RCU til Raspberry Pi 1 via Modbus RS 232), testnummer 2 9

Tabell 7 - Trap4 (Raspberry Pi 1 til Raspberry Pi 2 via Wi-Fi) 10

Tabell 8 - Trap5 (Enkle digitale serielle funksjoner Hexapod) 11

Tabell 9 - Trap6 (Inspeksjonstest av kommunikasjonsnettverket) 13

Tabell 10 - Trap7 (Avanserte analoge serielle funksjoner Hexapod) 14

Tabell 11 - Trap8 (Fullstendig systemtest), testnummer 1 15

Tabell 12 - Trap8 (Fullstendig systemtest), testnummer 2 15

Tabell 13 – Resultater fra Internal Acceptance Test 17

Tabell 14 - Feiliste fra Internal Acceptance Test 18

Tabell 15 - Resultater fra Factory Acceptance Test 20

Tabell 16 - Feiliste fra Factory Acceptance Test 21

Tabell 17 - Vedlegg 23

1 DOKUMENTHISTORIE

Tabell 1 - Dokumenthistorie

VERSJON NR:	DATO ENDRET	ENDRET AV	GODKJENT AV	BESKRIVELSE
0.1	12.05.2016	Masoud	Tor Gunnar	• Dokumentet ble opprettet
0.2	13.05.2016	Masoud	Tor Gunnar	• Lagt inn testrapporter for testfasene.
0.3	19.05.2016	Masoud	Aleksander	• Lagt inn to nye testrapporter
1.0	20.05.2016	Masoud	Aleksander	• Lagt inn resultater fra FAT • Dokumentet er godkjent og utgitt

2 INNLEDNING

Dette dokumentet presenterer testrapporter og testresultater som er gjort i forbindelse med bygging av en Autonom mobil sensorplattform, (robot) som skal kontrolleres via KM sitt kontrollsystem. Dokumentet tar for seg testing av rammekrav, funksjonskrav og maskin/programvarekrav fremsatt av Kongsberg Maritime. En beskrivelse av kravene finnes i kravspesifikasjonsdokumentet [1]. Vi har brukt to typer testmetoder, inspeksjonstest og funksjonstest. Testprosedyren er beskrevet i testplan hvor også testspesifikasjonen ligger vedlagt [2]. Som en kommentar til de forskjellige testene brukes det, godkjent eller ikke godkjent. Her blir prioritering av feilene klassifisert fra A-D i synkende rekkefølge der feilkategori A og B må fikses før leveranse. Et eksempel vil være et C-krav som vil kunne leveres med feilkategori C eller D.

3 SAMMENDRAG OG KONKLUSJON

3.1 Sammendrag

Under testene ble samtlige rammekrav, funksjonelle krav, maskin- og programvarekrav godkjent. Resultatet er spesifisert i tabell 9, 10 og 11 i Testplan-dokumentet. Ved test av funksjonelle krav ble 4 av 5 krav godkjent. Avviket her skyldes av at den ene testen (Test ID: TF04 og Krav ID: KF04) var en tilleggskrav(C-krav) som ikke var implementert. Det samme gjelder maskin-/programvare krav (Test ID: TM05 og Krav ID: KM05) som heller ikke var implementert i systemet. FAT ble godkjent med oppdragsgiver 20.05.2016.

3.2 Konklusjon

Systemet passerte samtlige A og B-krav tilhørende oppgaven. De to kravene som ikke ble godkjent var et C-krav. Totalt sett er oppdragiveren fornøyd med prosjektgruppens innsatt og produktet.

4 TESTFASER OG TESTRAPPORTER

4.1 Pre-Alfa-Test

Pre-Alfa-testing ble utført ved å sjekke at alle komponenter og subsystemer er funksjonelle før vi implementerer alt i det fulle systemet. Vi analyserte krav, design og hvordan implementasjonen av systemet skulle bli. I denne fasen testet vi hver enkel komponent for å verifisere at de fungerer. F.eks. kommunikasjon mellom AIM-RCU, testet at batteriene ga riktig spenning, Hexapoden er riktig koblet opp og ingen ødelagte komponenter. Med dette fikk vi et godt overblikk om hvordan hver komponent skal fungere i systemet. Mye av denne testen ble gjort forløpende for å avdekke eventuelle feil som kan oppstå når alle delsystemene blir sammenkoblet.

Tabell 2 - Resultater fra Pre-Alfa-test

Hva er testet?	Bestått test	Ikke bestått test
Kommunikasjon mellom AIM-RCU		
Batterier gir riktig spenning		
Hexapoden er riktig koblet opp		
Hexapoden reagerer med PS2-kontroller		
Raspberry PI fungerer riktig etter installasjon av operativsystem.		
Ingen defekte komponenter		

Fargekode

Godkjent	Ikke godkjent

Konklusjon

Resultatene var som forventet og tilfredsstillende. Ingen feil ble avdekket.

Resultat

Godkjent	Ikke godkjent

4.2 ALFA-TEST

Alfa-testen gikk videre på å ta alle de komponentene vi testet i pre-alfa og implementere disse sammen til det fulle systemet. Selve testen ble først utført når det var kommunikasjon mellom alle delsystemene. I kapitlene som kommer er det testrapporter fra testene som ble utført.

Tabell 3 - Trap1 (Styring av Hexapoden med ekstern kontroller)

Testrapportkode:	Trap1
Testnavn:	Styring av Hexapoden med ekstern kontroller
SPORBARHET	
Krav ID:	KF03
Test ID:	TF03
INFORMASJON	
Testansvarlig:	Tor Gunnar Finnerud
Deltakere:	Abdurahman Senkaya, Aleksander Tokle Poverud
Dato:	24.02.2016
Utstyr:	Fullmontert Hexapod, PS2 fjernkontroll, Batteri til servokortet og Arduino
Forventet resultat:	At Hexapoden kan styres av ekstern håndholdt kontroll
Testnummer:	1
Faktisk resultat:	Hexapoden reagerte som forventet på kommandoer gitt via håndkontrolleren.
FEILRAPPORT	
Feil kategori:	Ingen
Feil beskrivelse:	Ingen
Konklusjon:	Testen ble godkjent ved at håndkontrolleren gir kommandoer til hexapoden, som den utfører.
Resultat:	**Godkjent**

Tabell 4 - Trap2 (Botfunksjon og oppkobling)

Testrapportkode:	Trap2
Testnavn:	Botfunksjon og oppkobling
SPORBARHET	
Krav ID:	KR02, KR03, KF01
Test ID:	TR02, TR03, TF01
INFORMASJON	
Testansvarlig:	Tor Gunnar Finnerud
Deltakere:	Abdurahman Senkaya, Aleksander Tokle Poverud
Dato:	25.02.2016
Utstyr:	Fullmontert Hexapod, USB-kabel, PC til å kjøre Arduino kode
Forventet resultat:	At plattformens deler utfører bevegelsene korrekt i følge til datablad
Testnummer:	1
Faktisk resultat:	Et ben bevegde seg ikke langt nok
FEILRAPPORT	
Feil kategori:	**C**
Feil beskrivelse:	Bevegelsene var ikke som de skulle.
Utbedring:	Viste seg at et par av servomotorene ikke koblet helt riktig. Noe som da ble gjort.
Konklusjon:	Testen ble godkjent etter at servo koblingen ble utbedret.
Resultat	**Godkjent**

Tabell 5 - Trap3 (RCU til Raspberry Pi 1 via Modbus RS 232), testnummer 1

Testrapportkode:	Trap3

Testnavn:	RCU til Raspberry Pi 1 via Modbus RS 232
SPORBARHET	
Krav ID:	KR04, KF02, KM01, KM02, KM03
Test ID:	TR04, TF02, TM01, TM02, TM03
INFORMASJON	
Testansvarlig:	Tor Gunnar Finnerud
Deltakere:	Hanna Kåsastul, Tor Gunnar Finnerud
Dato:	17.03.2016
Utstyr	RCU, OS, Modbus, RPI 1, skjerm, tastatur og mus
Forventet resultat:	Det skulle være mulig å lese informasjon fra RCU på Raspberry Pi 1
Testnummer:	1
Faktisk resultat:	Det er ikke blitt mottatt den ønskelige informasjonen fra RCU
FEILRAPPORT	
Feil kategori:	A
Feil beskrivelse:	Det har blitt mottatt en rekke med nullverdier istedenfor de faktiske sendte verdiene
Utbedring:	Koden har blitt omprogrammert med hensyn til feil Modbus adresser
Konklusjon:	Har ikke mottatt noe informasjon
Resultat	**Ikke godkjent**

Tabell 6 - Trap3 (RCU til Raspberry Pi 1 via Modbus RS 232), testnummer 2

Testrapportkode:	Trap3
Testnavn:	RCU til Raspberry Pi 1 via Modbus RS 232
SPORBARHET	
Krav ID:	KR04, KF02, KM01, KM02, KM03
Test ID:	TR04, TF02, TM01, TM02, TM03
INFORMASJON	
Testansvarlig:	Tor Gunnar Finnerud
Deltakere:	Hanna Kåsastul, Tor Gunnar Finnerud
Dato:	30.03.2016
Utstyr:	RCU, OS, Modbus, RPI 1, skjerm, tastatur og mus
Forventet resultat:	Det skulle være mulig å lese informasjon fra RCU på Raspberry Pi 1
Testnummer:	2
Faktisk resultat:	RPI1 har mottatt ønskelig informasjon fra RCU
FEILRAPPORT	
Feil kategori:	D
Feil beskrivelse:	Selv om at informasjonen var mottatt med riktige verdier var det fortsatt vanskelig å håndtere for videre bruk
Utbedring:	Programmeringen har blitt utbedret for lettere å håndtere informasjon senere i systemet
Konklusjon:	Det er blitt mottatt den ønskelige informasjonen fra RCU
Resultat	**Godkjent**

Tabell 7 - Trap4 (Raspberry Pi 1 til Raspberry Pi 2 via Wi-Fi)

Testrapportkode:	Trap4
Testnavn:	Raspberry Pi 1 til Raspberry Pi 2 via Wi-Fi
SPORBARHET	
Krav ID:	KF05
Test ID:	TF05
INFORMASJON	
Testansvarlig:	Tor Gunnar Finnerud
Deltakere:	Paul Knutson Sæther, Tor Gunnar Finnerud
Dato:	31.03.2016
Utstyr:	OS, RCU, Modbus RS232, tastatur/mus, skjerm, Wifi modul, Power Supply, RPi 1 og RPi 2.
Forventet resultat:	Det skulle bli mulig å overføre informasjon fra Raspberry Pi 1 til Raspberry Pi 2 via Wi-Fi
Testnummer:	1
Faktisk resultat:	Informasjon har blitt mottatt riktig
FEILRAPPORT	
Feil kategori:	Ingen feil
Feil beskrivelse:	Ingen feil
Utbedring:	Ingen utbedring nødvendig
Konklusjon:	Resultatet har blitt mottatt som forventet
Resultat:	**Godkjent**

Tabell 8 - Trap5 (Enkle digitale serielle funksjoner Hexapod)

Testrapportkode:	Trap5
Testnavn:	Enkle digitale serielle funksjoner Hexapod
SPORBARHET	
Krav ID:	KR01, KR02, KR03, KF01, KF03
Test ID:	TR01, TR02, TR03, TF01, TF03
INFORMASJON	
Testansvarlig:	Aleksander Tokle Poverud
Deltakere:	Aleksander Tokle Poverud
Dato:	13.04.2016
Utstyr	Hexapod & data & seriell programvare
Forventet resultat:	At digitale hexapod funksjoner kan styres ved serielle inputs fra en data
Testnummer:	1
Faktisk resultat:	Hexapod aktiverte forskjellige digitale funksjoner
FEILRAPPORT	
Feil kategori:	Ingen
Feil beskrivelse:	Ingen
Utbedring:	Noen flere funksjoner må legges til
Konklusjon:	Hexapod utfører digitale serielle kommandoer fra en data og er klar til å kobles til en RPi
Resultat	Godkjent

4.3 BETA-TEST

Beta-testen ble utført når alle delsystemene var koblet sammen og produktet vårt hadde en stabil arkitektur. Utførelse av Beta-testen var for å avduke feil i kommunikasjonsnettverket og bearbeide funksjonaliteten i systemet slik at det endelige produktet tilfredsstilte kravene fra oppdragsgiveren.

Tabell 9 - Trap6 (Inspeksjonstest av kommunikasjonsnettverket)

Testrapportkode:	Trap6
Testnavn:	Inspeksjonstest av kommunikasjonsnettverket
SPORBARHET	
Krav ID:	KR01, KR02, KR03, KM01, KM02, KM03
Test ID:	TR01, TR02, TR03, TM01, TM02, TM03
INFORMASJON	
Testansvarlig:	Tor Gunnar Finnerud
Deltakere:	Hanna Kåsastul, Tor Gunnar Finnerud
Utstyr:	OS, RCU, Modbus RS232, tastatur/mus, skjerm, Wifi modul, Power Supply, fullstendig Hexapod, RPi 1 og RPi 2
Dato:	22.04.2016
Forventet resultat:	Alle delsystemene og komponentene er tilkoblet riktig og utseende er akseptabelt
Testnummer:	1
Faktisk resultat:	Delsystemene har blitt tilkoblet riktig og utseende er akseptabelt
FEILRAPPORT	
Feil kategori:	Ingen feil
Feil beskrivelse:	Ingen feil
Utbedring:	Ingen utbedring nødvendig
Konklusjon:	Alt har blitt koblet riktig og utseende er akseptabelt
Resultat:	Godkjent

Tabell 10 - Trap7 (Avanserte analoge serielle funksjoner Hexapod)

Testrapportkode:	Trap7
Testnavn:	Avanserte analoge serielle funksjoner Hexapod
SPORBARHET	
Krav ID:	KR01, KR02, KR03, KF01, KF03
Test ID:	TR01, TR02, TR03, TF01, TF03
INFORMASJON	
Testansvarlig:	Aleksander Tokle Poverud
Deltakere:	Aleksander Tokle Poverud
Dato:	27.04.2016
Utstyr	Hexapod & data & seriell programvare
Forventet resultat:	Avanserte analoge hexapod funksjoner kan styres ved serielle inputs fra en data
Testnummer:	1
Faktisk resultat:	Hexapod responderte stabilt med jevne avanserte bevegelser
FEILRAPPORT	
Feil kategori:	Ingen
Feil beskrivelse:	Ingen
Utbedring:	Flere analoge funksjoner må legges til
Konklusjon:	Hexapod utfører analoge serielle kommandoer fra en data og er klar til styres fra en RPi
Resultat	Godkjent

Tabell 11 - Trap8 (Fullstendig systemtest), testnummer 1

Testrapportkode:	Trap8
Testnavn:	Fullstendig systemtest
SPORBARHET	
Krav ID:	KR04, KF02, KF05, KM02, KM03
Test ID:	TR04, TF02, TF05, TM02, TM03
INFORMASJON	
Testansvarlig:	Tor Gunnar Finnerud
Deltakere:	Aleksander Tokle Poverud, Masoud Shah Pasand, Tor Gunnar Finnerud, Hanna Kåsastul, Paul Knutson Sæther, Abdurahman Senkaya
Dato:	28.04.2016
Utstyr	OS, RCU, Modbus RS232, tastatur/mus, skjerm, Wifi modul, Power Supply, fullstendig Hexapod, RPi 1 og RPi 2
Forventet resultat:	Hexapoden blir kontrollert av kommandoer fra AIM
Testnummer:	1
Faktisk resultat:	Kommunikasjonen fungerte hele veien, men det var ikke spesielt brukervennlig. Ingen funksjoner til å stoppe roboten.
FEILRAPPORT	
Feil kategori:	C
Feil beskrivelse:	Manglende ekstra funksjoner samt ikke implementert batteriløsning
Utbedring:	Finne en batteriløsning og implementere nye brukerfunksjoner for start og stop.
Konklusjon:	Systemet fungerte, men vi avbrøyt testen for å rette på feilene og gjøre systemet enda mer brukervennlig.
Resultat	**Ikke godkjent**

Tabell 12 - Trap8 (Fullstendig systemtest), testnummer 2

Testrapportkode:	Trap8
Testnavn:	Fullstendig systemtest
SPORBARHET	
Krav ID:	KR04, KF02, KF05, KM02, KM03
Test ID:	TR04, TF02, TF05, TM02, TM03
INFORMASJON	
Testansvarlig:	Tor Gunnar Finnerud
Deltakere:	Aleksander Tokle Poverud, Masoud Shah Pasand, Tor Gunnar Finnerud, Hanna Kåsastul, Paul Knutson Sæther, Abdurahman Senkaya
Dato:	04.05.2016
Utstyr	OS, RCU, Modbus RS232, tastatur/mus, skjerm, Wifi modul, Power Supply, fullstendig Hexapod, RPi 1 og RPi 2
Forventet resultat:	Hexapoden blir kontrollert av kommandoer fra AIM
Testnummer:	2
Faktisk resultat:	Kommunikasjonen fungerte hele veien
FEILRAPPORT	
Feil kategori:	Ingen
Feil beskrivelse:	Ingen
Utbedring:	Finne en batteriløsning og implementere nye brukerfunksjoner for start og stop.
Konklusjon:	Systemet fungerte akseptabelt. Vi har kjøpt en Powerbank som en mulig batteriløsning og satt inn start og stopp funksjon.
Resultat	**Godkjent**

4.4 IAT-TEST

Tabell 13 – Resultater fra Internal Acceptance Test

IAT sjekkliste			
TESTANSVARLIG: Tor Gunnar Finnerud	**DATO:** 12.05.2016		
DELTAKERE: Aleksander Tokle Poverud, Masoud Shah Pasand, Tor Gunnar Finnerud, Hanna Kåsastul, Paul Knutson Sæther, Abdurahman Senkaya.			
TESTMETODE OG GODKJENNINGSKRITERIUM	**TEST ID**	**KRAV ID**	**RESULTAT**
TEST AV RAMMEKRAV			
Å koble komponentene til roboten sammen og observere visuelt et fysisk legeme. Testen er godkjent ved inspeksjon at det er observert et fysisk legeme, og at bevegelige elementene er inkludert.	TR01	KR01	Godkjent
Å tilkoble roboten til et styresystem og observere robotens bevegelser. Testen er godkjent når robotens lemmer beveger seg som forventet.	TR02	KR02	Godkjent
Å teste funksjonene og observere utseende. Testen er godkjent når funksjonene er verifiserte og utseende ser presentabelt ut.	TR03	KR03	Godkjent
Å påse at det er RCU510 som utfører hoved sekvensene på roboten. Testen er godkjent når kontrollen av roboten skjer via RCU510 fra AIM7.	TR04	KR04	Godkjent
Å verifisere funksjonene hver for seg og samhandling mellom hverandre. Testen er godkjent når roboten reagerer på inputen fra sensorer som forventet.	TR05	KR05	Godkjent
TEST AV FUNKSJONELLE KRAV			
Å utføre en praktisk test, og observere resultatet. Testen er godkjent når roboten utfører sammensatte bevegelser.	TF01	KF01	Godkjent
Å koble systemet sammen og overføre data fra kontrollsystemet AIM7 via RCU510 til roboten. Testen er godkjent når roboten responderer på kommandoer fra kontrollsystemet AIM7 via RCU510.	TF02	KF02	Godkjent
	TF03	KF03	Godkjent

