Universitetet i Sørøst-Norge
Fakultet for teknologi og maritime fag

Bacheloroppgave
2022

Sigurd Terland Danielsen, Julie Håndlykken, Marie Lovise Ulvestad Bastesen, Rikard Wirkola Rasmussen, Sondre Daniel Lindkjølen

Data capture using augmented reality
ARque

ARque

Data capture application using augmented reality

Universitetet
i Sørøst-Norge

semcon

I Abstract

In this bachelor thesis given by Semcon, the project group ARque has developed a seamless way of using Augmented Reality (AR) technology for data capture when measuring the torque of fasteners during assembly. This saves critical data without the need for pen and paper or extra documentation. The prototype created uses the Microsoft Hololens 2 and the NorTronic torque wrench.

II Table of Contents

III List of Figures

IV List of Tables

V List of Listings

VI Acronyms

API	Application Programming Interface
AR	Augmented Reality
ASCII	American Standard Code for Information Interchange
CAD	Computer Aided Design
CLI	Common Language Infrastructure
CLR	Common Language Runtime
CSV	Comma Separated Values
DB	Database
DLL	Dynamic Link Library
DoD	Definition of Done
GUI	Graphical User Interface
HAZOP	Hazard and operability study
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
ID	Identification Documentation
ISP	Internet Service Provider
ISO	International Standards Organization
LAN	Local Area Network
LHS	Left Hand Side
LTS	Long Term Support
LUT	LookUp Table
MRTK	Mixed Reality Tool Kit
MR	Mixed Reality
MVP	Minimum Viable Product

OS	Operating System
RAM	Random Access Memory
REST	REpresentational State Transfer
RF	Radio Frequency
RHS	Right Hand Side
SDK	Software Development Kit
SQL	Structured Query Language
TDS	Torque Data System
UI	User Interface
URL	Uniform Resource Locator
URI	Uniform Resource Identifier
UWP	Universal Windows Platform
VR	Virtual Reality
WinRT	Windows RunTime
WinUI	Windows User Interface
QR Code	Quick Response Code
XR	Extended Reality
XML	Exstensible Markup Language
XAML	Extensible Application Markup Language

VII Words and definitions

- ARque - AR combined with torque
- Polygon (in AR applications) - Triangles used as building blocks in the Hololens
- SQLite - A lightweight SQL database file

- Unity - 3D development software
- Git - GitHub - a shared repository for working in teams
- Repository - A software project that is shared using source control
- Prefab - User defined game objects in Unity

1 Introduction

ARque is a bachelor's project completed at the University of South Eastern Norway (USN), campus Kongsberg, in cooperation with Semcon. Semcon is an international company from Sweden, with an office in Kongsberg. They specialize in product development and have particularly high competence on cybernetics and robotics. They are working to create innovative solutions and have projects in many interesting fields.

For this project Semcon were interested in developing a way to use AR technology in an industrial context. AR is a technology that combines a real world environment with virtual data. The augmented reality market has experienced great growth over the years and the global market size is projected to grow from USD 6.12 billion in 2021 to USD 97.76 billion in 2028 [1]. AR became especially useful during the Covid-19 pandemic when isolation and home offices became a reality for many. For example, it was made possible for on site laborers to get supervised expertise from an engineer working from home. AR has also affected other industries such as healthcare, education and retail, but it is probably most popular within the gaming and media world. AR technology is creating new possibilities and changing how we complete our rather mundane tasks, and we are excited to see if this could be a new enabling technology.

1.1 Project description

As Semcon wanted to explore the use of AR technology in an industrial setting, we were given the task of using a digital tool, specifically a torque wrench, to communicate and send values to an application running in a pair of HoloLens 2 glasses. The goal was for a user to be able to mark an item while wearing the glasses, such as a bolt in an installation, and save torque values to this specific bolt, using the digital torque wrench. The values would also need to be saved in some form of data storage, so that the user can view previously logged data. A reason for developing data capture using a digital torque wrench is to create a seamless way to measure and document the exact torque of all bolts during assembly, in hopes it will help reduce the frequency of fastener failure. We decided the system would consist of three parts. A Hololens application, a digital torque wrench and a server. The server would work as a middle man between the torque wrench and the Hololens, receiving, saving and sending data. The data would mainly consist of torque values measured by the torque wrench, connected to a part ID, chosen by the user in the HoloLens application. Figure 1 shows a simple system overview.

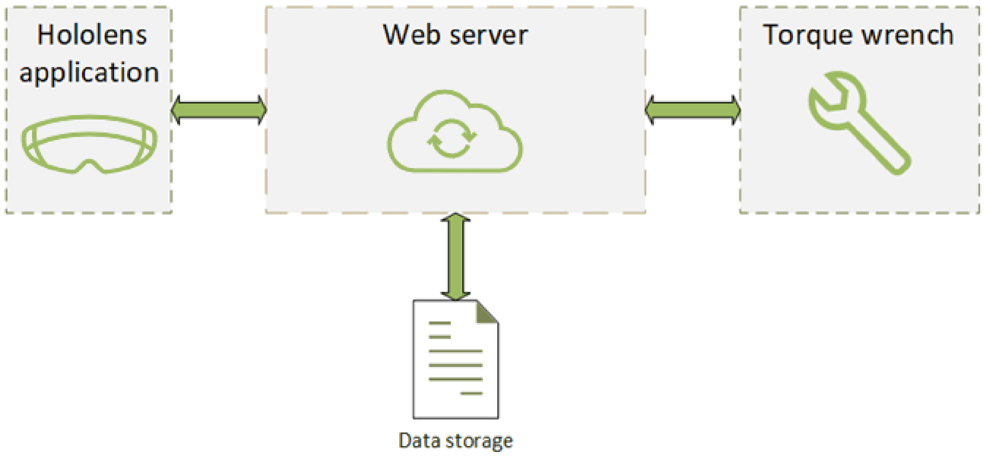


Figure 1: System overview

Another aspect of using AR glasses to collect documentation is for the user to be able to work with both hands free and not need to remember large amounts of information beforehand.

1.2 XR, MR and VR

Extended Reality (XR) is the general term for all real and virtual environments generated by computer graphics and wearables. The areas within this technology, that are worth mentioning, are Augmented Reality, Virtual Reality (VR) and Mixed Reality (MR). VR simulates a virtual environment which can be very different from the real world, such as in video games. But it can also be used for education and in military training, simulating special scenarios for the user to train on. AR is different from VR, because it only uses virtual elements and combines them with a real world environment, often seen through the camera lens on a smartphone or smart glasses. MR is a hybrid between AR and VR, where real world and digital objects interact.

Figure 2: A virtual model as seen through AR glasses (Wikimedia Commons)

1.3 Measuring torque

Typically torque is measured with a mechanical torque wrench. This device is adjusted to a preferred torque number, and gives a clicking sound when the torque value is met. Torque wrenches are sensitive measuring devices often used in harsh environments. This makes them prone to decalibration, and torque wrenches used on particularly sensitive equipment usually needs to be calibrated regularly, to verify that they measure correct. Digital torque wrenches use different technology to internally register the torque, and are therefore less prone to incorrect calibration.

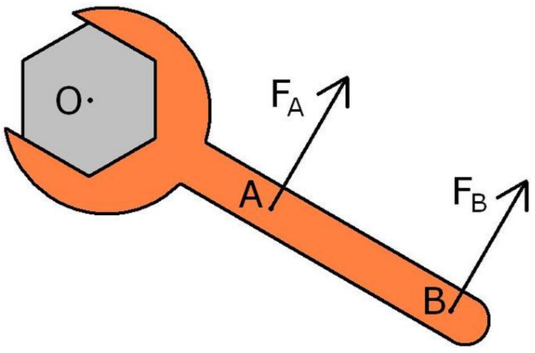


Figure 3: Torque is a measure of Newton meters (Nm)

The measurement of torque is done by measuring the applied rotational force by the wrench to the bolt. The digital torque wrench uses strain gauges to measure this forces. Strain gauges are a lot more precise and reliable than its mechanical counterpart.

1.4 Team members

Group Leader & Risk Manager
Sigurd Terland Danielsen
Electrical Engineer, Cybernetics

Product Owner
Sondre Lindkjølen
Software Engineer, Virtual Systems

Scrum Master
Marie Bastesen
Software Engineer, Embedded Systems

Test Lead & Requirement Manager
Rikard Wirkola Rasmussen
Software Engineer, Virtual Systems

Document Manager
Julie Håndlykken
Software Engineer, Virtual Systems

2 Project Management

In the initial phase of the project the group discussed different project methodologies and which project framework would be the best fit for the project group. The project models discussed were Kanban, Unified Process, Scrum and Agile. The result was an agile model based on the Scrum methodology. The agile process is an iterative method, where one can repeatedly evaluate the completed work, and quickly adapt to changes and discoveries along with the project timeline. Also, since the project was software based the Scrum methodology was a good fit. Because this project consists of many small functions added together, an agile and flexible model was decided to be the most useful.

2.1 Project tools

We started off using Trello for task organization, but once we decided on the Scrum model we changed to Jira, because it covered our project methodology better. The communication tool Discord was used for sending messages to the group, as this was a program the project group already was familiar with. Another advantage was the possibility to have multiple channels for different topics. Zoom was used for digital meetings.

2.2 Scrum and Agile process model

The agile process model gave the group the freedom to affect the supervision and progress of the project. It also made it possible to add aspects of other project methodologies that benefitted the groups preferred process, althoug many of these aspects were inlcuded in the Scrum model. Scrum is interactive, which provided better risk management. It also gave each team member insight in the project as a whole.

In Scrum there are tree main pillars, which are important to the quality of the project outcome. **Transparency**, so stakeholders and the project group have a clear overview of the project framework and the product to be made. **Inspection**, so that any issues and deviations which can hinder reaching the project goal can be addressed. **Adaption**, if any deviations what will put the Sprint Goal and/or the Product Goal at risk the project group should be able to adjust to minimize the damage [2].

2.2.1 Scrum Events

Scrum consists of multiple fixed events to minimise the need for meetings and keep a lightweight process. A project is divided into Sprints, which is a fixed period of time. A sprint starts with a Sprint Planning, followed by a development period and finishes off with a Review and Retrospective [2].

Sprint Planning

This timeboxed meeting is where the project group creates a plan for the upcoming Sprint. Work from the Product Backlog, planned to be done during the Sprint, are placed in the Sprint Backlog. Along with agreeing upon a Sprint Goal the project group answers the questions:

- What should/can be done?
- How will the task be achieved?