Å koble roboten og en ekstern fjernkontroll sammen og sende kommandoer fra fjernkontrollen til roboten. Testen er godkjent når roboten responderer på kommandoer fra en ekstern fjernkontroll.			
Å koble AIM7, RCU510 og roboten sammen og overføre måle - data fra robotens sensorer til operatørsystemet. Testen er godkjent når det skal være mottatt informasjon fra roboten til Operatør via kommunikasjonsnettverket.	TF04	KF04	Ikke implementert (C-krav)
Å sende kommandoer fra AIM7 trådløst via WiFi til roboten. Testen er godkjent når roboten responderer på kommandoer AIM7 via WiFi.	TF05	KF05	Godkjent
TEST AV MASKIN-/PROGRAMVARE KRAV			
Å observere en operatørstasjon. Testen er godkjent når det skal være observert at operatørstasjonen er tilkoblet til systemet og begge to er operative.	TM01	KM01	Godkjent
Å verifisere kontroll systemet. Testen er godkjent når RCU510 er tilkoblet og utfører styring av roboten.	TM02	KM02	Godkjent
Å verifisere kontroll systemet. Testen er godkjent når AIM7 styrer roboten.	TM03	KM03	Godkjent
Å skape en situasjon hvor installerte sensorene ville reagere. Testen er godkjent når installerte sensorene er observert og måledata skal være avlest av roboten og sendt til operatørstasjon.	TM04	KM04	Godkjent
Å skape en situasjon hvor installerte sensorene ville reagere. Testen er godkjent når installerte måledata er observert og sensorene skal være sendt til operatørstasjon fra roboten.	TM05	KM05	Ikke implementert (C-krav)

Tabell 14 - Feiliste fra Internal Acceptance Test

FEILLISTE			
TEST ID	**KRAV ID**	**FEIL/KOMMENTAR**	**FEILKATEGORI**
TF04	KF04	Ikke utført (tilleggskrav)	
TM05	KM05	Ikke utført (tilleggskrav)	

4.5 FAT-TEST

Testen gjennomføres den 20.05.2016 med representanter fra Kongsberg Maritime og studentgruppen på Kongsberg Maritime.

Ting som må være på plass før FAT:

- IAT må være fullført.
- Feilkategori A og B må rettes.
- Produktet må være helt klar, slik det kan overleveres til oppdragsgiver.
- Godkjenningskriteriene må være presise og klare for oppdragsgiver.

Ting som må gjøres under FAT:

- Kravspesifikasjons-dokumentet må være med
- Alle punktene på sjekklisten må sjekkes
- Testen må utføres med oppdragsgiver tilstede
- Må få dokumentert fra oppdragsgiver (underskrift) på at kravene er tilfredsstilt

Konklusjon:

Testen ble utført 20.05.2016 på Kongsberg Maritime. Tabell 15 og 16 viser resultater og feilliste. Vedlegg følger også godkjenning fra oppdragsgiver med signatur. Det er en C-feil som må rettes på før 3 presentasjon og to tilleggskrav (C-krav) som ikke er implementert i systemet. Oppdragsgiveren var fornøyd med prosjektgruppens innsatt og resultatet av oppgaven.

Tabell 15 - Resultater fra Factory Acceptance Test

FAT sjekkliste			
TESTANSVARLIG: Tor Gunnar Finnerud	**DATO:** 20.05.2016		
DELTAKERE: Aleksander Tokle Poverud, Masoud Shah Pasand, Tor Gunnar Finnerud, Hanna Kåsastul, Paul Knutson Sæther, Abdurahman Senkaya.			
TESTMETODE OG GODKJENNINGSKRITERIUM	**TEST ID**	**KRAV ID**	**RESULTAT**
TEST AV RAMMEKRAV			
Å koble komponentene til roboten sammen og observere visuelt et fysisk legeme. Testen er godkjent ved inspeksjon at det er observert et fysisk legeme, og at bevegelige elementene er inkludert.	TR01	KR01	Godkjent
Å tilkoble roboten til et styresystem og observere robotens bevegelser. Testen er godkjent når robotens lemmer beveger seg som forventet.	TR02	KR02	Godkjent
Å teste funksjonene og observere utseende. Testen er godkjent når funksjonene er verifiserte og utseende ser presentabelt ut.	TR03	KR03	Godkjent
Å påse at det er RCU510 som utfører hoved sekvensene på roboten. Testen er godkjent når kontrollen av roboten skjer via RCU510 fra AIM7.	TR04	KR04	Godkjent (Med feilkategori C)
Å verifisere funksjonene hver for seg og samhandling mellom hverandre. Testen er godkjent når roboten reagerer på inputen fra sensorer som forventet.	TR05	KR05	Godkjent (Med feilkategori C)
TEST AV FUNKSJONELLE KRAV			
Å utføre en praktisk test, og observere resultatet. Testen er godkjent når roboten utfører sammensatte bevegelser.	TF01	KF01	Godkjent
Å koble systemet sammen og overføre data fra kontrollsystemet AIM7 via RCU510 til roboten. Testen er godkjent når roboten responderer på kommandoer fra kontrollsystemet AIM7 via RCU510.	TF02	KF02	Godkjent
Å koble roboten og en ekstern fjernkontroll sammen og sende kommandoer fra fjernkontrollen til roboten. Testen er godkjent når roboten responderer på kommandoer fra en ekstern fjernkontroll.	TF03	KF03	Godkjent

Å koble AIM7, RCU510 og roboten sammen og overføre måle - data fra robotens sensorer til operatørsystemet. Testen er godkjent når det skal være mottatt informasjon fra roboten til Operatør via kommunikasjonsnettverket.	TF04	KF04	Ikke implementert (C-krav)
Å sende kommandoer fra AIM7 trådløst via WiFi til roboten. Testen er godkjent når roboten responderer på kommandoer AIM7 via WiFi.	TF05	KF05	Godkjent
TEST AV MASKIN-/PROGRAMVARE KRAV			
Å observere en operatørstasjon. Testen er godkjent når det skal være observert at operatørstasjonen er tilkoblet til systemet og begge to er operative.	TM01	KM01	Godkjent
Å verifisere kontroll systemet. Testen er godkjent når RCU510 er tilkoblet og utfører styring av roboten.	TM02	KM02	Godkjent
Å verifisere kontroll systemet. Testen er godkjent når AIM7 styrer roboten.	TM03	KM03	Godkjent
Å skape en situasjon hvor installerte sensorene ville reagere. Testen er godkjent når installerte sensorene er observert og måled ata skal være avlest av roboten og sendt til operatørstasjon.	TM04	KM04	Godkjent
Å skape en situasjon hvor installerte sensorene ville reagere. Testen er godkjent når installerte måledata er observert og sensorene skal være sendt til operatørstasjon fra roboten.	TM05	KM05	Ikke implementert (C-krav)

Tabell 16 - Feiliste fra Factory Acceptance Test

FEILLISTE			
TEST ID	**KRAV ID**	**FEIL/KOMMENTAR**	**FEILKATEGORI**
TR04	KR04	Oppstart og stoppfunksjon	C
TR05	KR05	Sensor distanse må endres	C
TF04	KF04	Ikke utført (Tillegskrav)	
TM05	KM05	Ikke utført (Tillegskrav)	

Høgskolen i Sørøst-Norge

KONGSBERG

I dette dokumentet finnes underskrift fra Kongsberg Maritimes representant og prosjektgruppens deltakere. Dokumentet bekrefter resultatet av Factory Acceptance Test (FAT) gjennomført 20.05.2016 på Kongsberg Maritime møterom West Mystic.

GODKJENNING

Ved signering bekreftes det at FAT er gjennomført og godkjent.

Dokumentet er lest og godkjent av følgende representanter:

Dato: 20. Mai 2016

Qui-Huu Le-Viet
Ekstern Veileder

Aleksander Tokle Poverud

Masoud Shah Pasand

Tor Gunnar Finnerud

Hanna Kåsastul

Paul Knutson Sæther

Abdurahman Senkaya

5 VEDLEGG

Tabell 17 - Vedlegg

Vedlegg	Beskrivelse
1	Original dokument for godkjennelse av FAT og krav.

6 REFERANSER

[1] K-Spider 2016, *Kravspesifikasjon Rev 2.0,* Høgskolen i Sørøst-Norge, avdeling Kongsberg.

[2] K-Spider 2016, *Testplan Rev 3.0,* Høgskolen i Sørøst-Norge, avdeling Kongsberg.

Høgskolen i Sørøst-Norge

KONGSBERG

ØKONOMIDOKUMENT

PROSJEKT	K-Spider		
OPDRAGSGIVER	Kongsberg Maritime AS		
UTFØRT VED	Høgskolen i Sørøst-Norge, avd. Kongsberg		
REVISJON	2.0		
MEDLEMMER	Aleksander Tokle Poverud, Masoud Shah Pasand, Tor Gunnar Finnerud, Hanna Kåsastul, Paul Knutson Sæther, Abdurahman Senkaya		
Dokumenthistorikk	**REVISJON**	**UTGITT**	**BESKRIVELSE**
	1.0	17.02.2016	Første utgave
	2.0	20.05.2016	Andre utgave

INNHOLDSFORTEGNELSE

1 DOKUMENTHISTORIE 3
2 INNLEDNING 3
3 BUDSJETT 4
4 REGNSKAP 5
5 NYTTIG INFORMASJON OM INNKJØP 7

TABELLER

Tabell 1 Dokumenthistorie 3
Tabell 2 - Materiell budsjettert 4
Tabell 3 - Totale kostnader budsjettert 4
Tabell 4 - Administrative kostnader 5
Tabell 5 - Materielle kostnader 6
Tabell 6 - Totalt kostnader 7
Tabell 7 - Informasjon om leverandører 9
Tabell 8 - Begrunnelse for valg av leverandører 9

FIGURER

Figur 1 - Flowchart til innkjøp 8

1 DOKUMENTHISTORIE

Tabell 1 Dokumenthistorie

VERSJON NR:	DATO ENDRET	ENDRET AV	GODKJENT AV	BESKRIVELSE
0.1	16.02.2016	Masoud	Aleksander	• Dokumentet ble opprettet
1.0	17.02.2016	Masoud	Aleksander	• Oppdatert utgiftene • Dokumentet er godkjent og utgitt.
1.1	11.05.2016	Masoud	Aleksander	• Regnskap oppdatert
1.2	16.05.2016	Masoud	Aleksander	• Lagt inn nødvendig informasjon i kapittel 5. • Figur 1
2.0	20.05.2016	Masoud	Aleksander	• Godkjent og utgitt

2 INNLEDNING

Dette dokumentet viser oversikt over utgiftene som prosjektgruppen har hatt i løpet av prosjektet. Det ble ikke gitt konkrete budsjettrammer fra Kongsberg Maritime, men budsjettet skal holde seg innenfor rimelighetens grenser. Kongsberg Maritime dekker programvare og maskinvare som blir brukt igjennom prosjektet og vil derfor ikke bli med i regnskapet.

Utgiftene i prosjektet er nærmere spesifisert i tabeller som kommer i de neste kapitlene.

3 BUDSJETT

Prosjektgruppen hadde utført et budsjettestimat tidlig i prosjektet for å ha en oversikt over komponenter som var nødvendig å bestille. Vi satt opp tre bestillingspakker med forskjellig kvalitet og kostnadsnivåer for de delene vi skulle ha. I samarbeid med oppdragsgiveren valgte vi den dyreste løsningen på grunn av kvaliteten.

Selv om dette estimatet viste seg å være noe feil senere i fasen, var det fortsatt viktig å ha en grov oversikt som vi måtte ta hensyn til. Det var også viktig med hensyn på leveringstiden, da noen deler kunne ta lengre tid å motta. Det var nødvendig å ha komponentene på plass når vi skulle begynne å bygge roboten.

Tabell 2 viser prosjektgruppas budsjett under hele prosjektperioden.

Tabell 2 - Materiell budsjettert

BESKRIVELSE	ANTALL	ENHETSPRIS	TOTALT
PS2 Controller	1	Kr 206,00	Kr 206,00
Adjustable Step-down Module	2	Kr 67,50	Kr 135,00
Lynxmotion -Hexapod	1	Kr 2149,00	Kr 2149,00
Servo controller	1	Kr 388,00	Kr 388,00
Robot controller	1	Kr 302,00	Kr 302,00
USB to Mini B Cable 1.3m	1	Kr 21,00	Kr 21,00
Servo Extender Cable – 6''	1	Kr 17,00	Kr 17,00
6V – 12V Battery Charger	1	Kr 189,50	Kr 189,50
Servo motor	18	Kr 233,00	Kr 4149,00
Rechargeable-Battery	2	Kr 146,00	Kr 292,00
		SUM	**Kr 7848,50**

Tabell 3 - Totale kostnader budsjettert

BESKRIVELSE	TOTALT
Materiell budsjettert	Kr 7848,50
SUM	**Kr 7848,50**

4 REGNSKAP

Dette kapitelet gir en oversikt over hvilke utgifter prosjektgruppa har hatt under hele prosjektperioden. Dette vil være et løpende dokument siden utgifter vil løpe frem til prosjektet er ferdig. Dokumentet vil bli oppdatert etter hvert som det kommer nye bestillinger eller andre endringer. Det endelige regnskapet vil dermed ikke være komplett før den siste offisielle revisjonen er utgitt.

Prosjektgruppen hadde ikke budsjettert administrative kostnader, men vi har kjøpt inn diverse ting i forbindelse med prosjektet. Dette er vist i tabell 4.

Legg merke til at alle verdier er i norske kroner (NOK).

Tabell 4 - Administrative kostnader

BESKRIVELSE	ANTALL	ENHETSPRIS	TOTALT
Strikkmappe A4	1	Kr 19,90	Kr 19,90
Brevordner, A4 50MM	1	Kr 19.90	Kr 19.90
Avisstativ	1	Kr 29,90	Kr 29,90
Register, A4	1	Kr 19.90	Kr 19.90
Presentasjonsfjernkontroll	1	Kr 299,00	Kr 299,00
Hjemmearkiv Spectrafile	1	Kr 49,00	Kr 49,00
Smårekvisita	2	Kr 13,00	Kr 26,00
Registermappe til 2.presentasjon	2	Kr 22,00	Kr 44,00
T-skjorter Classic Bomull	6	Kr 39,50	Kr 237,50
Kongsberg Logo	6	Kr 42,00	Kr 252,00
K-Spider logo 25stk (Min. Kvantum)	25	Kr 1425,00	Kr 1425,00
		SUM	**Kr 2422,00**

Tabell 5 - Materielle kostnader

BESKRIVELSE	ANTALL	ENHETSPRIS	TOTALT	BUDSJETTERT	FAKTURERT	DIFFERANSE
PS2 Controller	1	Kr 206,00	Kr 206,00	-	-	-
Adjustable Step-down Module	2	Kr 67,50	Kr 135,00	-	-	-
Lynxmotion - Hexapod	1	Kr 2149,00	Kr 2149,00	-	-	-
Servo controller	1	Kr 388,00	Kr 388,00	-	-	-
Robot controller	1	Kr 302,00	Kr 302,00	-	-	-
USB to Mini B Cable 1.3m	1	Kr 21,00	Kr 21,00	-	-	-
Servo Extender Cable – 6''	1	Kr 17,00	Kr 17,00	-	-	-
6V – 12V Battery Charger	1	Kr 189,50	Kr 189,50	-	-	-
Servo motor	23	Kr 233,00	Kr 5359,00	-	Kr 5359,00	Kr 1165,00
Rechargeable-Battery	2	Kr 146,00	Kr 292,00	-	-	-
Pixy CAM - Vision Sensor Camera	1	Kr 517,00	Kr 517,00	-	Kr 517,00	Kr 517,00
RS-232 Serial / Modbus Module for Arduino	1	Kr 158,00	Kr 158,00	-	Kr 158,00	Kr 158,00
Multiprotocol Radio Shield for Arduino	1	Kr 306,00	Kr 306,00	-	Kr 306,00	Kr 306,00
Borrelåsbånd Sort 3Meter	1	Kr 89,50	Kr 89,50	-	Kr 89,50	Kr 89,50
Smeltelim for limpistoler	1	Kr 40,00	Kr 40,00	-	Kr 40,00	Kr 40,00
Kabelstrømper 10Meter	1	Kr 433,50	Kr 433,50	-	Kr 433,50	Kr 433,50
A UBEC ''Universal Battery Elimination Circuit''	1	Kr 318,50	Kr 318,50	-	Kr 318,50	Kr 318,50
Servicekoffert	2	Kr 699,00	Kr 1398,00	-	Kr 1398,00	Kr 1398,00
Grenuttak 5-veis	1	Kr 99,00	Kr 99,00	-	Kr 99,00	Kr 99,00
Batterier 9V	2	Kr 14,00	Kr 28,00	-	Kr 28,00	Kr 28,00
Batterier AAA 4PK	1	Kr 35,00	Kr 35,00	-	Kr 35,00	Kr 35,00
	SUM	**Kr 6428,50**	**Kr 12481**	**Kr 7848,50**	**Kr 12481,00**	**Kr 4632,50**