Daily Scrum

The Daily Scrum is a short meeting of maximum 15 minutes. It is scheduled every workday where each group member answers the following questions:

- What have you done since the last Daily Scrum?
- What will you work on today/until next Daily Scrum?
- Is there any issues or concerns regards your work that have an impact on the Sprint Goal?

Also, if there are any interesting discoveries that could be of use to the rest of the group. Any further discussions are to be had after the Daily Scrum meeting. This short meeting is important for inspection of the completed work and for the transparency of the project. This is so that the project group can adapt, with updating the Sprint Backlog, and adjust the future work in the Sprint. An example is if there is a task that needs immediate attention or more resources the group can rearrange to focus on these needs as soon as possible.

Sprint Review

The Sprint Review is scheduled at the end of every Sprint. Inspecting the completed work of the finished Sprint is the key for this event. After the Sprint Review, the Backlog should be ready for the upcoming Sprint Planning meeting.

The meeting agenda should follow these steps:

- The Product Owner explains what has been done, and what has not been done.
- The development team gives a demo/explain their work during the Sprint. Discuss what went well and not.
- The Product Owner and team members discuss the Backlog status. The whole Scrum team collaborates to define future work based on the previous Sprint.

The new revision of the Product Backlog after the review meeting should reflect upon the changes that occured during the Sprint.

Sprint Retrospective

After the Sprint Review meeting there should be a Retrospective meeting. The purpose of the Retrospective is to highlight what went well and what could be better, and how to implement that improvement. Compared to the Sprint Review where the focus was "*what* has been done during the Sprint", the focus in the retrospective is "*how* has the work been done following the project methodology".

The Retrospective should give the answers to the following questions:

- What went well in the previous Sprint?
- Can something be improved?
- How can that improvement be implemented?

2.2.2 Scrum Artifacts

Scrum artifacts is the Product Backlog, the Sprint Backlog and the Increment [2]. Several Increments can be created by group members during a Sprint. An Increment is a piece of work that adds value towards the Product Goal. The Sprint Backlog is the items from the Product Backlog selected to be done during the Sprint. The issues in the Sprint Backlog should give the group members a good understanding of the necessary work to be done. The Product Backlog is a list of items which is organized by priority based on importance and project timeline. The project group had a low threshold for adding new items to the Backlog throughout the project. The Product Backlog is not finalized until the end of the project, as new issues are added continuously [2]. The Product Backlog items include functionality, requirements, improvements and more.

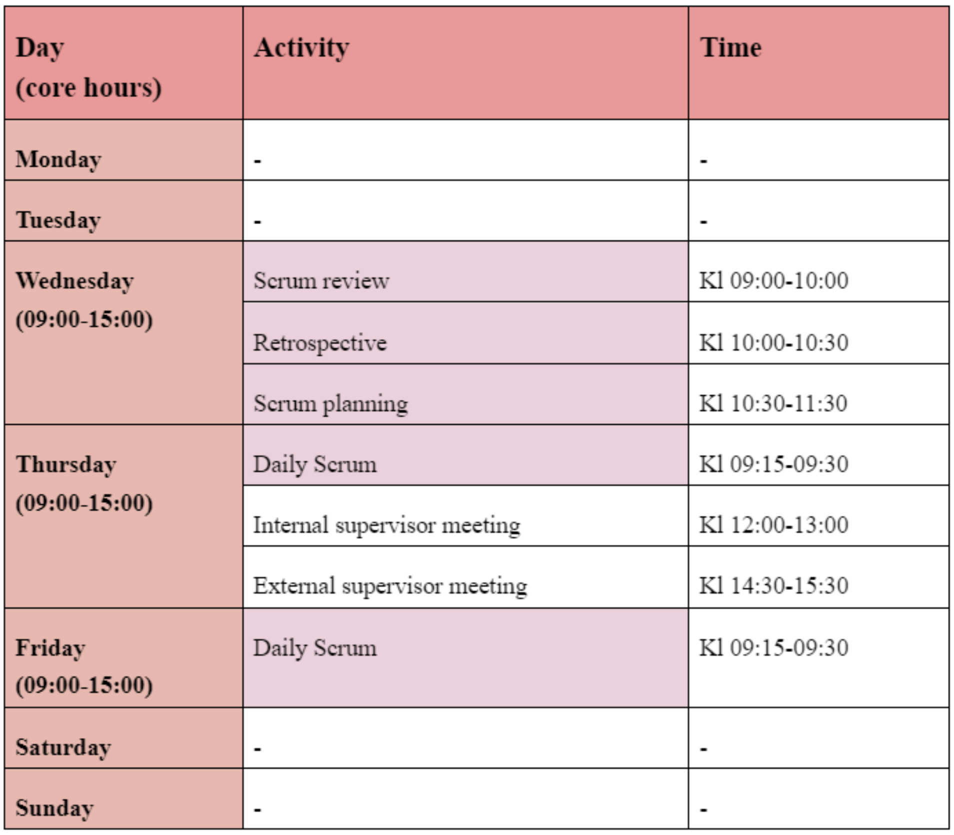
2.3 Practice of the methodology

2.3.1 Meetings and core hours

In table 1 and table 2 all weekly group activities is shown. The Scrum events are marked with a light pink color. These scheduled events are fixed.

During the initial phase the project group tried various ways of scheduling the Scrum events, but after a few weeks of testing we found the presented weekly schedule to be the best fit.