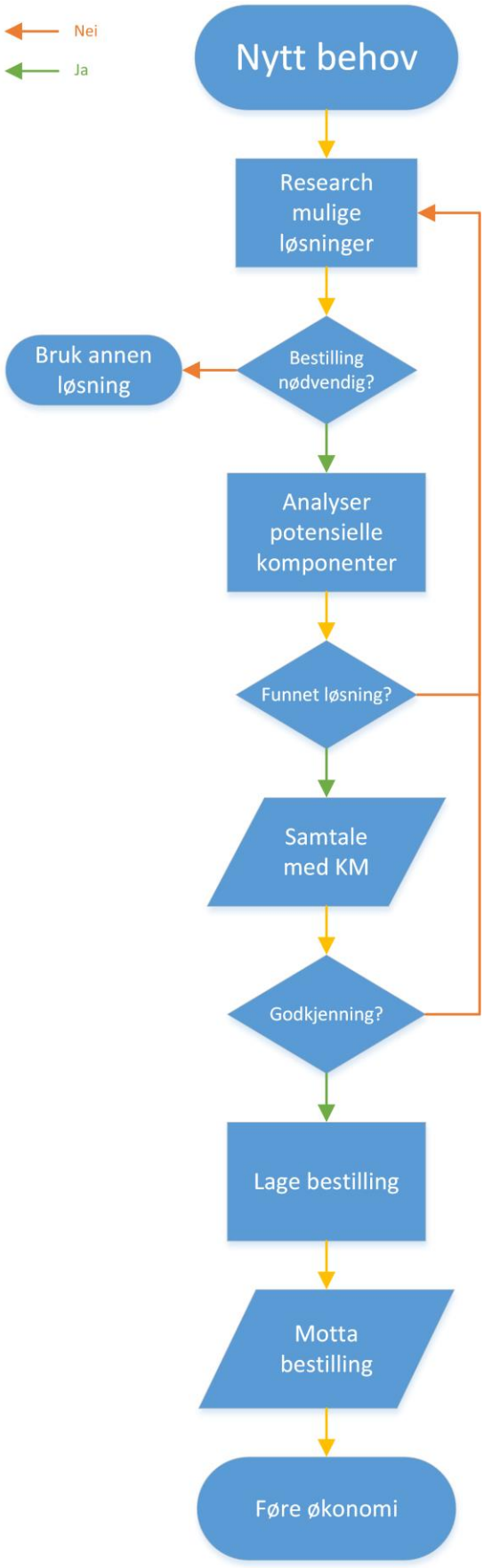
Tabell 6 - Totalt kostnader

BESKRIVELSE	BUDSJETTERT	FAKTURERT	DIFFERANSE
Materielle kostnader	Kr 7848,50	Kr 12481,00	Kr 4632,50
Administrative kostnader	Kr 0,00	Kr 2422,00	Kr 2422,00
Merverdiavgift (moms)	Kr 0,00	Kr 2931,00	Kr 2931,00
Frakt	Kr 0,00	Kr 2270,00	Kr 2270,00
SUM	**Kr 7848,50**	**Kr 20104,00**	**Kr 12255,50**

5 NYTTIG INFORMASJON OM INNKJØP

I dette kapittelet beskrives leverandørene og ressursene vi har brukt for bestilling av utstyr til systemet vi skal bygge. I tabell 7 og 8 er det kort fortalt hvorfor vi har valgt å bruke de oppgitte leverandørene samt annen relevant informasjon om de.

Innkjøp av alt nødvendig utstyr ble foretatt av prosjektgruppen i samarbeid med ekstern veileder Qui-Huu Le-Viet. Figur 1 viser et enkelt Flowchart av hvordan vi har foretatt bestillinger.



Figur 1 - Flowchart til innkjøp

Tabell 7 - Informasjon om leverandører

Produkt beskrivelse	Leverandør	Sted	Nettside
Lynxmotion Phoenix 3DOF Hexapod	RobotShop inc. (Lynxmotion)	United States	www.robotshop.com
Kofferter	Clas Ohlson	Kongsberg	http://www.clasohlson.com/no/
PIXY CAM – Vision Sensor Camera	Devantech Ltd Industrial Electronic systems	United Kingdom	http://www.robot-electronics.co.uk/
RS-232 Serial / Modbus Module for Arduino	Cooking Hacks	Spain	https://www.cooking-hacks.com/
Multiprotocol Radio Shield for Arduino	Cooking Hacks	Spain	https://www.cooking-hacks.com/
T-skjorter	TESS AS	Kongsberg	http://tess.no/tess-servicesenter/tess-kongsgaardmoen

Tabell 8 - Begrunnelse for valg av leverandører

Leverandør	Hvorfor
RobotShop Inc.	Vi valgte å kjøpe vår Hexapod robot fra RobotShop fordi butikken er ekspert på faglig robotteknologi. Butikken selger profesjonelle roboter, utviklingsplattformer, kits og spesialiserte robot deler. RobotShop er også en viktig kilde for robotikk utdanning og forskning.
Cookings Hacks	De selger elektriske komponenter som Arduino, Raspberry Pi, sensorer, aktuatorer osv. Mange av ansatte er elektro og data ingeniører som har mye erfaring innen elektronikk og programmering.
Devantech Ltd	De har 30 års erfaring innen elektronikk og tilbyr kvalitetsprodukt til en rimelig pris, samt å være nyskapende med ny design og konsepter.
Clas Ohlson	De er best på dingser og har 15 000 forskjellige dingser å velge mellom. Vi kjøpte kofferter til utstyret vår herifra på grunn av bra priser og god kvalitet. Vi handlet også forskjellige dingser som permer, avisstativ og presentasjonsfjernkontroll.
Tess AS	Vi fikk en rimelig pris på t-skjorter og logoer via en venn.

BRUKERMANUAL

PROSJEKT	K-Spider		
OPDRAGSGIVER	Kongsberg Maritime AS		
UTFØRT VED	Høgskolen i Sørøst-Norge, avd. Kongsberg		
REVISJON	1.0		
MEDLEMMER	Aleksander Tokle Poverud, Masoud Shah Pasand, Tor Gunnar Finnerud, Hanna Kåsastul, Paul Knutson Sæther, Abdurahman Senkaya		
Dokumenthistorikk	**REVISJON**	**UTGITT**	**BESKRIVELSE**
	1.0	22.05.2016	Første utgave

INNHOLDSFORTEGNELSE

1 DOKUMENTHISTORIE 4

2 INNLEDNING 4

3 BRUKERVEILEDNING 5

3.1 Kommunikasjon mellom Arduino Mega og Rpi 2 5
3.2 Kommunikasjon mellom Arduino Mega og SCC32U 5
3.3 Koble sammen Arduino og PS2 mottaker sammen 6
3.4 Komponenter 7
3.5 Kobling av servo til SCC32U 8
3.6 Ultrasonic Rpi2 10
3.7 Rpi2 kameramodul 12
3.8 Rpi2 wifi 12
3.9 Auto run Rpi2 13
3.10 VNC (Virtual Network Computing) 13

4 SPESIELLE FUNKSJONER 14

4.1 Overføring hastighet 14
4.2 Strøm tilkobling til SCC32U 15
4.2.1 VL terminal 15
4.2.2 VS1 terminal 16
4.2.3 VS2 Terminal 16
4.2.4 VS1 = VS2 Jumper 16
4.3 Batterier 18
4.3.1 6.0 vdc Ni-MH 1600mAh til SCC32U 18
4.3.2 Powerbank til Rpi2 19
4.3.4 Alternativ strømkilde til servomotorer (LiPo batteri) 20
4.3.5 Alternativ strømkilde til Rpi (Lipo batteri) 21
4.3.6 Arduino Mega minijack inngang spenning 21
4.4 PS2 funksjoner 22
4.5 Kalibrering av robotens leg 23

5 RCU til RPI1 25

5.1 OPPKOBLING AV SYSTEMET 25
5.1.1 Oppkobling av RCU og RPI 1 25
5.1.1 Operasjon av RPi1 26
5.1.2 Operasjon av RPi2 26

6 AIM 28

7 REFERANSER 32

TABELLER

Tabell 1 - Dokumenthistorie..... 4
Tabell 2 - Arduino og SCC32U..... 5
Tabell 3 PS2 mottaker..... 6
Tabell 4 - Komponenter..... 7
Tabell 5 - SCC32U VS1..... 9
Tabell 6 - SCC32 VS2..... 10
Tabell 7 - Ultrasonic & Rpi2..... 10
Tabell 8 - Wipi..... 12
Tabell 9 - Baud rate..... 14
Tabell 10 - VL terminal..... 15
Tabell 11 - VS1 terminal..... 16
Tabell 12 - VS2 terminal..... 16
Tabell 13 - NI-HM..... 18
Tabell 14 - Powerbank..... 19
Tabell 15 - UBEC spesifikasjoner..... 20

FIGURER

Figur 1 - Kommunikasjon mellom er Arduino Mega og Rpi 2..... 5
Figur 2 - Arduino mega og Scc32U..... 6
Figur 3 - PS2 mottaker..... 7
Figur 4 - led og Buzz-er..... 8
Figur 5 - Servo pinne konfigurasjon..... 9
Figur 6- Ultrasonic..... 11
Figur 7 - Rpi2 kameramodul..... 12
Figur 8 - Rpi wipi..... 13
Figur 9 - Baud rate..... 15
Figur 10 - SCC32U Power..... 17
Figur 11 - NI-MH batteri..... 18
Figur 12 - Powerbank..... 19
Figur 13 - lm2596S..... 21
Figur 14 - UBEC..... 22
Figur 15 - PS2 Funksjoner..... 23
Figur 16 - SCC-32 servo sequencer..... 24
Figur 17 - Kobling RCU 510..... 25
Figur 18 - Modbus..... 26
Figur 19 - Systemstatus i OS..... 28
Figur 20- KSpider (operasjonsvinduet)..... 28
Figur 21 - HotSpot med brytere Set1, Set2, Set3, Pause og Abort..... 29
Figur 22 - Kommandopakke Set 1..... 30
Figur 23 - Kommandopakke Set 2..... 30
Figur 24 - Kommandopakke 3..... 31

1 DOKUMENTHISTORIE

Tabell 1 - Dokumenthistorie

VERSJON NR:	DATO ENDRET	ENDRET AV	GODKJENT AV	BESKRIVELSE
0.1	16.05.2016	Abdurahman	Tor Gunnar	• Dokumentet ble opprettet
0.2	20.05.2016	Tor Gunnar	Abdurahman	• Lagt AIM og RCU biten
1.0	22.05.2016	Masoud	Aleksander	• Revidert dokumentet • Godkjent og utgitt

2 INNLEDNING

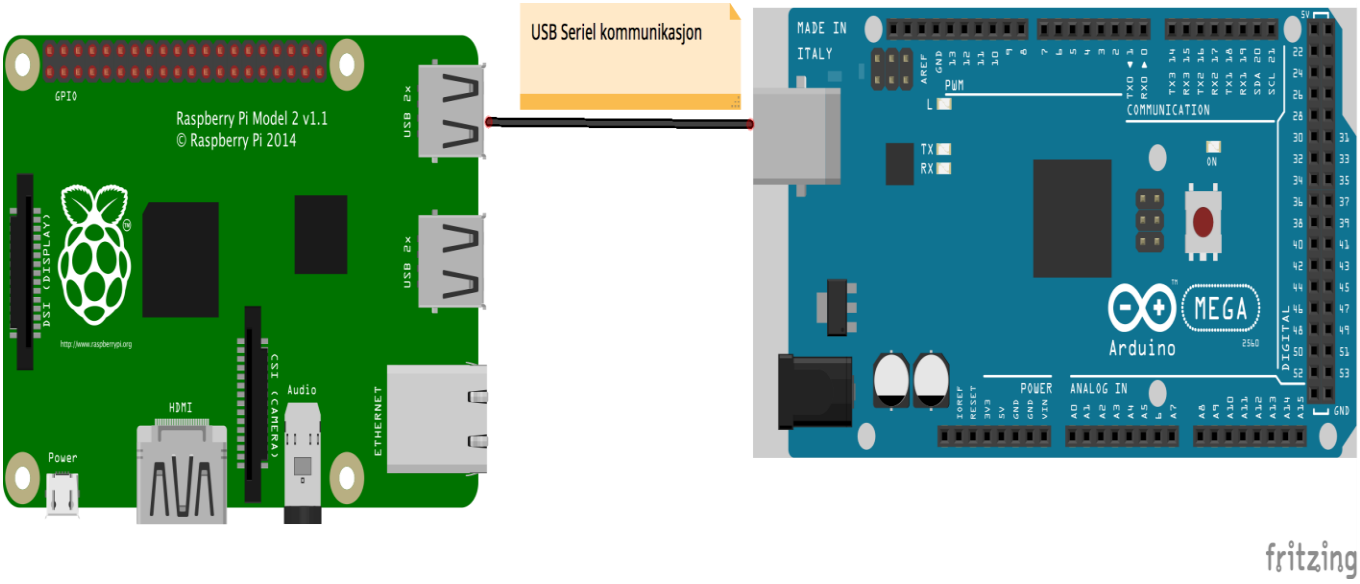
Dette dokumentet er en brukermanual for systemet, slik at nye brukere og senere grupper som skal jobbe med det, får en innføring i hvordan en skal koble opp og sette i gang systemet. Denne manualen kommer til å inneholde hvordan en kobler opp de forskjellige komponentene, hvilke koblinger som blir brukt og protokollene som er anvendt. Dette kommer til å inneholde koblingen mellom RCU og RPi1, trådløse koblingen mellom RPi1 og RPi2, mellom RPi2 og hexapod, og hvordan hexapod i seg selv har blitt koblet.

Manualen kommer også til å inneholde informasjon om hvordan man operer systemet, dette er da hva en må gjøre i AIM for å kunne kontrollerer hexapod. Det vil også følge med en oversiktsliste over verdier og symboler, og hvilke funksjoner disse tilhører.

3 BRUKERVEILEDNING

3.1 Kommunikasjon mellom Arduino Mega og Rpi 2

Vi har brukt USB seriell kobling for Rpi 2 og Arduino til å snakke sammen. Fra USB sendes 1 og 0 (bits) som inneholder nødvendig informasjon for serielt kommunikasjon. Disse bitene danner sammen et byte (bestående av 8 bits)



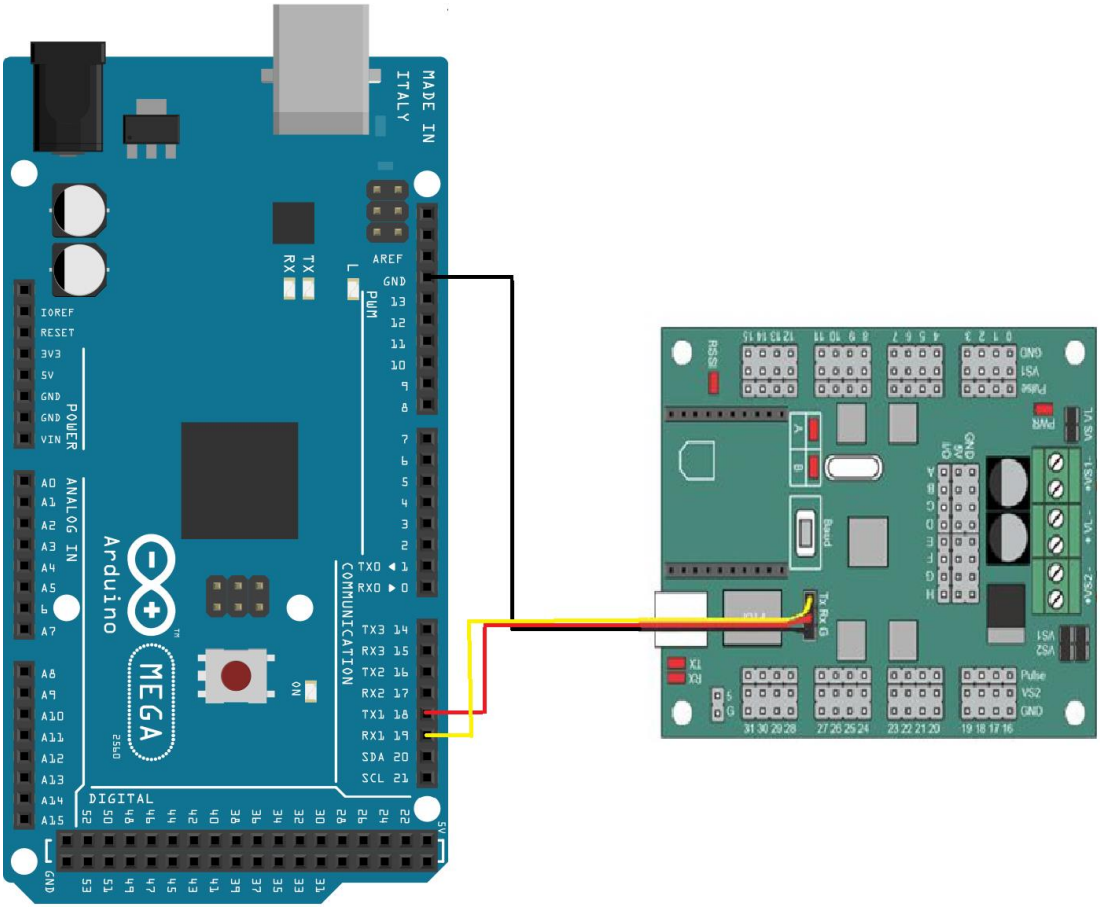
Figur 1 - Kommunikasjon mellom er Arduino Mega og Rpi 2

3.2 Kommunikasjon mellom Arduino Mega og SCC32U

Kommandoer sendes fra SSC32U ved hjelp av Tx pinne, mens kommandoer som skal mottas av SSC32U gjøres via Rx pinne. Fra disse pinnene sendes kommandoer til servo kontrolleren fra Arduino Mega. For å gjøre dette, har vi koblet Tx pinne på Arduino til Rx pinne på SSC32U, Rx pinne på Arduino til Tx pinne på SSC32 og GND til GND.

Tabell 2 - Arduino og SCC32U

Arduino Mega	SCC32U
TX	RX
RX	TX
GND	GND



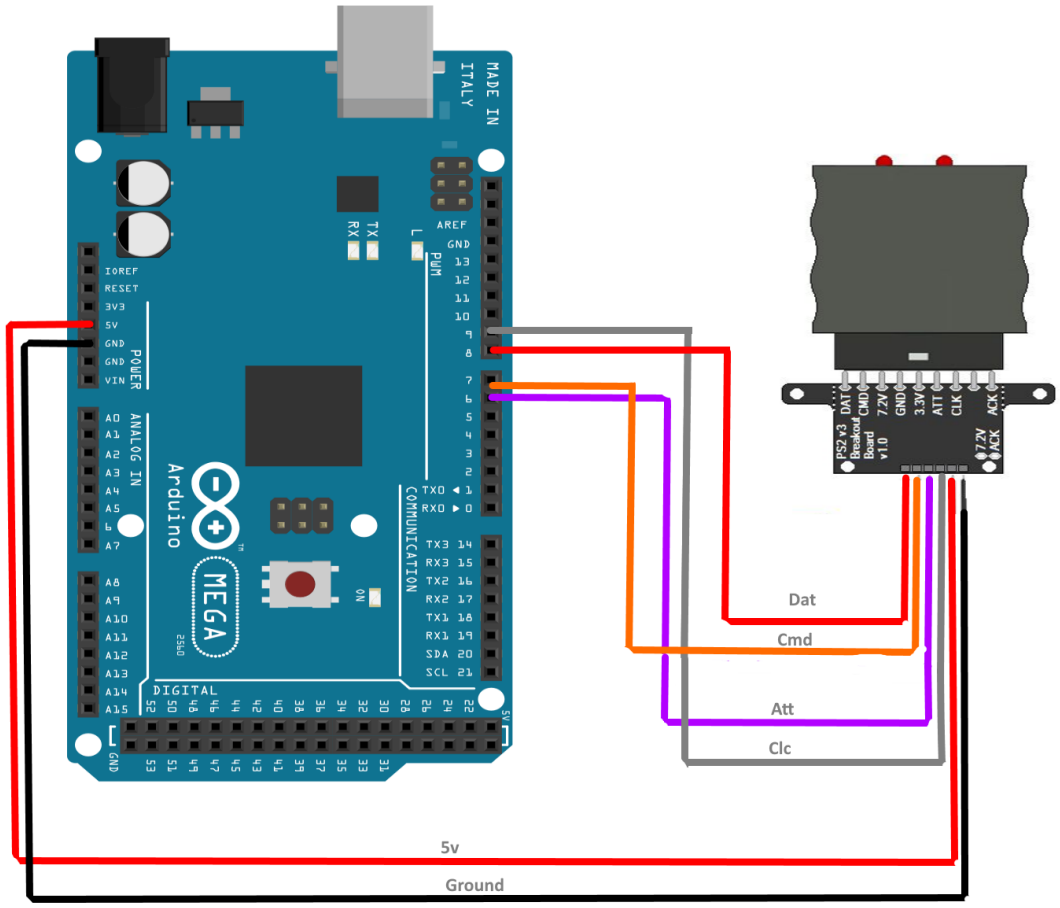
Figur 2 - Arduino mega og Scc32U

3.3 Koble sammen Arduino og PS2 mottaker sammen

Lynxmotion PS2 kontroller V4 er en 2,4 GHz trådløs Play Station 2 spillkontroller. Det inkluderer en liten mottaker modul som kobles til Arduino Mega

Tabell 3 PS2 mottaker

Pinne konfigurasjon	PS2 mottaker
Pin 6	ATT
Pin 7	CMD
Pin 8	DAT
Pin 9	CLC
5v	+
GND	-



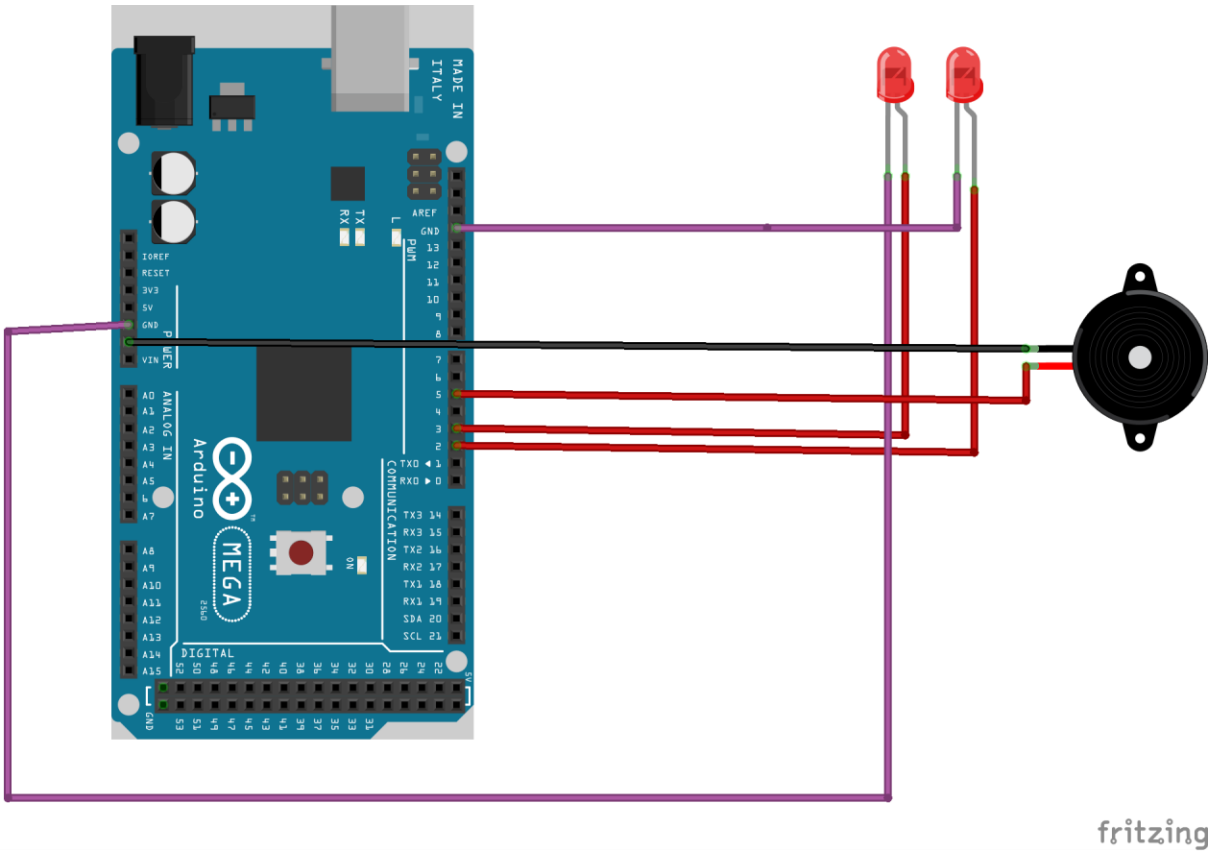
Figur 3 - PS2 mottaker

3.4 Komponenter

Vi har implementert 2 stk. led lys som forestiller robotens øye, og Buzz-er som gir beskjed når roboten er av og på.

Tabell 4 - Komponenter

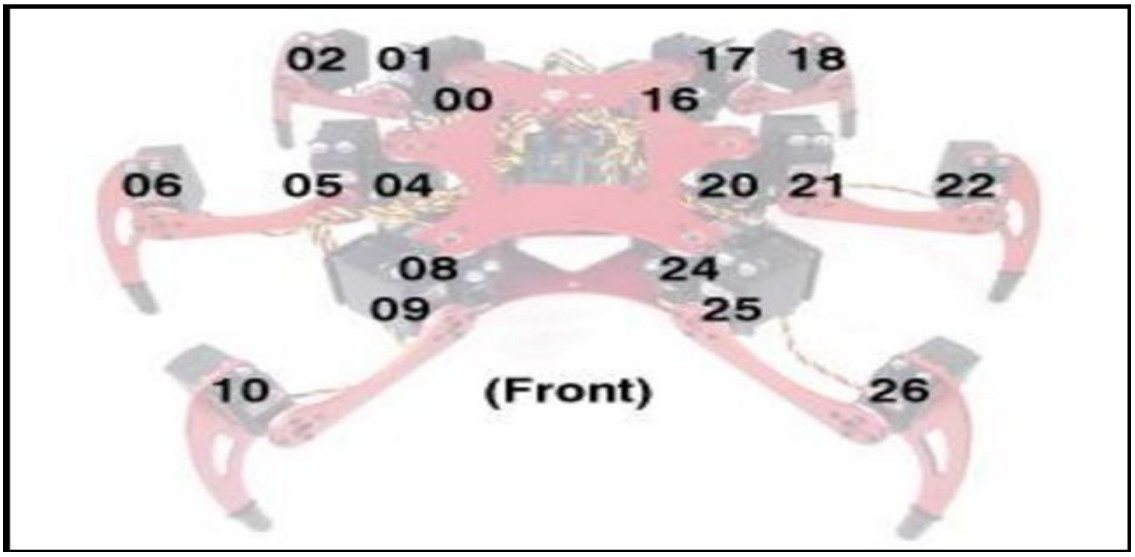
Pinne konfigurasjon	Komponent
Pin 2 / GND	Led1
Pin 3 / GND	Led2
Pin 5 / GND	Buzz-er

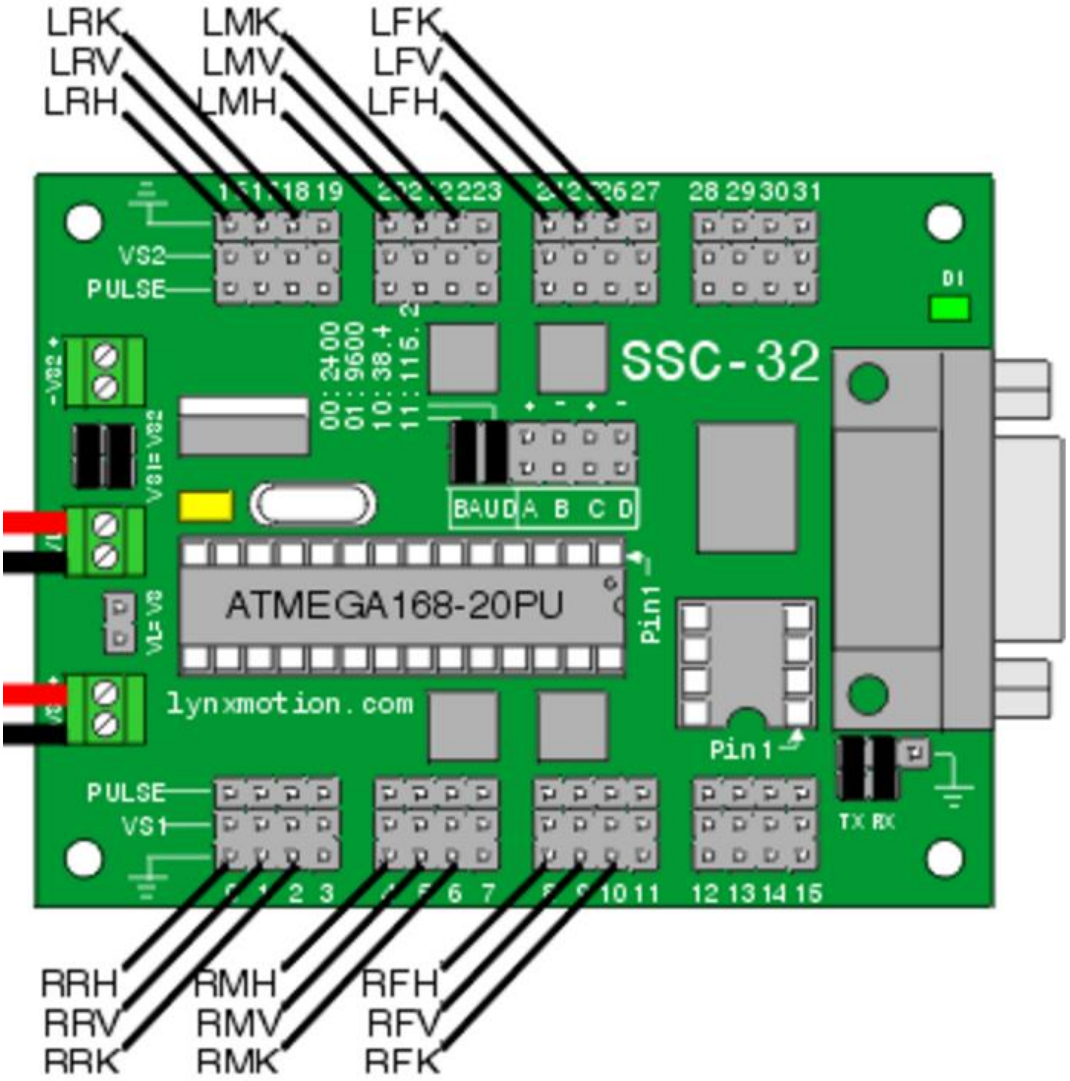


Figur 4 - led og Buzz-er

3.5 Kobling av servo til SCC32U

SSC32U er en servo kontroller som mottar og utfører kommandoer sendt til den fra Arduino Mega. Ved hjelp av en servo kontroller frigjøres minne fra Arduino som ellers ville brukt til å oppdatere servo posisjonene.





Figur 5 - Servo pinnekonfigurasjon

Tabell 5 - SCC32U VS1

Leg	Servo	PULSE pinne	VS1	GND
Venstre bak	Innerst	0	+	-
Venstre bak	Midten	1	+	-
Venstre bak	Ytters	2	+	-
Venstre midte	Innerst	4	+	-
Venstre midte	Midten	5	+	-
Venstre midte	Ytters	6	+	-
Venstre foran	Innerst	8	+	-
Venstre foran	Midten	9	+	-
Venstre foran	Ytters	10	+	-

Tabell 6 - SCC32 VS2

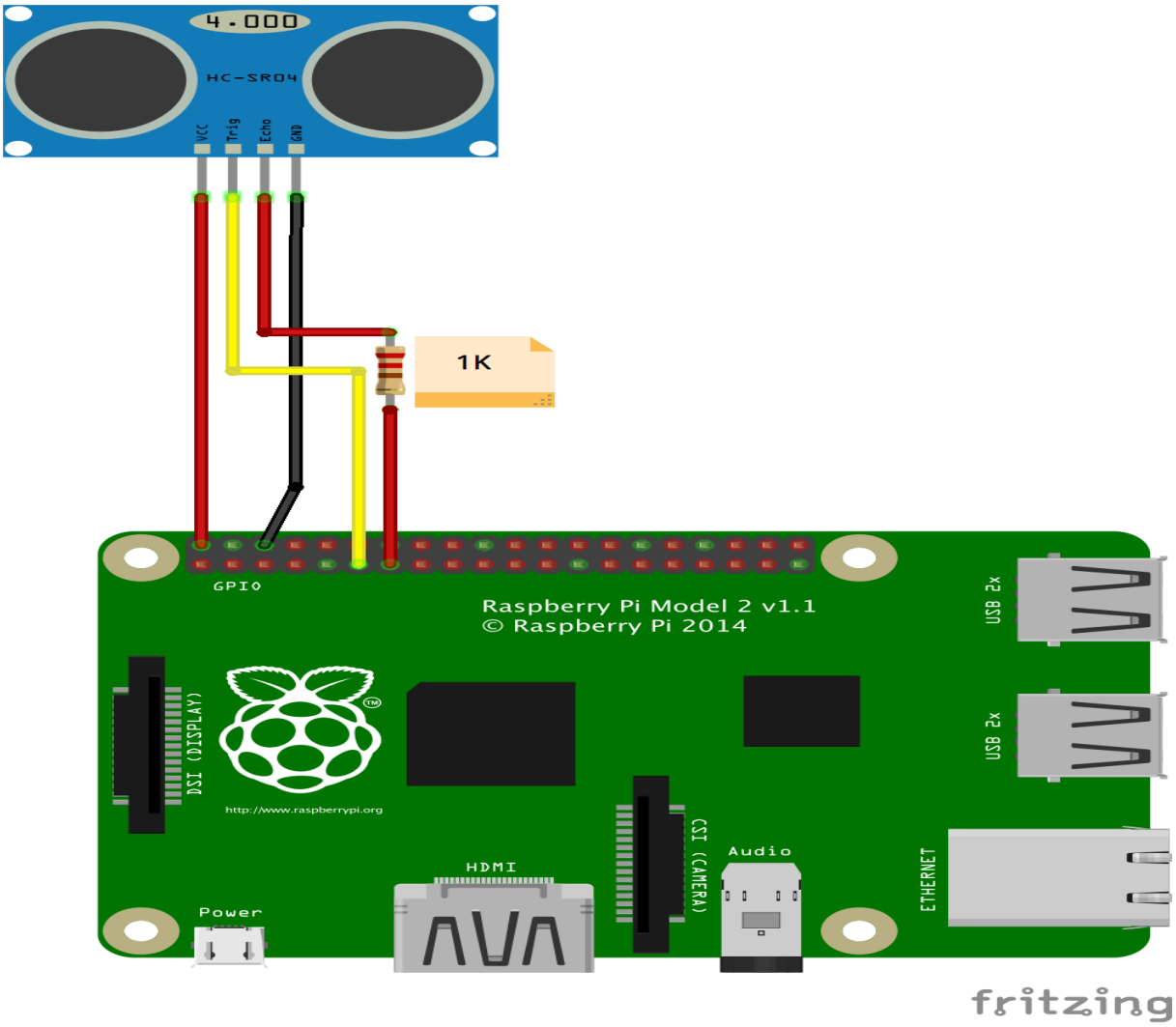
Leg	Servo	Pulse pinne	VS2	GND
Høyre bak	Innerst	16	+	-
Høyre bak	Midten	17	+	-
Høyre bak	Ytters	18	+	-
Høyre midten	Innerst	20	+	-
Høyre midten	Midten	21	+	-
Høyre midten	Ytters	20	+	-
Høyre foran	Innerst	24	+	-
Høyre foran	Midten	25	+	-
Høyre foran	Ytters	26	+	-

3.6 Ultrasonic Rpi2

Vi har koblet til ultrasonic til Rpi 2 som leser avstand, dette for at roboten skal manøvrere unna gjenstander som kommer i veien for den når roboten styres fra AIM. HC-SR04 sensor trenger 5V for strøm, trigger signal kan kobles direkte til GPIO pinnen. Echo pinnen krever at vi bruker 1k Ohm motstand som egentlig bruker 5V logikk som kan skade Raspberry pi. Ved å bruke en motstand senker vi spenningen og beskytter Raspberry pi. Vår siste pinne er GND ellers kjent som Ground, denne er koblet til Raspberry Pi Ground.

Tabell 7 - Ultrasonic & Rpi2

Ultrasonic	Raspberry pi 2
Trig	GPiO 17
Echo	GPIO 18
VCC	5V
GND	GND



Figur 6- Ultrasonic

3.7 Rpi2 kameramodul

Det er implementert live feed kamera for å følge med roboten når den styres trådløs via AIM. Kameramodulen må kobles til CSI-porten, som ligger mellom HDMI og Audio i Raspberry pi.