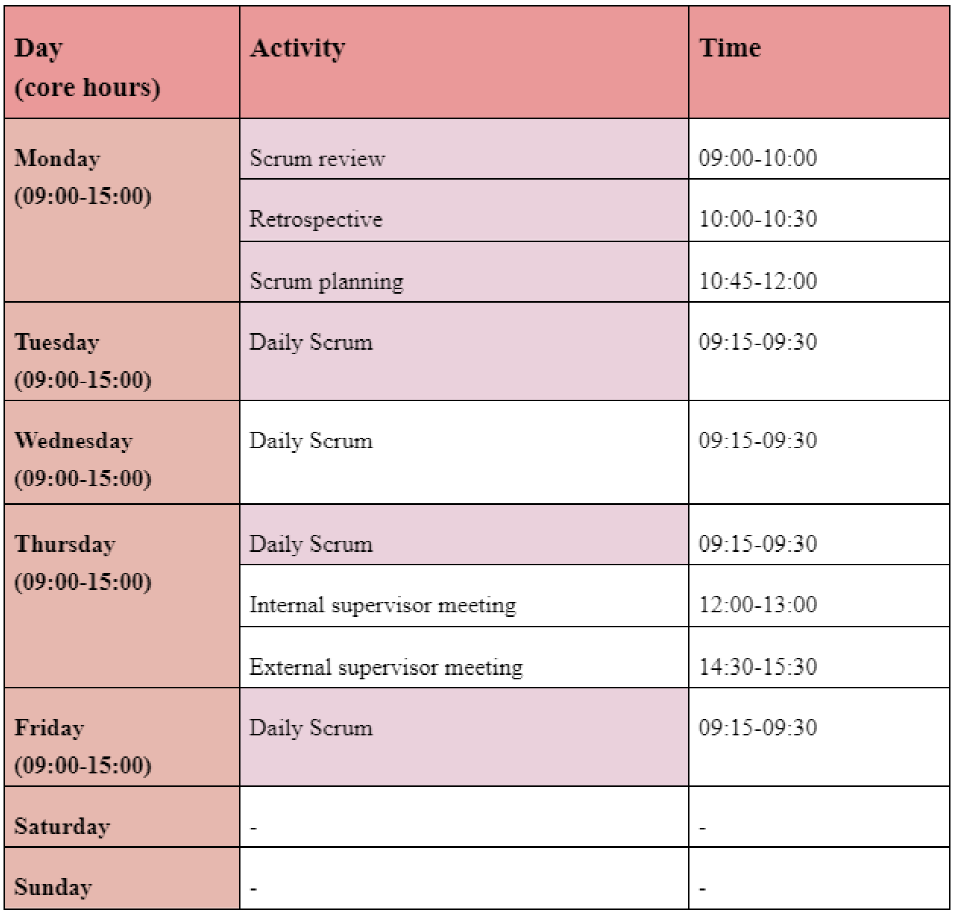
Table 1: Weekly activities before exams



Day (core hours)	**Activity**	**Time**
Monday	-	-
Tuesday	-	-
Wednesday (09:00-15:00)	Scrum review	Kl 09:00-10:00
	Retrospective	Kl 10:00-10:30
	Scrum planning	Kl 10:30-11:30
Thursday (09:00-15:00)	Daily Scrum	Kl 09:15-09:30
	Internal supervisor meeting	Kl 12:00-13:00
	External supervisor meeting	Kl 14:30-15:30
Friday (09:00-15:00)	Daily Scrum	Kl 09:15-09:30
Saturday	-	-
Sunday	-	-

The project group set core hours to be between 9-15 on Wednesdays, Thursdays and Fridays. Core hours is when the group members should be available, unless else is agreed upon.

Table 2: Weekly activities after exams



Day (core hours)	Activity	Time
Monday (09:00-15:00)	Scrum review	09:00-10:00
	Retrospective	10:00-10:30
	Scrum planning	10:45-12:00
Tuesday (09:00-15:00)	Daily Scrum	09:15-09:30
Wednesday (09:00-15:00)	Daily Scrum	09:15-09:30
Thursday (09:00-15:00)	Daily Scrum	09:15-09:30
	Internal supervisor meeting	12:00-13:00
	External supervisor meeting	14:30-15:30
Friday (09:00-15:00)	Daily Scrum	09:15-09:30
Saturday	-	-
Sunday	-	-

2.3.2 Jira for handling workflow and progress

The project group used the project tool Jira for process handling and Sprint Planning. A hierarchical view of how the work was divided is shown in fig. 4, with a structure consisting of an initiative, epics, stories, and sub tasks. The initiative is a large component that lead to a one common goal, and consists of epics which are smaller pieces of work but still not small enough to be finished in one Sprint. The epics are therefore divided into stories or tasks that can be completed throughout a Sprint. The epics and stories are written in a typical user story expression, such as: "As a [persona], I want [need], so that [purpose]" [3]. User stories gives context to the the work ahead with focus on the goal and user.

As shown in fig. 4, we structured two categories of the project; Product and Project management. The epics in Project management are the Sprint admin epic and the documentation and presentation epic, which again is divided into stories and tasks, and then sub tasks.