Figur 7 - Rpi2 kameramodul

3.8 Rpi2 wifi

For at systemet skulle kommunisere trådløs med hverandre har vi brukt wifi adapter som plugges i USB inngangene til Rpi 2. Wi-Pi-modulen støtter 802.11n standard med data rate opp til 150Mbps, har også en høyere trådløs LAN båndbredde, noe som gjør dataoverføring mer effektiv.

Spesifikasjoner:

Tabell 8 - Wipi

Overføringshastighet	11b - 1/2 / 5,5 / 11Mbps, 11g - 6/9 / 12/18/24/36/48 / 54Mbps, 11n - opp til 150Mbps
Frekvensområde	2,4 GHz til 2.4835GHz
Arbeidskanal	1 til 13
Overføre kraft	20dBm (max)
Sikkerhetsfunksjoner	WPA-PSK / WPA2-PSK, WPA / WPA2, 64/128/152 bit WEP-kryptering
Driftstemperatur	0 ° C til 40 ° C

Figur 8 - Rpi wipi

3.9 Auto run Rpi2

Kommunikasjonsprogrammet skal startes opp i bakgrunnen automatisk, uten debug interface.
Om man vil se debug interface må man stoppe den andre scriptet, eller nekte den autostart og så restarte. Dette kan gjøres ved å rename «run.sh» til for eksempel «_run.sh» og så restarte RP2, slik at den ikke finner «run.sh» og ikke kan autostarte. Hvis du starter «cmd_RP2/RP2.py» og får feilmelding som sier noe sånt som «98 address already in use», betyr det mest sannsynlig at scriptet allerede kjører.

Hvis den får en feilmelding som sier noe om «'dev/ttyUSB0'» betyr det sannsynligvis at den har byttet USB-port på den seriell koblingen med Arduinoen. Kan fikses ved å dra USBen ut og putte den inn igjen en eller et par ganger, evt. Å endre i koden (Desktop/RunOnStart/cmd_RP2/RP2.py, Setup: RP2 -> Arduino) fra «dev/ttyUSB0» til «dev/ttyUSB1» eller «dev/ttyUSB2», osv. For å finne hvilken port som er koblet til Rpi bruker kan du også åpne terminal vinduet i Rpi og skrive ls /dev/tyy* (Merk: ikke tallet 1 men bokstaven l), da kommer det oversikt over forskjellige porter som er brukt.

3.10 VNC (Virtual Network Computing)

Noen ganger er det ikke praktisk å arbeide direkte på Raspberry Pi. Når roboten er ute kjører er det greit å ha tilgang til Rpi fra en annen datamaskin via fjernkontroll. VNC er et grafisk skriveborddeling program som lar deg fjernstyre Raspberry pi sitt grensesnitt fra en annen ekstern datamaskin, som kan også være hvor som helst i verden over internett. Når tilgangen er opprettet er det mulig å ha tilgang til koden i Raspberry pi (Python) og kan også åpne arduino IDE i Raspberry pi fra ekstern datamaskin for å feilsøking eller endre Arduino koden til roboten, dette lær seg gjøre så lenge Rpi og Arduino er koblet sammen via USB seriell.

Vær oppmerksom på at grunnleggende VNC er ikke sikkert. Det er ikke kryptert med mindre en setter avanserte innstillinger. Lag et passord som er sikker, men selv da er det noen ganger mulig å ta over tilkoblingen. Denne veiledningen viser hvordan du kan vise og kontrollere Raspberry Pi skrivebordet fra skrivebordet på datamaskinen ved hjelp av spesiell programvare.

https://www.raspberrypi.org/documentation/remote-access/vnc/windows.md

4 SPESIELLE FUNKSJONER

4.1 Overføring hastighet

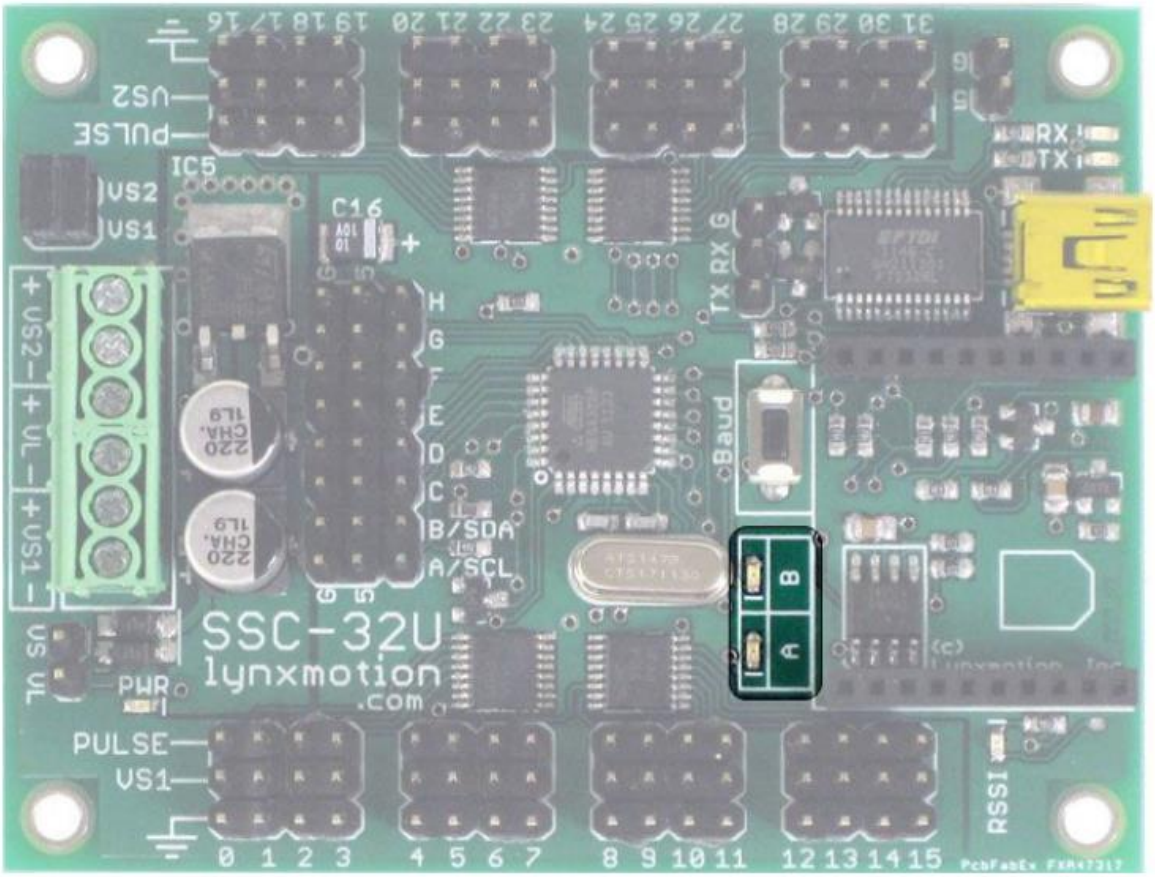
Det er to forskjellige lysdioder på SCC32U korte merket med A og B som forteller hvilken Baud rate som er på,
LED indikerer følgende:

- Både A og B er på før en byte er mottatt.
- Grønn blinker når en gyldig byte er mottatt.
- Røde blinker når en ugyldig byte er mottatt.

Slik stilles Baud rate på SCC32U korte:
Trykk og hold knappen. Ved første lampene lyser for å indikere gjeldende Baud rate. Etter 2 sekunder vil lysdiodene begynne å veksle, det indikerer at du kan endre Baud rate.
Slipp knappen. Trykk på knappen for å bla gjennom overføringshastighet ene, når du har valgt overføringshastigheten du vil etter 5 sekunder vil lysdiodene gå tilbake til normal modus og den nye overføringshastigheten vil bli skrevet til EEPROM. Vi bruker 38400, dette kan endres fra koden.

Tabell 9 - Baud rate

Baudrate	Led
9600	Grønn
38400	Rød
115200	Grønn og rød



Figur 9 - Baud rate

4.2 Strøm tilkobling til SCC32U

4.2.1 VL terminal

VL tillater uregulerte inngangssignaler til logikk spenningen. Logikk spenning velges automatisk mellom VL og VS1. Så lenge VS1 er over 5.3V trenger vi ikke å koble noe til VL terminaler.

Tabell 10 - VL terminal

Ideal	Ingenting tilkoblet
Nominell	6v-12v
Absolutt	12v-16v

4.2.2 VS1 terminal

VS1 terminalen kobles direkte til strøm og GND linjer med servo pins fra 0 til 15. Spenningen som påtrykkes på VS1 bør ideelt sett tilsvare den nominelle spenning av servoer. For de fleste servoer, er dette på 6V. Noen servoer kan operere med 7.4V eller 11.1V LiPo batteri, dette er angitt i servo spesifikasjoner. Ved bruk av 7.4V LiPo batteri er en normal servo motløs og bruke 11.1V med en vanlig servo vil sannsynligvis ødelegge den. Servoer vi bruker tillater kun 6V .

Tabell 11 - VS1 terminal

Nominell	6V
Absolutt	0V-16V

4.2.3 VS2 Terminal

VS2 terminalen er koblet direkte til strøm og GND linjer med servo pinner fra 16 til 31.

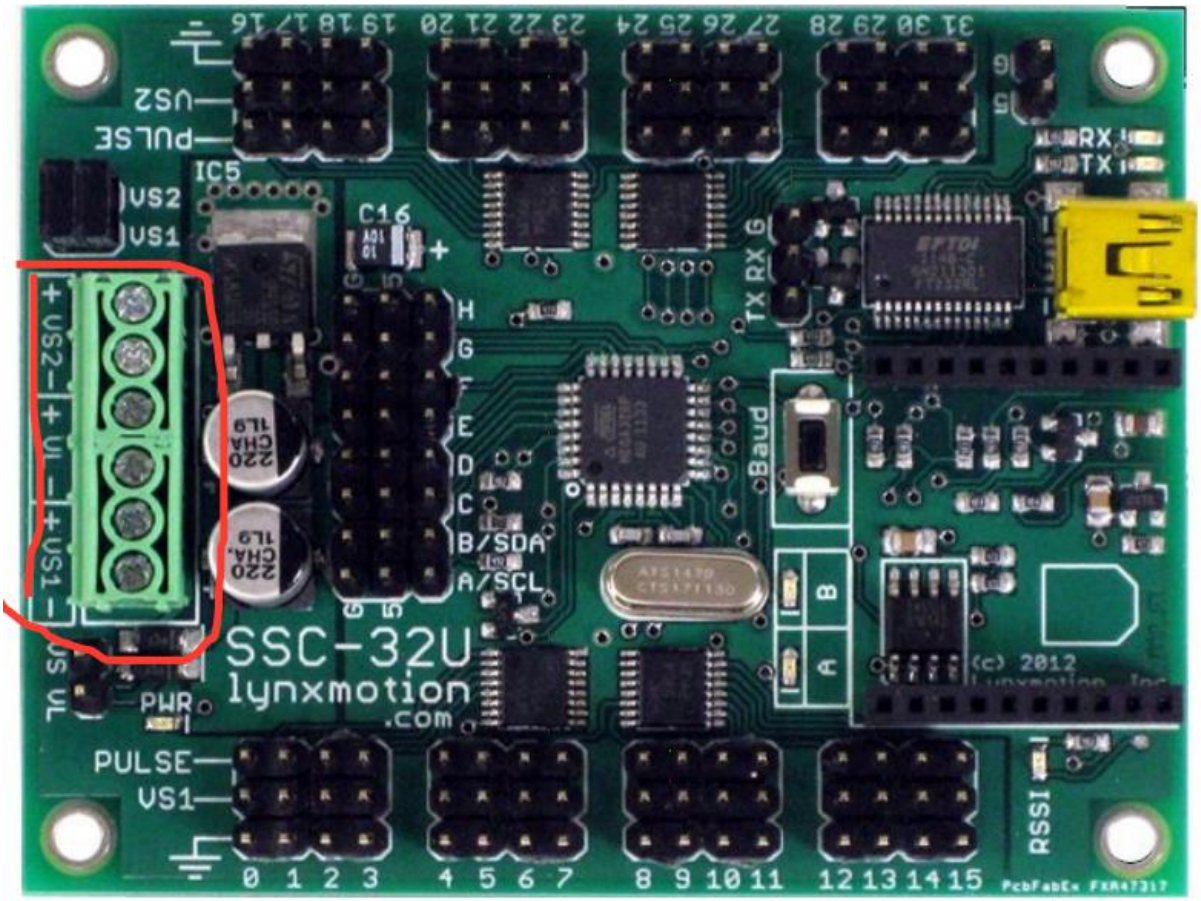
Tabell 12 - VS2 terminal

Nominell	6V
Absolutt	0V-16V

4.2.4 VS1 = VS2 Jumper

Den andre halvdelen av servo pinnene som er nummererte fra16 til 31 kan drives ved hjelp av den samme strømkilden som VS1 ved å sette jumpere på VS1 = VS . Dersom en ønsker å drive linje med servoer som er koblet til VS2 separat fra VS1, kan disse to jumperne fjernes (se figur 8).

Vs1=Vs2
Jumper



Figur 10 - SCC32U Power

4.3 Batterier

4.3.1 6.0 vdc Ni-MH 1600mAh til SCC32U

Batteri pakken vi bruker er lagd av fem høykapasitets AA-celler og 18 multi-leder. Med en slik batteri pakke operer roboten maks 30 min for den er utladet.

Spesifikasjoner:

Tabell 13 - NI-HM

Full ladespenning	7.25V
Nominell spenning	6V
Nominell strøm	1600mAh
Max utladestrøm	16A
Ladetid	1600mAh/1.2 timer

Figur 11 - NI-MH batteri

4.3.2 Powerbank til Rpi2

Powerbanken vi bruker gir strøm til Rpi 2, som igjen gir strøm til Arduino. Vi kobler 2 stk sammen den ene powerbanken lader mens den andre driver Rpi 2.

Spesifikasjoner:

Tabell 14 - Powerbank

Nominell spenning	5V
Nominell strøm	2600mAh
Kobling	USB

Figur 12 - Powerbank

4.3.4 Alternativ strømkilde til servomotorer (LiPo batteri)

LiPo, eller Lithium-ion polymer som det egentlig står for, er en" gammel" batteriteknologi, men forholdsvis ny innen RC. Denne" nye" batteritypen har en del andre karakteristika som gjør den bedre egnet innen RC enn de gamle nikkelbatteriene. Disse kommer her:

Fordeler:

- Veldig liten indre resistans, noe som gjør at batterier kan levere høy utgangseffekt, samt at spenningen under belastning ikke synker drastisk
- Leverer tilsvarende samme utgangseffekt fra batteriet er fulladet til det er tomt (i motsetning til nikkelbatteriene som er merkbart dabber av etterhvert)
- Veldig høy energi kapasitet (noen som fører til lav vekt og fysisk størrelse kontra nikkelbatterier
-
- Tilnærmet ingen utladning
- Har ingen minneeffekt, tar dermed ingen skadde av kladdeladning

Ulemper:

- Må behandles riktig for ikke å ta skade (dette gjelder også så vidt for nikkelbatterier også. Men med lipo er behandling av batterier enda viktigere)
- Farligere, kan lettere ta fyr og/eller eksplodere ved skade (som kan komme av feil behandling)
- For langtidslagring (uker eller mer), må batteriene lades opptil 40 prosent og passe på den går aldri tom når den brukes, helst ikke under 20 prosent det kan skade batteriene.

Siden servomotorene vi bruker ikke takler mer enn 6V, kan vi ikke kjøre liPo batteriene direkte. Vi har brukt spenningsregulator (LM2596S) for å senke spenningen på liPo batteriene fra 11.1V til 6V, men vi har funnet ut at regulatoren gir ikke mer 2A, noe som ikke anvendes til servo korte. Når servomotorene kjører for fullt bruker den 450mA pr motor, vi har 18 av disse motorene, som utgjør 8A til sammen. Vi har brukt ny regulator (Turnigy 8-15 UBEC) som vil levere en konstant strøm på 8A eller mer med korte støt på opptil 15A, som har varsling når det er tid for ladning for å beskytte batteriene.

Spesifikasjoner:

Tabell 15 - UBEC spesifikasjoner

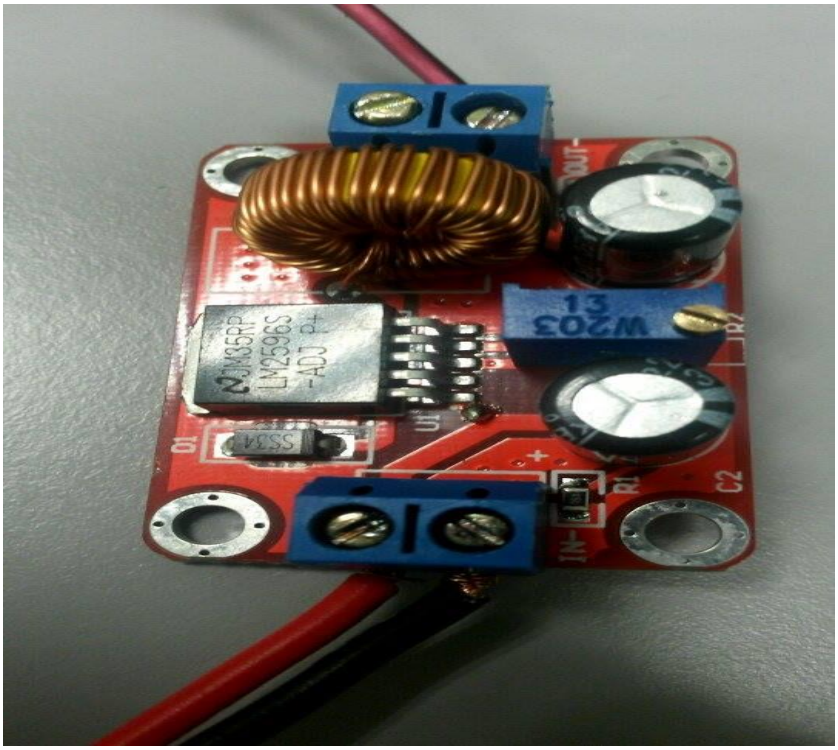
Utgang konstant	5V eller 6V	8A
Inngang	6V-12.6V	2 eller 3 celler

4.3.5 Alternativ strømkilde tils Rpi (Lipo batteri)

Siden Raspberry pi takler 5V fra USB, har vi bestemt oss for å kjøre ekstra lipo batteri med regulator til Rpi. Grunnen til vi gjør dette er for at Raspberry pi gir strøm til Arduino og tømmer ut powerbanken ganske fort. Vi har også funnet ut at Systemet stopper å fungerer hvis det er koblet servomotor direkte til Arduino, en trenger ekstra batterikilde til Arduino om en ønsker å koblet til servomotorer eller komponenter som drar mye strøm.

4.3.6 Arduino Mega minijack inngang spenning

Arduino Mega bruker minijack som inngang spenning fra batterikilde, for å bruke dette må en bruke inngangsspenning mellom 9V-12V ellers fungerer ikke Arduino Mega som den skal. Fra USB inngangen er det vanlig 5V.



Figur 13 - lm2596S

Figur 14 - UBEC

4.4 PS2 funksjoner

Tabell 16 referer til funksjonene til hexapod fra PS2 kontrolleren, det er endring fra originalen. Om det skulle være ønskelig å bruke kryss knappen på PS2 kontrolleren ligger alt klart i gjort koden, en trenger kun å lage ny funksjon eller legge til kryss knappen, kryss kan også brukes til å lage egne bevegelses sekvenser fra SCC-32 Servo Sequencer Utility. Tabellen viser også verdier som desimal, hexadesimal og Ascii som er brukt på Rpi og AIM for å kommunisere med roboten.

Dec	ASCII	HEX	Button	Function description
65	A	41	L3 ↑	Move forward
66	B	42	L3 ↓	Move backward
67	C	43	L3 ←	Move left
68	D	44	L3 →	Move right
69	E	45	R3 ↑	Push up
70	F	46	R3 ↓	Push down
71	G	47	R3 ←	Rotate left
72	H	48	R3 →	Rotate right
73	I	49	Start	Start servos
74	J	4A	L1	Translate mode (dance)
75	K	4B	L2	Rotate mode (twist)
76	L	4C	O	Single leg mode
77	M	4D	X	GP-Player
78	N	4E	□	Balance mode
79	O	4F	△	Stand up / sit down
80	P	50	DPad ↑	+ 10mm height
81	Q	51	DPad ↓	- 10mm height
82	R	52	Select	Select modes
83	S	53	R1	Double lift height mode
84	T	54	R2	Double travel length mode (removed)
85	U	55	R3	Walkspeed mode
86	V	56	Dpad →	Walk speed up
87	W	57	Dpad ←	Walk speed down
88	X	58		
89	Y	59		
90	Z	5A		

Figur 15 - PS2 Funksjoner

4.5 Kalibrering av robotens leg

Servomotorer kommer alltid med lite avvik fra produksjon, dette oppdaget vi når roboten ikke sto i water og beina ikke traff bakken, servo motorene hadde forskjellige midtstilling posisjon som måtte kalibreres med spesiell designet programvare. Programvaren vi brukte heter SCC-32 Servo Sequencer Utility som er spesiell laget for SCC32U korte. Med programvaren kan du enkelt flytte servomotorer, kalibrere sine posisjoner, lagre og spille av bevegelsessekvenser, oppgradere SSC32U programvaren og mer. Merk: Når roboten skal kalibreres må baud rate på SCC32U korte være på 9600, ellers for man ikke kontakt med servokorte.

Trykk på auto så vil den finne USB porten som er koblet til servokorte. Når en kalibrerer beina kan en trykk calibrate knappen, når beina er kalibrert trykk igjen på calibrate knappen. Dette vil lagre kalibreringen.

Programvaren kan lastes ned fra linken under:

http://www.lynxmotion.com/p-895-free-download-ssc-32-servo-sequencer-utility-created-using-flowbotics-studio.aspx

Figur 16 - SCC-32 servo sequencer

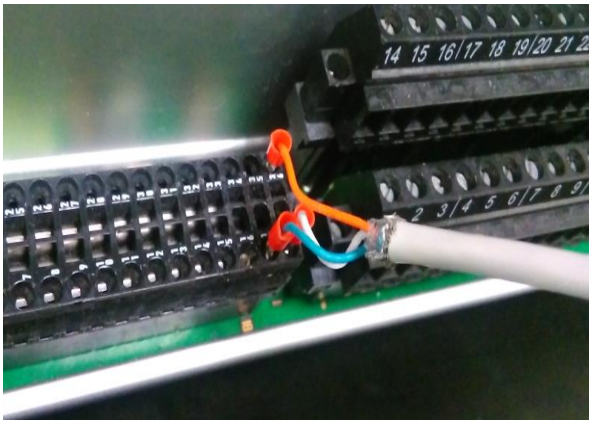
5 RCU til RPI1

5.1 OPPKOBLING AV SYSTEMET

Systemet vil være pakket i 2 kofferter, hvor den ene kofferten vil inneholde den stasjonære delen av systemet, dette vil da være RCU, RPi1, strømtilførsel og tilkoblingsmuligheter til stikkontakt(230v). I koffert nummer 2 så vil den mobile delen være å finne. Det vil være hexapoden med div tilkoblede ekstra utstyr, deriblant RPi2.

5.1.1 Oppkobling av RCU og RPI 1

Strømkobling til RCU
Oppkoblingen av RCU foregår ved først koble til strømtilførselen, dette gjøres ved å ta en ledning fra positiv utgang på forsyning til den positive (+) inngangen på RCU, ved klemmen med porter 14-25. Så ta en jordledning fra negativ utgang på forsyninga, til den negative (-) inngang på RCU, ved klemmen med porter 1-12.
Modbus kobling til RPI1
For koblingen mellom RCU og RPi1, så har vi en cat7 kabel (Hvor endene er fjernet) som er koblet slik



Figur 17 - Kobling RCU 510

Hvor Orange ledning er koblet til port 36, blå ledning til 17 og hvit ledning til port 18.
På den andre siden av cat7 ledningen, så er det koblet til et USB adapter. Her kreves det også at en installerer USB driver på den virtuelle maskinen som styrer AIM. I koden vil det nok stå at USB0 er det valgte tilkoblingen, men dette kan det hende man vil endre på, noe som kan gjøres i RPi1 skriptet. Hvis det er problemer med at Python skript krasjer på grunn av feil porter, så kan dette ordnes ved å endre i Python hvilke port en skal bruke eller dra USB ut og inn et par ganger.

Figur 18 - Modbus

Trådløs kobling mellom RPi1 og RPi2
Når det kommer til RPi1-2 så er det først og fremst at en må koble til strømforsyningen, på RPi1 er det et vanlig strømadapter med micro- USB, som blir koblet i skjøteledningen som følger med i kofferten. På RPi2 så blir den koblet med en USB-microUSB fra en powerbank (eller en annen batteri løsning som er valgt)
På RPiene er det også mulig å koble til skjerm, tastatur/mus igjennom USB portene, dette er valgfritt. Kan være greit for å se om alt fungerer som det skal.
For å koble til den trådløse kommunikasjonen så er det 2 Wifi USB adapterer, som blir plassert i en av USB portene, på begge RPiene. Routern som følger med, og som en setter opp et lokalt nettverk i, er den som gir ut IP adresser til RPiene. For å finne disse adressene så åpner man LXterminal på RPi og skriver «ifconfig» og henter adressene fra informasjonene en får opp da.

5.1.1 Operasjon av RPi1

I utgangspunktet så skal det ikke være nødvendig å gjøre så mange endringer i RPI1 når systemet er i bruk. Det kan være nødvending å manuelt sette i gang skriptet som skal brukes, varierende på om den løsningen er aktiv eller ikke.

5.1.2 Operasjon av RPi2

RP2 kommunikasjonsprogram
Kommunikasjonsprogrammet skal startes opp i bakgrunnen automatisk, uten debug interface.
Om man vil se debug interface må man stoppe det andre scriptet, eller nekte den auto-start og så restarte. Dette kan gjøres ved å gi et nytt navn til filen «run.sh» til for eksempel «_run.sh» og så restarte RP2, slik at den ikke finner «run.sh» og ikke kan auto-starte. Hvis du starter «cmd_RP2/RP2.py» og får feilmelding som sier noe sånt som « 98 address already in use », betyr det mest sannsynlig at scriptet allerede kjører.
Hvis den får en feilmelding som sier noe om «'dev/ttyUSB0'» betyr det san nsynligvis at den har byttet USB-port på den seriell koblingen med Arduinoen. Kan fikses ved å dra USBen ut og putte den inn igjen en eller et par ganger, evt. Å endre i koden (Desktop/RunOnStart/cmd_RP2/RP2.py, Setup: RP2 -> Arduino) fra «dev/ttyUSB0» til «dev/ttyUSB1» eller «dev/ttyUSB2». Kan også finne hvilke porter som er koblet til Raspberry pi ved å åpne terminal fra Rpi og taste ls /dev/tyy* for å se hvilke porter som er aktive.

Kamera-feed

RP2: Desktop/RunOnStart/ Info.txt og designdokument om RP2 har mer om dette.
Streamen skal auto-startes på RP2. Hvis den ikke funker, eller den skal startes manuelt, kan kommandoen som ligger i Info.txt og run.sh kjøres i terminalen.
Koble til og se på live feed:

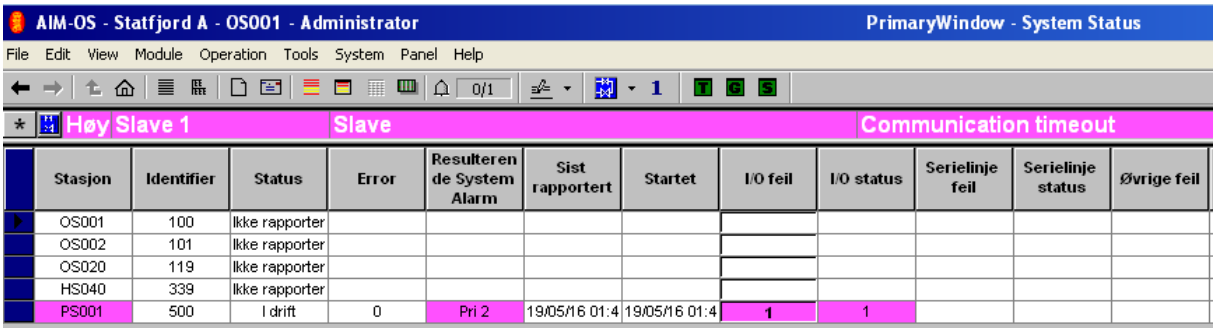
1. Åpne VLC Player
2. Åpne network stream (ctrl+n)
3. Koble til «rtsp://192.168.1.123:8554» (hvis IP og port ikke endres)

Den har en latency på ca. 3-4 sekunder. Innstillinger om hvordan streamen funker finnes i RP2:Desktop/RunOnStart/run.sh.

6 AIM

RCU510 kobles sammen med Operatørstasjon og Modbus RT232. Modbus utgangen kobles videre med Raspberry Pi.

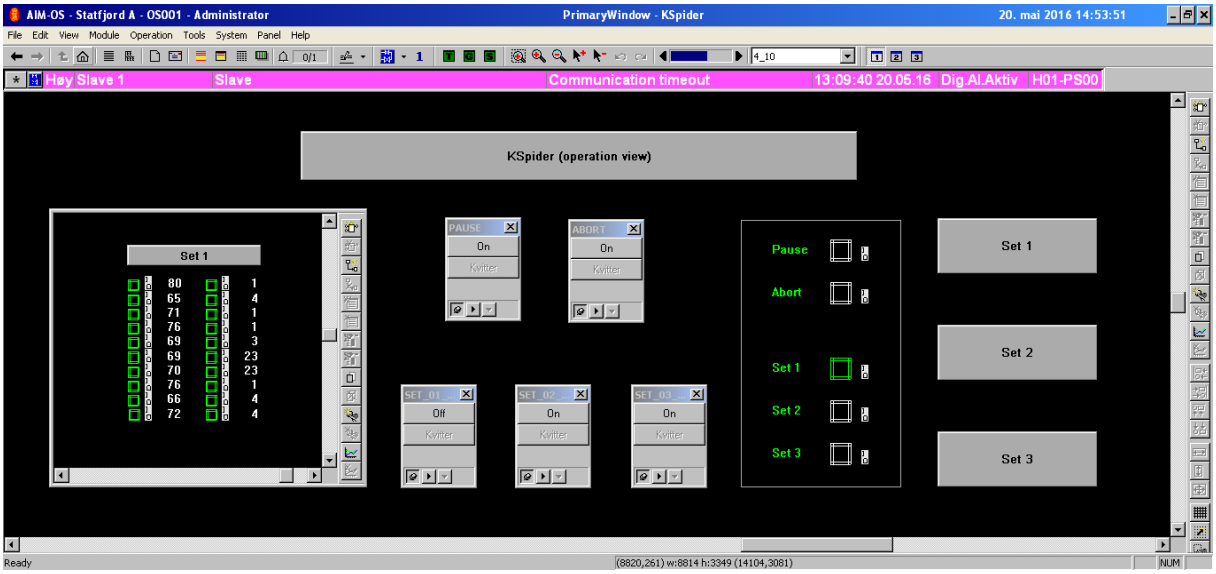
På operatørstasjonen kjører operatøren Loopsim – VMware Player. Når programmet er klar til bruk kobles RCU510 til en strømkilde. Operatøren kontrollerer statusen i systemstatus.



Stasjon	Identifier	Status	Error	Resulteren de System Alarm	Sist rapportert	Startet	I/O feil	I/O status	Serielinje feil	Serielinje status	Øvrige feil
OS001	100	Ikke rapporter									
OS002	101	Ikke rapporter									
OS020	119	Ikke rapporter									
HS040	339	Ikke rapporter									
PS001	500	I drift	0	Pri 2	19/05/16 01:4	19/05/16 01:4	1	1			

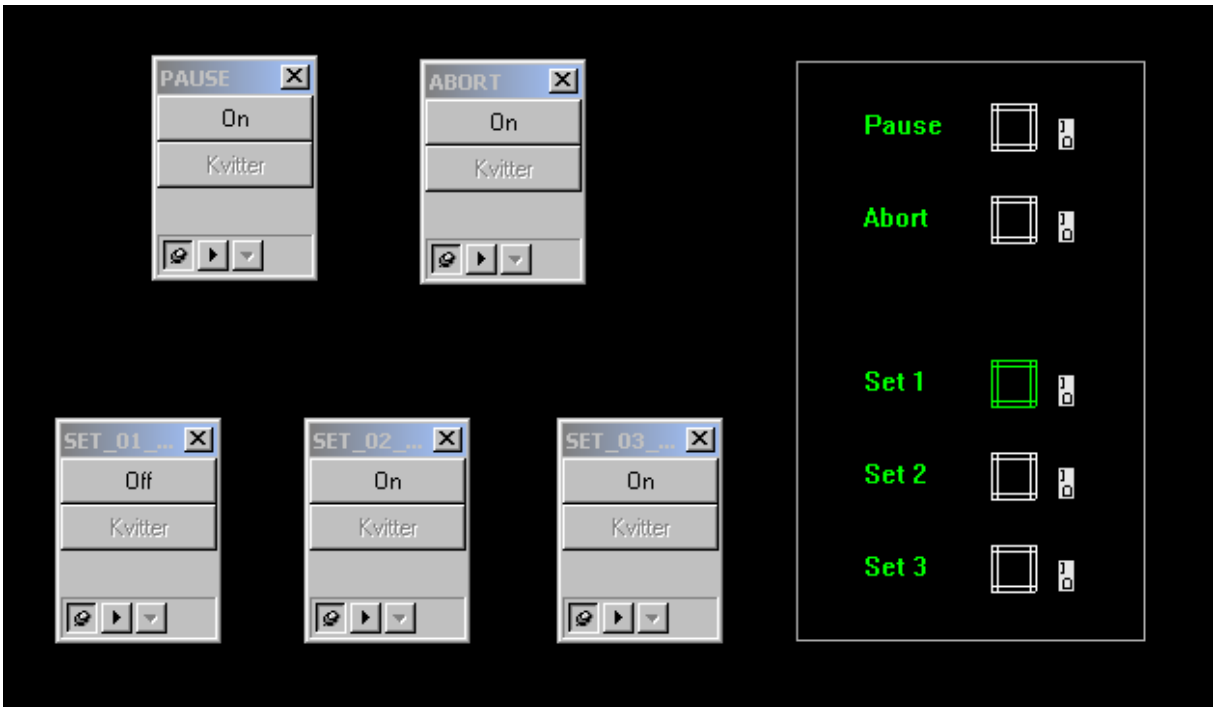
Figur 19 - Systemstatus i OS

Ved hjelp av navigatorfunksjonen finner operatøren Flow Sheet KSpider. Systemet må settes i PS Configuration Mode.



Figur 20- KSpider (operasjonsvinduet)

Det hat blitt laget 3 forskjellige kommandopakker med 10 kommandoer og tidslengde i hver pakke som operatøren kan velge mellom for å sende de til Hexapoden. De tre kommandopakkene er ferdig laget i tre forskjellige Flow Image, men operatøren skal bruke HotSpot som inneholder brytere Set 1, Set 2, Set 3, Pause og Abort.



Figur 21 - HotSpot med brytere Set1, Set2, Set3, Pause og Abort

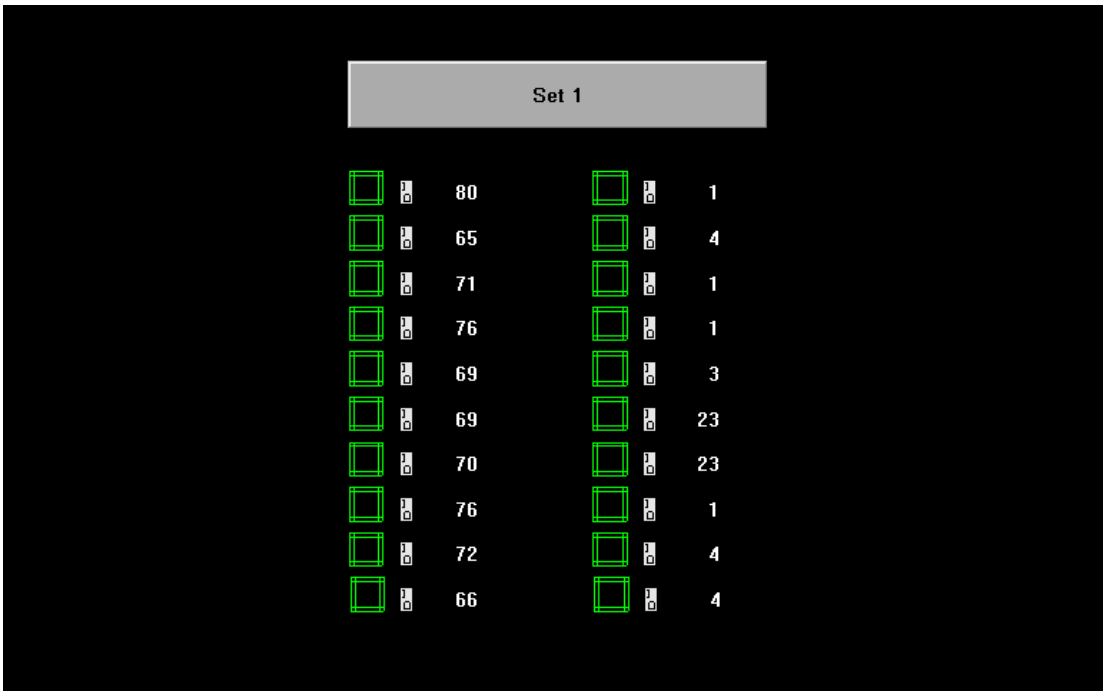
Operatøren kan feste de operasjonsbryterne inn i operasjonsvinduet for å forenkle styring. På Figur 20 sender operatøren Set 1 (bryteren for Set 1 er høy og modulsymbolet lyser grønt).