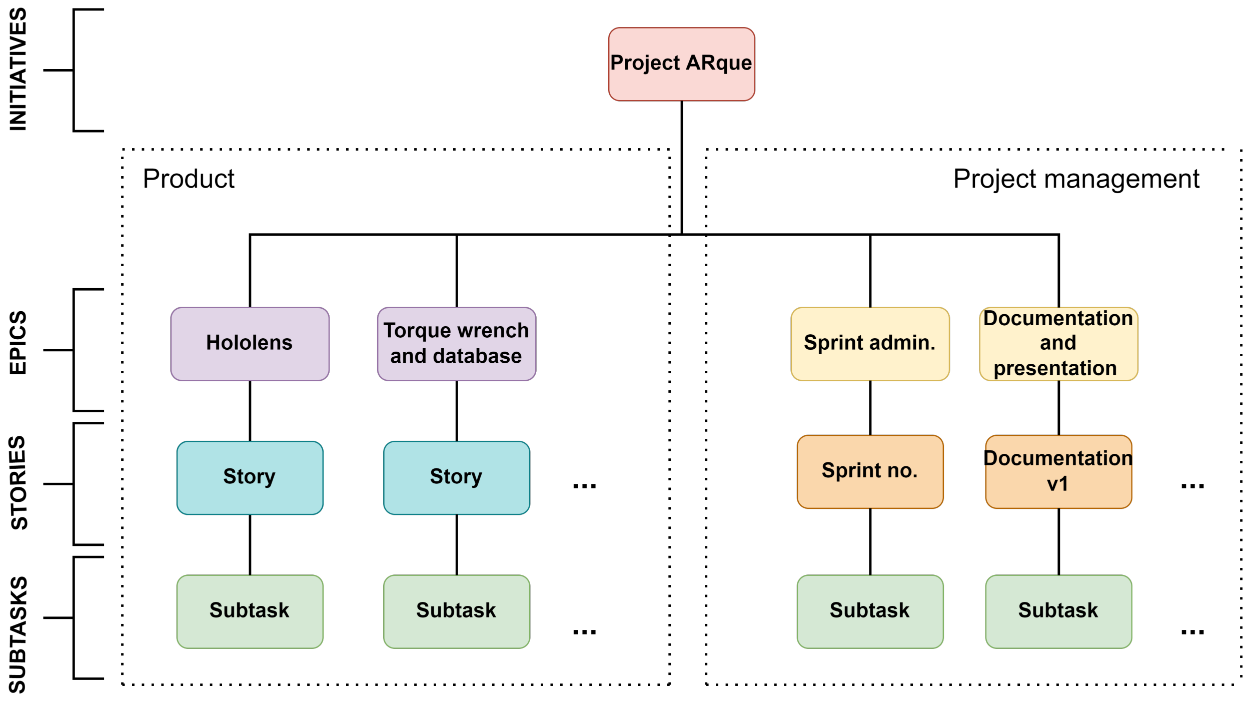


Figure 4: The structure of work

2.4 Project milestones and progress

As seen in table 3, the project group created milestones they would work towards, so the process could be measured.

Table 3: Project milestones

Milestones	Date
M1 - Initial phase	17.02
M2 - MVP	23.03
M3 - Prototype	29.04
M4 - Documentation hand in	23.05

Initial phase

The goal of the initial phase was to create a foundation for the project by the first presentation.

- Get a common understanding of the project and the final product to be made
- Research and finding the correct tools, workflow and project model fitting for the team and project
- Distribution responsibility in the team
- Create templates for necessary documents and structure for the documents

Nothing in the initial phase is 100% fixed, since Scrum is an agile model, changes throughout the project is expected. However, it was necessary for the group to focus on these bullet points during this period to create a good overview of the project.

MVP

After the initial phase, the client asked for a demonstration of a Minimum Viable Product (MVP). An MVP should give the client an indication of what the finished product will look like. This gives the client the opportunity to give feedback early in the process, and the project group to implement eventual changes based on the clients wishes. This milestone was important because it forced the group to develop a product based on the most essential components and requirements, in a short amount of time. It also gave the group an indication of new risks and challenges to be taken into account before continuing further development.

Prototype

The third milestone set to 29th of April, was a further developed prototype, then the MVP. The goal was for the prototype to have a product where the torque wrench could be connected to and communicate with the HoloLens. It should also have the ability to have data storage. Another added goal for the prototype was to have an application in the HoloLens to position the 3D model. By the milestone date we had accomplished to get a connection between the server and the torque wrench application using Named Pipes. The server had the possibility to store data in an Extensible Markup Language (XML) file and could communicate with a client using HyperText Transfer Protocol (HTTP) GET requests. The only thing missing was the communication from the HoloLens which was achieved in the following week, on May 6th. The HoloLens application was able to send HTTP requests to the server, and in that way the communication could go all the way from the torque wrench to the HoloLens application.

Documentation hand in

Optimize the product for final hand in and complete documentation. After the third milestone the group worked on further development by adding more functionalities to the system, to get a more robust system. Development stopped the last week before the project deadline, as the project group had decided to spend the last week on documentation.

2.5 Project roles

In the Scrum model there are three types of roles defined; Scrum Master, Product Owner and Team Member/Developer. In addition to these, the project group decided to add the following roles; Team Lead, Document Manager, Test Lead, Requirement Manager and Risk Manager. The group found this to be a useful addition because it provides everyone with an area they can specialize in, and feel a sense of responsibility.

Scrum Master

The Scrum Master is responsible for making sure the group follows the agile framework by promoting the Scrum values, theory, rules and practice. They are responsible for ensuring the Scrum events take place within the fixed timeboxes and follow the agenda. The Scrum Master should make sure the three pillars of Scrum are met and the four activities of Scrum are done, such as explained in section 2.2.

Product Owner

The Product Owner has the responsibility of maximizing the value of the project. One of the most important responsibilities is to order the Backlog, and to keep it comprehensible and clearly expressed for the team to understand. The Backlog is fundamental to make the Sprints as productive as possible and to maximize the value of the work the team delivers. Tasks related to the Backlog can be delegated to the development team, but the Product Owner is still the one accountable [2].

Team Lead

The Team Lead was responsible for the contact with our primary stakeholder, Semcon. This was to ensure continuity in the communication and that the company only had to refer to one person throughout the project. The Team Lead was also responsible for the overall atmosphere in the group and to eliminate the risk of conflict.

Risk Manager

The Risk Manager was responsible for keeping a thorough risk analysis throughout the project. The analysis needed to be updated frequently to make sure risks and risk levels are a reflection of the projects current status. The Risk Manager also had the main responsibility of highlighting risks that might appear, and communicating these to the rest of the group.

Requirement Manager

The Requirement Manager was responsible for making sure requirements and user stories align, and that they are easy to understand for the rest of the group. Requirements also had to be testable and include a Definition of Done (DoD).

Test Lead

The Test Lead was responsible for making sure the requirements were testable and that they align with the wanted outcome of the project user stories. The Test Lead also had to make sure the requirements go through a sufficient test before before completion.

Document Manager

The Document Manager was responsible for ensuring that documentation is delivered on time and that all necessary documents are available to the group. The manager was also responsible for document templates, the structure of content and references in project documents, and the file structure in Google Drive.

3 Risk Management

To prevent unnecessary delays in the project a risk analysis was established in the initial phase as an attempt to predict where the team could encounter obstacles. The risk analysis is a tool and is used to ensure that any obstacles or challenges are handled in a meaningful manner, and contribute to the project's progress in a positive way. For this project, the risk analysis was carried out early, and helped lay the foundation for how the project group would handle incidents, deviations and other challenges. The risk analysis was separated into project management risks and technical risks, where we assess the system performance.

3.1 Project management risks

The risks in the project management part were an important part of the skeleton of the model we had chosen. Since Scrum is an agile method, the risk analysis should be under constant consideration with revisions along the way. This lets us document when we see that something is either not as big of a risk or that something is a greater risk than what it was considered to be during the project's start. It was also not realistic to be able to map all the risks at the beginning of the project. Undeniably, unforeseen issues will emerge. These require targeted action and will help define the course of the product development. The group collaborated on a risk analysis session to map the vulnerabilities we found related to the project management. The most severe risks identified were related to Covid-19 infection, with a very high probability of occurring, and conflicts within the group. Possible conflicts scored high on severity, but not as high on probability.

3.2 Technical risks

The risks associated with the technical part were also necessary to revise during the course of the project. For the client to be presented with a good product that meets their requirements, it was important to use the risk analysis actively when mapping the technical specifications. If the system was to be developed past a proof of concept level the systems expected performance would have to be evaluated at each milestone. The technical risk analysis was used as a tool for the continuing evaluation of the progress, ranking of to dos and adjusting the realistic expectations during the course of the development.

3.3 Categorizing and scoring risks

The identified risks were categorized according to severity and probability. We based our assessment on a collective score of how likely a risk were to occur, and how severe each team member thought the risk would impact the project. The final scores were weighted as an average of the scores. The final scores are a mix of guesstimates based on discussion and experience from members familiar with the technology. We created the matrix in table 4 with inspiration from risk scoring in HAZOP analyses.The colorscheme we used was later used to identify and prioritize system tests for performance evaluation. For this specific project the