Operatøren velger ett sett (en pakke med kommandoer) som skal sendes til Hexapoden ved å sette det ønskelige settet høy. Mens Hexapoden utfører kommandoer i settet kan operatøren stoppe den kjørende kommandoen med å sette Pause-bryteren høy (modulsymbolet til pause skal lyse grønt). Hexapoden skal stå i ro i ventemodusen. For å gå ut fra pausemodusen må operatøren sette Pause-bryteren i lavt. Hexapoden skal fortsette med den neste kommandoen i settet.

Dersom operatøren ønsker å bytte settet eller stoppe kjørende settet fullstendig setter operatøren Abort-bryteren høy. Da kan operatøren velge et annet sett med å sette det ønskelige settet høy og gå ut fra stopp-modusen med sette Abort-bryteren lav. Hexapoden skal fortsette med det nye settet. Dersom operatører ønsker å kjøre det samme settet rett etter Abort-kommando må det utføres forandring i det settet. Det kan være forandring i en/flere kommandoer og/eller tidslengde. Det samme uforandret settet skal ikke kjøres rett etter Abort-kommandoen.

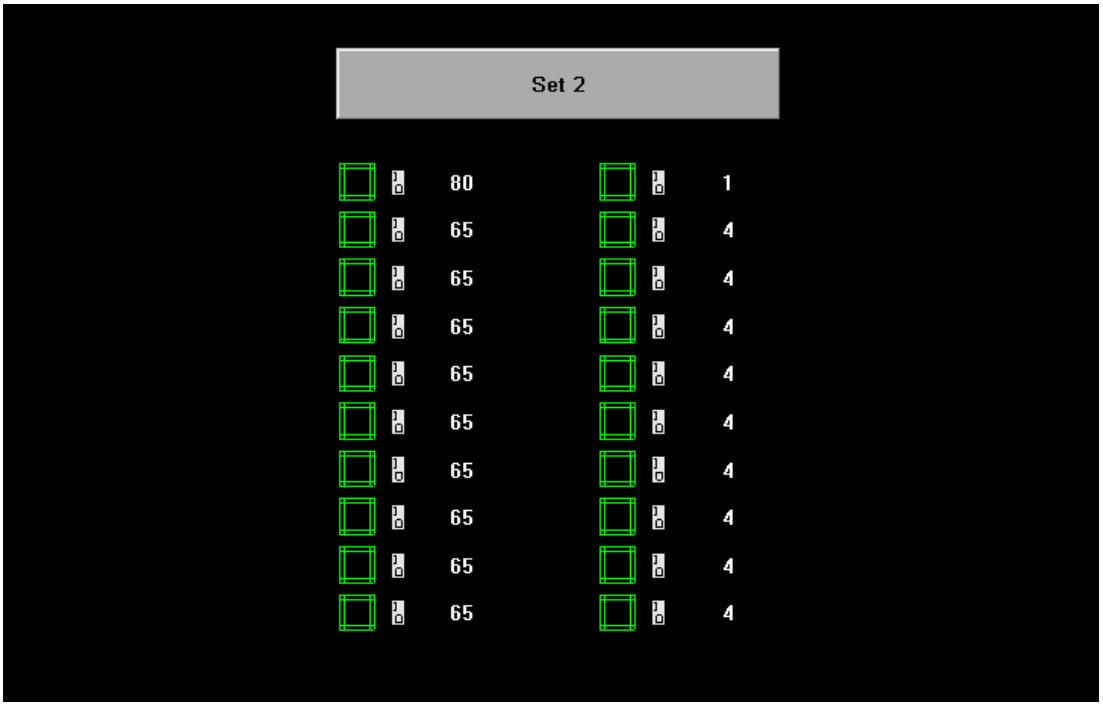
Dersom operatøren utfører endringer i kommandoer og/eller tidslengde og ønsker å beholde de endringene må operatøren lagre utførte forandringene og gjøre Backup i prosesstasjonen (dette er beskrevet i Designdokumentet 3.0).

De tre ferdig laget settene inneholder tre forskjellige kommandopakker for:
Kommandopakke Set 1 lar Hexapoden presentere seg, dvs. Hexapoden skal løfte kroppen opp, gå fremover i 4 sekunder, snu seg i 90 grader til venstre og vinke med et bein, snu seg bak og gå tilbake. Alle kommandoene er definert i figur 15.



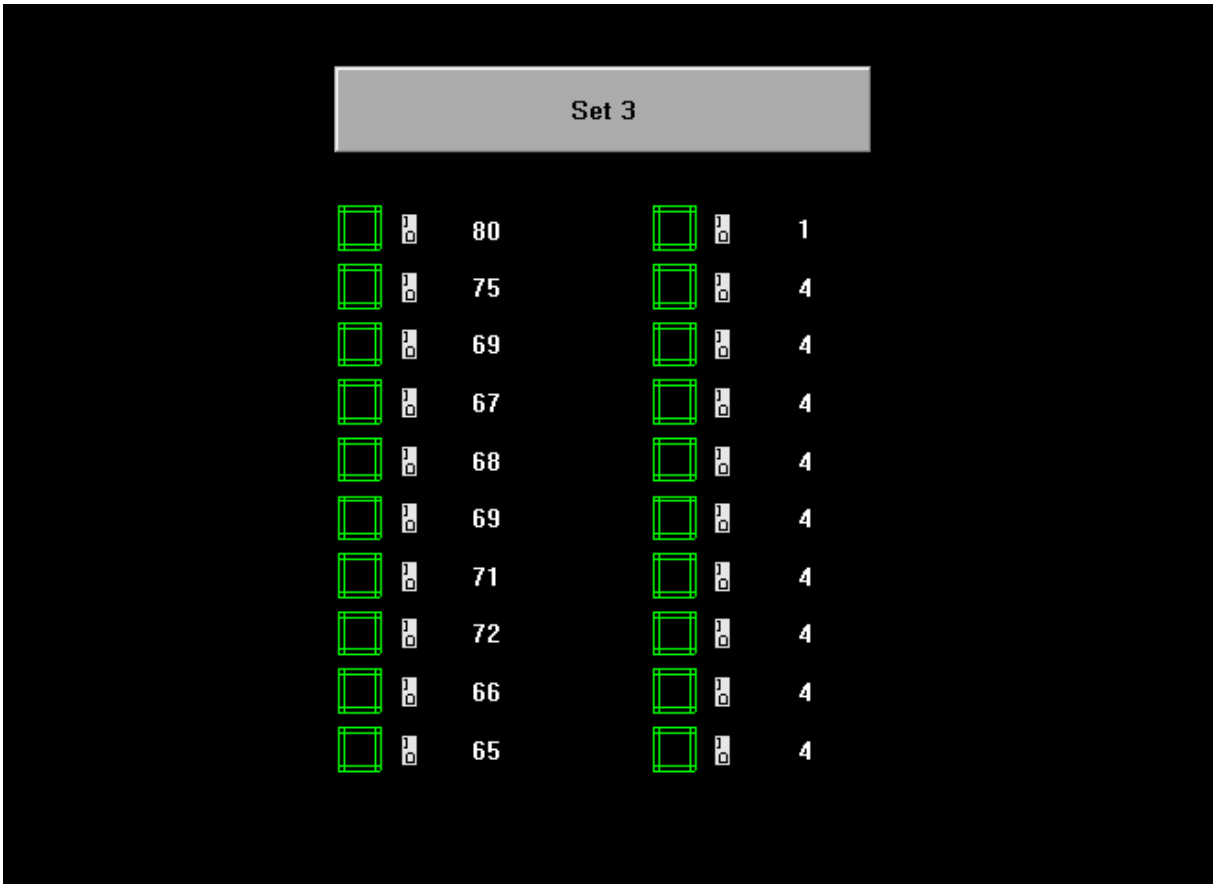
Figur 22 - Kommandopakke Set 1

Kommandopakke Set 2 inneholder kun bevegelsen fremover med korte tidslengder. Med Set 2 kjører Hexapoden fremover i 36 sekunder totalt. Dette er en enkel kommandopakke hvor operatøren kan observere funksjonaliteten til nærhetssensoren (Ultrasonic). Ved hjelp av sensoren registrerer Hexapoden forstyrrelser og unngår kollisjon ved å snu seg til siden og fortsetter å gå videre.



Figur 23 - Kommandopakke Set 2

Kommandopakke Set 3 inneholder kombinasjon av kommandoer for mer avanserte bevegelser av Hexapoden. Det skal sendes en pakke med kombinasjon av rotasjonsbevegelser som til sammen skal presentere en dans.



Figur 24 - Kommandopakke 3

7 REFERANSER

[1] http://www.lynxmotion.com/c-15-software.aspx

[2] https://www.raspberrypi.org/documentation/remote-access/vnc/windows.m

[3] http://www.hobbyking.com/hobbyking/store/__6233__turnigy_8_15a_ubec_for_lipoly.html

[4] https://www.element14.com/community/community/raspberry-pi/raspberry-pi-bplus/blog/2014/11/06/holiday-special--santa-trap

[5] https://www.raspberrypi.org/help/camera-module-setup/

[6] http://www.radiostyrt.no/showthread.php/48265-LiPo-Battery-FAQ

HELSE, MILJØ OG SIKKERHET

PROSJEKT	K-Spider		
OPDRAGSGIVER	Kongsberg Maritime AS		
UTFØRT VED	Høgskolen i Sørøst-Norge, avd. Kongsberg		
REVISJON	1.0		
MEDLEMMER	Aleksander Tokle Poverud, Masoud Shah Pasand, Tor Gunnar Finnerud, Hanna Kåsastul, Paul Knutson Sæther, Abdurahman Senkaya		
Dokumenthistorikk	**REVISJON**	**UTGITT**	**BESKRIVELSE**
	1.0	20.05.2016	Første utgave

INNHOLDSFORTEGNELSE

1 DOKUMENTHISTORIE 3

2 INNLEDNING 3

3 SIKKERHETSFUNKSJONER 4

3.1 Produktdesign 4

3.2 Oppbevaring og frakt av utstyr 4

3.3 Nødstopp 4

1 DOKUMENTHISTORIE

Tabell 1 - Dokumenthistorie

VERSJON NR:	DATO ENDRET	ENDRET AV	GODKJENT AV	BESKRIVELSE
0.1	15.05.2016	Abdurahman	Masoud	• Dokumentet ble opprettet
0.2	18.05.2016	Masoud	Abdurahman	• Revidert oppsett
1.0	20.05.2016	Masoud	Abdurahman	• Godkjent og utgitt

2 INNLEDNING

Helse, miljø og sikkerhet (HMS) er et sentralt tema for utførelsen av oppgaven og implementering av sluttproduktet. Dette dokument er opprettet for å sikre bevissthet rundt tema HMS. Prosjektgruppas fokus på HMS er først og fremst å sikre personer i nærheten av produktet. Når dette er i varetatt skal produktet sikres. Dette ønskes oppnådd, ved å hele veien ha fokus på alle disse faktorene som kan påvirke omgivelsene og sikkerheten.

3 SIKKERHETSFUNKSJONER

Det er viktig å ha fokus på å begrense mulighetene for skade på personer og utstyr under utvikling av software og hardware.

3.1 Produktdesign

For å få en så sikker produktdesign som mulig, er det avhengig av at skarpe gjenstander ikke forkommer slik at en kan påføre seg personskader. Derfor er man avhengig av å ha avrundede kanter, slik at man ikke kan påføre noen, kuttskader. Med skarpe kanter kan det i tillegg påføre skader på kabler som er rundt produktet. Dette kan påføre hull i kabelen og man kan ende opp med en defekt kabel, som gjør at produktet ikke fungerer som den skal. For å beskytte kablene ytterligere har prosjektgruppen montert ekstra kabel strømper rundt kablene for å forebygge mot skader på kablene i produktet.

3.2 Oppbevaring og frakt av utstyr

Prosjektgruppen har gått til innkjøp av to kofferter for å beskytte ømfintlig og kostbart utstyr under frakt av produktene. Koffertene er støtdempende, vannavstøtende, støvtett og har skuminnlegg for tilpasning til innholdet, og er praktisk og lett å frakte med seg.

3.3 Nødstopp

Systemet er utstyrt med en nødstopp funksjon i AIM. Ved å aktivere nødstopp, brytes all aktivitetene i roboten. Dette er for å gi brukeren mulighet til å stoppe roboten i farlig situasjoner for mennesker eller for utstyret.

ETTERANALYSE

PROSJEKT	K-Spider
OPDRAGSGIVER	Kongsberg Maritime AS
UTFØRT VED	Høgskolen i Sørøst-Norge, avd. Kongsberg
REVISJON	1.0
MEDLEMMER	Aleksander Tokle Poverud, Masoud Shah Pasand, Tor Gunnar Finnerud, Hanna Kåsastul, Paul Knutson Sæther, Abdurahman Senkaya

Dokumenthistorikk	**REVISJON**	**UTGITT**	**BESKRIVELSE**
	1.0	20.05.2016	Første utgave

INNHOLDSFORTEGNELSE

1 DOKUMENTHISTORIE 3

2 INNLEDNING 3

3 ADMINISTRATIV VURDERING 4
3.1 Planlegging 4
3.2 Timeestimat 4
3.3 Møter 5
3.4 Oppfølgning og rapportering 5
3.5 Presentasjoner 5

4 TEKNISK VURDERING 6
4.1 Konstruksjon 6
4.2 Programmering 6

5 EGENVURDERING 7
5.1 Aleksander Tokle Poverud 7
5.2 Masoud Shah Pasand 9
5.3 Tor Gunnar Finnerud 10
5.4 Hanna Kåsastul 11
5.5 Paul Knutson Sæther 12
5.6 Abdurahman Senkaya 13

6 REFEREANSER 14

TABELLER

Tabell 1 - Dokumenthistorie 3

1 DOKUMENTHISTORIE

Tabell 1 - Dokumenthistorie

VERSJON NR:	DATO ENDRET	ENDRET AV	GODKJENT AV	BESKRIVELSE
0.1	18.05.2016	Masoud	Aleksander	• Dokumentet ble opprettet.
1.0	20.05.2016	Masoud	Aleksander	• Lagt inn egenvurdering fra alle medlemmene • Dokumentet er godkjent og utgitt.

2 INNLEDNING

Dette dokumentet vil fremstille en avsluttende vurdering av prosjektet. I kapitlene som kommer vil vi ta for oss administrative, tekniske og personlige erfaringer gjort i prosjektperioden.

I den administrative delen vil vi si litt om hvordan vi har jobbet med prosjektet, med fokus på planlegging, timeestimat, møter og presentasjoner. Vurdering av det tekniske arbeidet vil være fokusert på bygging av kommunikasjonsnettverket og programmering i AIM2000, C og Python. I siste kapitel vil det være en egenvurdering for samtlige av gruppemedlemmene.

3 ADMINISTRATIV VURDERING

Det har blitt lagt stor vekt på den administrative delen av prosjektet både av høgskolen og studentgruppen. I dette kapitlet vil gi en vurdering på det administrative arbeidet som er gjort i prosjektperioden. Det administrative arbeidet har hjulpet oss til å fullføre prosjektet på en best mulig måte.

3.1 Planlegging

En stor del av den administrative delen har vært å planlegge arbeidet i prosjektet. Vi hadde tidlig dannet prosjektgruppen og opprettet kontakt med Kongsberg Maritime, da vi ønsket et spennende hovedprosjekt. Etter mye research valgte vi prosjektmodellen « Unified Process» som vi har brukt gjennom prosjektperioden. Denne prosjektmodellen har passet oss utmerket og har hjulpet oss med å gjennomføre prosjektet på en god måte. Vi har erfart at det kreves god planlegging for å gjennomføre et slikt omfattende prosjekt.

3.2 Timeestimat

Som en del av oppgaven har gruppen estimert timeforbruk til de forskjellige aktivitetene i prosjektplanen, både administrative- og tekniske [1]. Ingen av gruppemedlemmene hadde erfaring med slik timeestimering og derfor ble det noe avvik rundt timeestimering for de forskjellige aktivitetene. For eksempel, så viste det seg at vi hadde estimert for mange timer på visse aktiviteter, og for få på noen andre. Det skal også nevnes at vi underveis i prosjektet har lagt til nye aktiviteter, som vi ikke forutså i starten. Selv om det ble noe avvik, så ble prosjektgruppen flinkere på timeestimering og det har gitt oss en god læringsprosess.

Fra Høgskolen sin side, så har de estimert ca. 600 timer pr.student for bacheloroppgaven som omfatter 20 studiepoeng. I praksis viste dette seg å bli mye, men ikke uoverkommelig. Prosjektgruppa estimerte å jobbe ca. 28 timer i uken før eksamen og 37.5 timer etter eksamen. Med konkrete arbeidsoppgaver så fungerte dette veldig bra.

3.3 Møter

Gjennom hele prosjektperioden har gruppen hatt ukentlig møter med intern veileder hvor vi har gått gjennom oppfølgningsdokument og andre relevante temaer rundt prosjektet. Prosjektgruppen har også hatt interne gruppemøter der vi har innført 10 minutters «Scrum» møte om morgenen. I tillegg har vi hatt god kontakt med ekstern veileder via telefon og email.