Table 4: Risk-scoring matrix

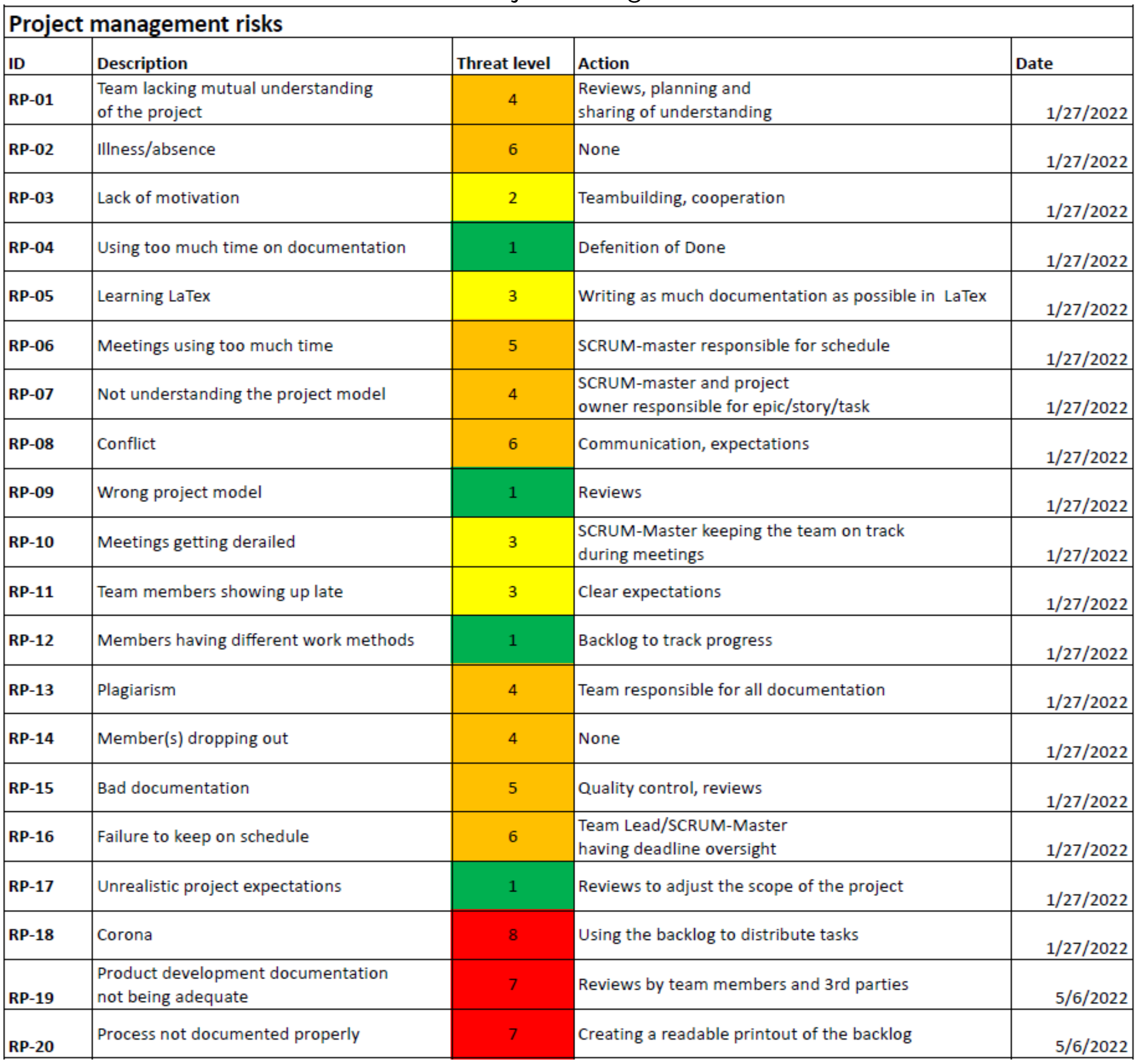
		Severity		
		Low	**Medium**	**High**
Probability	**Low**	1	2	4
	Medium	3	5	7
	High	6	8	9

immediate risks regarding project management was a mix of not fully understanding or utilising the chosen model, or a failure to set realistic expectations for the progress. Early on, the group created templates for all recurring documents, so incorrect documentation was seen as an insignificant risk. This does not mean there were no risk of failure, but if we implemented Scrum correctly it would stop any development of bad habits as soon as they appeared. Using Scrum we were confident that the project would be as agile as possible.

3.4 Risk analysis

The technology used in this project was still under development, and did not have the same familiarity as other mature products. This meant that the documentation would be sparse, or non existent for some important functionality. We were also using equipment that was not open source, which meant software updates could potentially make something the team had implemented unusable. To mitigate this the team was looking to create a standalone application to run on the HoloLens operating system, which is Microsoft Holographic. We were also tasked with making two separate units communicate, hopefully in real time. The torque wrench we were using uses proprietary software to collect the measured values with their work IDs from the wrench, further described in section 7. The software relied on user interaction, meaning this process had to be automated in some way.