Erfaring med møtene har vært at vi har fått god oversikt over fremdriften i prosjektet og at vi kunne raskt iverksette tiltak ved avvik. Møtene med intern veileder og ekstern veileder har vært verdifulle.

3.4 Oppfølgning og rapportering

Oppfølgningsdokumenter og timelister ble sendt til intern veileder hver uke, 24 timer før med møteinnkalling. Oppfølgningsdokumenter inneholder en beskrivelse av aktivitetene som er gjort uken før, samt estimert tid og planlagt aktiviteter for neste uke. Timelistene inneholder en oversikt over samtlige gruppemedlemmers tidsbruk.

3.5 Presentasjoner

Underveis i prosjektet har vi hatt to presentasjoner. De to første presentasjonene på 20 minutter ble hovedsakelig brukt til å presentere prosjektoppgaven, prosjektgruppen, prosjektmodell, og framdriftsplan for prosjektet. Den tredje og siste presentasjonen er 02.05.2016, etter at denne vurderingen er skrevet. Presentasjonene har vært lærerikt og vi skal jobbe konkret med å lage en god avsluttende presentasjon for en ukjent forsamling.

4 TEKNISK VURDERING

I dette kapitelet vil vi kort presenterer den tekniske delen av oppgaven. Vårt system består av mange delsystemer som skal kommunisere sammen. Det har vært veldig utfordrende å få dette til.

4.1 Konstruksjon

Vi kunne valgt hvilken som helst sensorplattform, men valg av Hexapod gjør at vi kan heise, senke og gjøre mer kompliserte bevegelser enn de fleste andre plattformer. Et av kravene fra Kongsberg Maritime var at den skulle kunne bevege seg i ulendt terreng, så dette passer bra. Vi kjøpte robotdeler fra Robotshop som vi monterte sammen til en hel Hexapod robot.

4.2 Programmering

Som kanskje den største delen av oppgaven var det programmering i Python, C og Kongsberg Maritimes software, AIM2000. Før prosjektet startet hadde ingen av prosjektmedlemmene kjennskap til programmeringsverktøyet til Kongsberg Maritime. Heller ikke noe erfaring i Python. Tidlig i prosjektet fikk tre av prosjektmedlemmene innføring i Kongsberg Maritimes software via vår ekstern veileder. Representanter fra Kongsberg Maritime har vært behjelpelige med spørsmål og utfordringer som har oppstått underveis i prosjektet. Prosjektgruppen har også vært effektive til søke og finne løsninger for hindrer som oppstod under prosjektet.

Mye av tiden har gått på å gjøre seg kjent med programmering og hvordan få ting til å fungere sammen. Oppgaven baserte seg på å teste ut om Kongsberg Maritimes sanntidskontroller, RCU 510 kunne anvendes i andre bruksområder enn slik det er brukt til per i dag. Prosjektgruppen kom i mål med oppgaven og er de første til å lage et trådløst kommunikasjonsnettverk mellom Kongsberg Maritime sitt sanntidskontroller og en mobil sensorplattform.

5 EGENVURDERING

Som et siste kapittel i dette dokumentet vil hvert gruppemedlem gi en egenvurdering på prosjektet og samarbeidet som har ført til det endelige produktet.

5.1 Aleksander Tokle Poverud

Det nærmer seg slutten av en 3års lang bachelorgrad på HSN og hovedprosjektet vi har mottatt fra Kongsberg Maritime er i sluttfasen. Jeg ønsker å reflektere over de erfaringene jeg sitter igjen med etter bachelor prosjektet, som utvilsomt har vert den største utfordringen i studiet.

Det har vert krevende å planlegge og utføre ett slikt omfattende prosjekt. Men det å se at en visjon sakte men sikkert går fra plan og prosess til fungerende delsystemer, og så til ett ferdig produkt, har vert veldig givende. Det har vert tungt, nervepirrende og samtidig ufattelig moro å være prosjektleder, samt gruppemedlem i ett så lærerikt prosjekt.

Faglig sett er jeg fornøyd med oppgaven vi har fått da den er relevant og i stor grad omhandler fagene vi har hatt på HSN. Programmering&mikrokontrollere, elektronikk, signalbehandling, mekatronikk, og ikke minst Systems Engineering, som jeg har fått ekstra god utnytte av som prosjektleder. I tillegg har vi måtte tilegne oss kunnskaper hurtig og tilpasse oss ukjente elementer ved oppgaven, noe studiet har rustet oss til. Selv har jeg hatt god nytte av å ta det jeg har lært i teorien og brukt det i praksis, det har vert interessant å se hvordan kunnskapene overføres til virkeligheten, og hvor annerledes det kan være.

Læringsutbytte for meg har vært enormt, ved siden av det tekniske har jeg fått jobbet mye med organisering og tilrettelegging for gruppen. Jeg har fått ett unikt innblikk i prosess og utførelse, der mye av arbeidet har bestått av å holde gruppen på riktig kurs fram mot ett ferdig produkt. Det har vert spennende å ha ett overblikk over den fulle prosessen, og ett innsyn i alle delsystemene som skulle integreres.

Noe jeg mener vi kan se tilbake på å være stolte av er måten vi har samarbeidet og holdt motet oppe. Kompleksiteten i systemet kunne til tider gjøre at oppgaven virket for stor, hvert enkelt delsystem var avgjørende for at oppgaven og visjonen vi hadde i starten skulle bli virkelighet, og ingen kunne sette seg dypt inn i alt. Jeg ser ved tilbakeblikk at en sterk fremdriftsplan med klare aktiviteter for vær deltaker har vert avgjørende. Mens oppgaven kunne virke uoppnåelig for ett enkelt gruppemedlem, har en strukturert framgangsplan, der vi har hatt muligheten til å se de forskjellige utviklingsstadiene av produktet vert viktig. Det at vi kunne se en gradvis framgang i planen og i utviklingen av systemet mens iterasjonene gikk forbi har vert en motiverende faktor. Det har gitt klarhet og pågangsmot, der vi hele veien har måtte legge tillit til hverandres gjennomføringsevne for å komme i mål. Hvert enkel medlem har spesialisert seg i ett område, som i følge av planen, og kommet tilbake med en løsning vi har tilpasset til hverandres arbeide og integrert. På denne måten har vi som ett team gjennomført prosjektet der alle bitene i puslespillet har kommet sakte men sikkert på plass.

En enestående kvalitet gruppen har oppnådd og utøvd gjennom hele prosjektet er kommunikasjon og tilpasningsevne der vi har jobbet rygg i rygg på samme rom. Planen har ledet oss og ved god kommunikasjon samlet vi systemet til ett, med den funksjonaliteten som har vert ønsket. Vi er ikke lenger en prosjektgruppe, men ett godt sammensatt team, der alle utfyller sin rolle og hjelper hverandre.

Jeg vil gjerne takke Kongsberg Maritime for å gi oss muligheten til denne flotte oppgaven, samt alle veiledere og ressurspersoner som har gitt innspill i dette prosjektet. En ekstra stor takk til gruppen som har gjort dette mulig ved flott samarbeide. Dette har vert en utrolig lærerik, spennende og utfordrende periode, og en super avslutning på bachelorstudiet.

5.2 Masoud Shah Pasand

Hovedprosjektet nærmer seg slutten og jeg ønsker med det å reflektere over de erfaringene som jeg sitter igjen med gjennom denne perioden. Det har utvilsomt vært den største utfordringen i mitt bachelorstudiet, men samtidig så er det den oppgaven jeg har fått størst utbytte av. Prosjektet har vært krevende, men det har gitt meg muligheten til å teste mine teoretiske og praktiske erfaringer som jeg har fått gjennom fagene på mitt bachelorstudiet, og fra tidligere jobberfaring.

Jeg var forberedt på at det skulle gå mye tid til dette prosjektet, men det å komme inn i en gruppe der ikke alle kjenner hverandre var utfordrende. Det tok litt tid før gruppen fant tonen sammen, men vi alle hadde samme visjoner med utførelsen av prosjektet. Mye bruk av tid på god planlegging og deling av spesifikke ansvarsområder på enkeltpersoner gjorde at arbeidet ble lettere og vi har klart å utfylle hverandre. Prosjektgruppen var disiplinerte og hadde faste arbeidstider og oppmøter. I tillegg til god dokumentering av arbeidet som ble utført fikk vi et godt jobbstruktur videre.

Faglig sett syns jeg det har vært interessant, og ikke minst lærerikt å ta del i et såpass stort prosjekt. Det har vært veldig givende å gå fra teori til praksis og fra idé til et ferdig produkt. Ikke bare har det faglige nivået mitt satt på prøve, men jeg har også fått muligheten til å teste mine evner til å samarbeide i en gruppe hvor det har vært mye diskusjoner og faglige drøftinger. Det har også vært interessant å bli kjent med Kongsberg Maritimes sanntidskontroller og Software program, AIM200. Prosjektoppgaven i seg selv har også utfordret meg til å gå utenfor det vi har lært på skolen og søke kunnskap på egenhånd, spesielt med så mange delsystemer i oppgaven. Samarbeidet i gruppen har vært den viktigste bidragsyteren for at dette prosjektet har gått så bra. Prosjektmodellen som vi valgte har også vært et sterkt hjelpemiddel til å veilede prosjektet fra kravspesifikasjoner, ideer og tanker til et ferdigstilt produkt.

Jeg vil til slutt takke alle ressurspersoner som har hjulpet oss med dette prosjektet. Jeg vil også rette en stor takk til de andre medlemmene på gruppen for samarbeidet og et vellykket prosjekt.

5.3 Tor Gunnar Finnerud

Hovedprosjektet og semesteret nærmer seg slutten og med det så vil jeg ta å se litt på hvordan det har vært. Å arbeide med et prosjekt som dette har uten tvil vært det mest krevende i løpet av studiet og jeg sitter igjen med mange inntrykk av det som følger med dette. Må si at det har vært ganske så lærerikt å jobbe med dette.

Oppgaven vi fikk fra Kongsberg Maritime(KM) gikk ut på at vi skulle utvikle kommunikasjons nettverk mellom KMs RCU enhet og en mobil plattform, en edderkopprobot i dette tilfellet. Dette var i stor grad for å teste ut om RCU kunne bli brukt på andre områder enn hva det er i dag. Faglig sett så har det vært ganske programmerings tungt i dette prosjektet, men det har gitt meg en helt ny kjennskap til programmering, samt også en bedre forståelse av hva det innebærer å arbeidet i med et prosjekt. Så jeg har fått bruk for en del av det jeg hadde lært på skolebenken, men også blitt tvunget til å tilegne meg mange nye kunnskaper underveis.

Å jobbe i ei gruppe over såpass lang tid har også vært en ny opplevelse, det har også vært lærerikt med alt det dette innebærer. Fordelingen av arbeidsoppgaver har god, da dette i stor har blitt på bakgrunn av hva hver enkeltes ønsker og hva slags bakgrunn de måtte ha. Det har vært diskusjoner under prosjektet med dette er blitt håndter på en god og ålreit måte. Dette har ikke vært en enkel jobb hele tiden, det har vært perioder med motgang og perioder med medgang, som er en ting det er greit å kjenne på.

Til slutt vil jeg takke alle som har vært tilknyttet på prosjektet på et eller annet hvis, for denne muligheten. Også til gruppen for å ha vært med på gjøre dette til et artig og lærerikt semester.

5.4 Hanna Kåsastul

Bachelorprosjektet nærmer seg slutten samtidig og treårs studieperioden. Ved å analysere denne tiden er det uten tvil vært en lærerik, hektisk og gøy periode som er oppfylt alle mine forventninger. Selv om at det siste semestret har vært det mest hektiske i dette studiet har det også vært den mest spennende tiden hvor hver gruppemedlem viste sin motivasjon, ansvar, samarbeidsevne og kreativitet.

Bacheloroppgaven har vært veldig krevende, og gruppen møtte stadig store utfordringer underveis. Oppgaven er gått ut fra å åpna kommunikasjon mellom Kongsberg Maritime sin realtime kontroller og en mobil plattform som kan styres fra kontrolleren. Gruppen har hatt et stort behov for bruk av kunnskap fra flere fag både elektro, data og maskin linjene og lære mye som i kke har vært inkludert i studieplanen.
Det har vært også en opplevelse å jobbe tett i en liten gruppe i en lang periode. Det har blitt utgjort områdefordeling ut fra personlige interesser og tidlige kunnskaper og erfaring. Men det har vært et samarbeid hele tiden, og alle gruppemedlemmene har vært innblandet i alle delområdene i en viss grad.

Jeg vil takke alle gruppemedlemmer jeg har jobbet sammen med i bachelorprosjektet for stor innsats, god samarbeid og støtte under prosjektperioden. Jeg vil også takke Kongsberg Maritime for den gitte muligheten å delta i så spennende og krevende prosjektet, som er gitt meg nye kunnskaper og samtidig mulighet for å bruke mine kunnskaper jeg fått under studie t. Ikke i det minste vil jeg takke Høgskolen i Sør-Øst Norge for alle de lærerike og dynamiske årene med høyt profesjonelt lærerpersonale. Jeg føler at jeg har fått en stor bagasje med mye kunnskap i forskjellige fagområder for å gå ut i arbeidslivet.

5.5 Paul Knutson Sæther

Vi har nå arbeidet i over ca. et semester, og prosjektet går mot slutten. I relasjon til dette skal hver av oss vurdere hvordan KSpider-prosjektet gikk, som en gruppe, som et prosjekt og hvordan det gikk for oss selv. En slags personlig konklusjon på prosjektet.

Når vi startet prosjektet kjente jeg, som den eneste som ikke var elektrostudent, ikke alle på gruppen. Dette ble relativt fort rettet opp og jeg synes vi fikk et godt gruppemiljø. Vi fikk godt til å jobbe strukturert gjennom hele prosjektet, med faste arbeidstider og god dokumentasjon av arbeidet vi utførte. Jeg synes at både faste arbeidstider og dokumenteringen av det vi gjorde hjalp oss mye til å få til alt vi utførte gjennom prosjektet.

I utgangspunktet fikk vi i oppgave å forbedre en tidligere bacheloroppgave, som også var for Kongsberg Maritime. Dette var en opp-ned pendel, som ble styrt av kontrollsystemet til KM, AIM. Ikke så lenge etter endret de oppgaven til den vi har nå. Oppgaven vår gikk da ut på å delvis lage, delvis konfigurere et kontrollsystem som skulle styre en sensorplatform (som vi også skulle lage).

Gjennom prosjektet arbeidet vi mye, med faste arbeidstider og oppmøte. I starten brukte vi mye tid på å planlegge både prosjektet og systemet vi skulle lage. Dette hjalp oss gjennom hele prosjektet.

Jeg mener vi fikk til oppgaven veldig bra. Vi har oppnådd de fleste kravene bra, og Hexapoden fungerer som den skal. Nå som vi har kjørt en FAT hos KM som gikk bra, tenker jeg at vi har fått til prosjektet ok. Jeg føler selv at jeg har fått mye ut av dette prosjektet, og har lært masse nytt. Spesielt når det kommer til kommunikasjon mellom forskjellige systemer, og kontrollering av autonome eller fjernkontrollerte roboter/sensorplatformer. Jeg har også lært mye når det kommer til prosjektplanlegger, håndtering, dokumentering, o.l.

En av de tingene jeg synes vi kunne ha gjort forskjellig i starten var nok å endre litt på gruppen. Vi var fem elektrostudenter og en datastudent, og dette passet bra til den første oppgaven vår, og vi trodde det passet greit til den vi fikk etterpå. Jeg synes at det hadde vært en fordel å hatt en datastudent til som kunne bidratt med programmeringsbiten fra start, slik at det ikke ble for mye tynge på elektrostudentene. Men det visste seg at elektrostudentene taklet dette veldig bra og de har gjort en god jobb og jeg synes vi greide å utføre oppgaven godt nok.

5.6 Abdurahman Senkaya

I denne egenvurderingen legger jeg fram erfaringer jeg har fått med bachelor prosjektet. Egenvurderingen ser nærmere på hvilke erfaringer jeg har fått i løpet av prosjektet og ser nærmere inn på fagopplegget, konklusjoner jeg har kommet fram til. Jeg har sett at faget har skapt et godt læringsmiljø, hvor vi studenter har fått utbytte av samarbeid, økt motivasjon og læringseffekt av å jobbe i prosjektet og har hatt det gøy.

Vi har som gruppe jobbet med hovedoppgave som omhandler om å sette opp kommunikasjon protokoll via Kongsberg maritime sine kontroller (RCU510) og programvære AIM 2000, som skal styre mobilt plattform.

For meg personlig har kombinasjon å vare far til fire småbarns og studie resultert i meget hektisk periode, jeg måtte legge hobby og andre interesser til side for å fullføre bacheloroppgaven. Oppgaven vi valgte har bydd på store utfordringer, både teknisk og teoretisk. Et slik prinsipp som Arduino robot har blitt utarbeidet mange ganger tidligere, men det er første gang noen gjør det med Kongsberg sine RCU 510.

Gjennom prosjektet har jeg blitt godt kjent med de forskjellige koblings muligheter raspberry pi og Arduino har, jeg hadde kunnskap om arduino miljøet fra før, men hadde aldri vært bort i raspberry pi. Jeg måtte lære meg grunnprinsippet med å bruke raspberry pi og enkelt Python programmering for å få Arduino og raspberry pi til å kommunisere sammen, vi har sett på i2c og seriell kommunikasjon mulighetene, men har blitt enig om å bruke seriell siden fordelene var betydelig større. Jeg har for meste jobbet mye med funksjonene og koblingene til roboten og noe programmering i Arduino.

Jeg føler gruppen vår har fått en god dynamikk der vi har erfart hvordan det er dele på ansvar og målsetninger. Vi har som utgangspunkt vært 6 personer med 6 ulike mål og ambisjoner. Dette har gitt oss utfordringer som vi har måttet løse underveis for å skape fremgang i pro sjektet. Selv om gruppemedlemmene hadde hvert sitt ansvar området, har vi ofte sklidd sammen å klart å utfylle hverandre.

Oppsummeringen min er at jeg har oppnådd gode erfaringer innen prosjektarbeid og samarbeid med gruppemedlemmer. Dette har vært spennende og lærerikt prosjekt, som jeg vil kunne se tilbake til på som positiv opplevelse. Jeg er stolt og fornøyd med det resultatet vi har oppnådd og levert.

6 REFEREANSER

[1] K-Spider 2016, *Prosjektplan 3.0,* Høgskolen i Sørøst-Norge, avdeling Kongsberg.