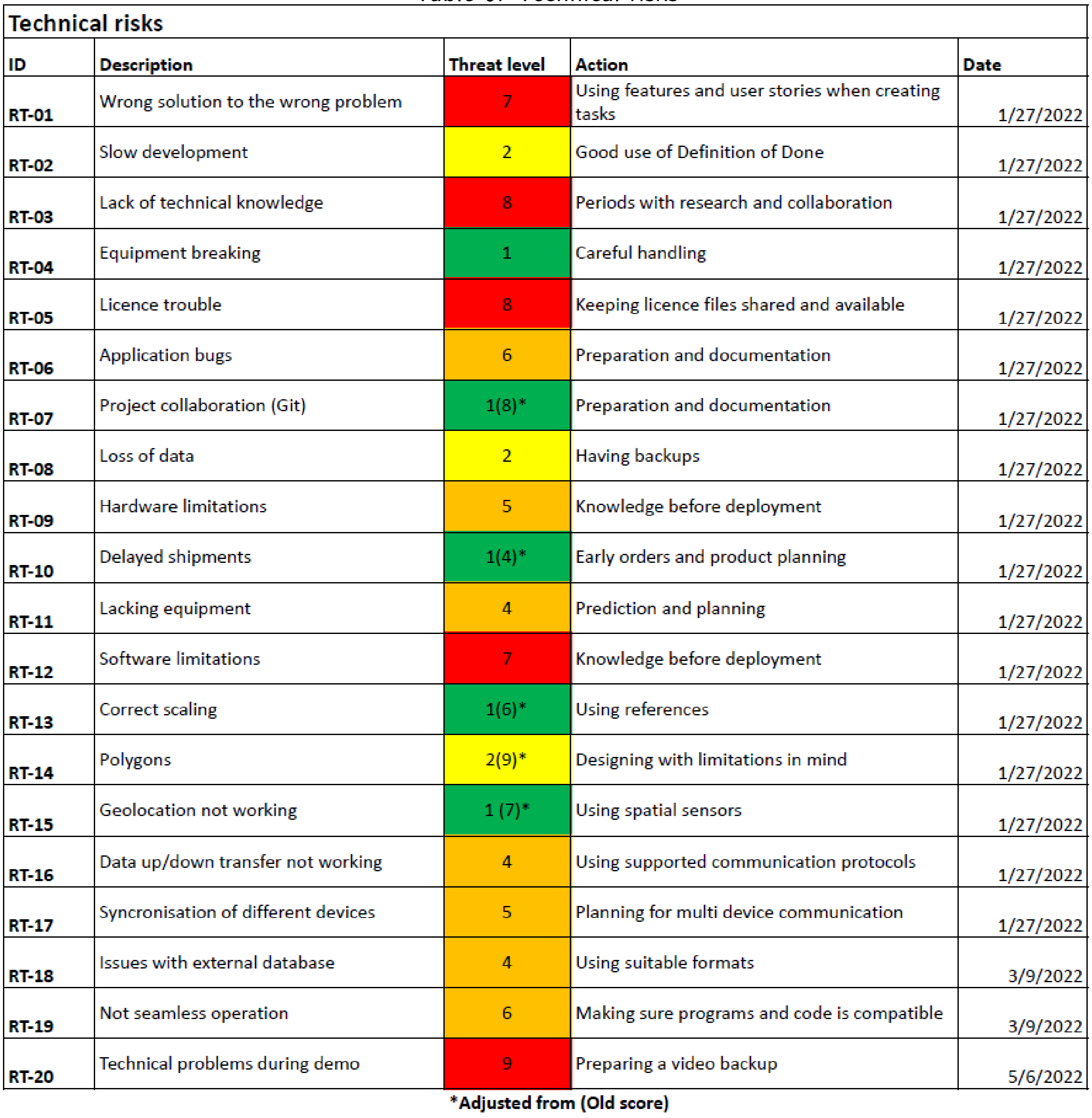
Table 5: Project management risks



Project management risks

ID	Description	Threat level	Action	Date
RP-01	Team lacking mutual understanding of the project	4	Reviews, planning and sharing of understanding	1/27/2022
RP-02	Illness/absence	6	None	1/27/2022
RP-03	Lack of motivation	2	Teambuilding, cooperation	1/27/2022
RP-04	Using too much time on documentation	1	Defenition of Done	1/27/2022
RP-05	Learning LaTex	3	Writing as much documentation as possible in LaTex	1/27/2022
RP-06	Meetings using too much time	5	SCRUM-master responsible for schedule	1/27/2022
RP-07	Not understanding the project model	4	SCRUM-master and project owner responsible for epic/story/task	1/27/2022
RP-08	Conflict	6	Communication, expectations	1/27/2022
RP-09	Wrong project model	1	Reviews	1/27/2022
RP-10	Meetings getting derailed	3	SCRUM-Master keeping the team on track during meetings	1/27/2022
RP-11	Team members showing up late	3	Clear expectations	1/27/2022
RP-12	Members having different work methods	1	Backlog to track progress	1/27/2022
RP-13	Plagiarism	4	Team responsible for all documentation	1/27/2022
RP-14	Member(s) dropping out	4	None	1/27/2022
RP-15	Bad documentation	5	Quality control, reviews	1/27/2022
RP-16	Failure to keep on schedule	6	Team Lead/SCRUM-Master having deadline oversight	1/27/2022
RP-17	Unrealistic project expectations	1	Reviews to adjust the scope of the project	1/27/2022
RP-18	Corona	8	Using the backlog to distribute tasks	1/27/2022
RP-19	Product development documentation not being adequate	7	Reviews by team members and 3rd parties	5/6/2022
RP-20	Process not documented properly	7	Creating a readable printout of the backlog	5/6/2022

Table 6: Technical risks



Technical risks

ID	Description	Threat level	Action	Date
RT-01	Wrong solution to the wrong problem	7	Using features and user stories when creating tasks	1/27/2022
RT-02	Slow development	2	Good use of Definition of Done	1/27/2022
RT-03	Lack of technical knowledge	8	Periods with research and collaboration	1/27/2022
RT-04	Equipment breaking	1	Careful handling	1/27/2022
RT-05	Licence trouble	8	Keeping licence files shared and available	1/27/2022
RT-06	Application bugs	6	Preparation and documentation	1/27/2022
RT-07	Project collaboration (Git)	1(8)*	Preparation and documentation	1/27/2022
RT-08	Loss of data	2	Having backups	1/27/2022
RT-09	Hardware limitations	5	Knowledge before deployment	1/27/2022
RT-10	Delayed shipments	1(4)*	Early orders and product planning	1/27/2022
RT-11	Lacking equipment	4	Prediction and planning	1/27/2022
RT-12	Software limitations	7	Knowledge before deployment	1/27/2022
RT-13	Correct scaling	1(6)*	Using references	1/27/2022
RT-14	Polygons	2(9)*	Designing with limitations in mind	1/27/2022
RT-15	Geolocation not working	1 (7)*	Using spatial sensors	1/27/2022
RT-16	Data up/down transfer not working	4	Using supported communication protocols	1/27/2022
RT-17	Syncronisation of different devices	5	Planning for multi device communication	1/27/2022
RT-18	Issues with external database	4	Using suitable formats	3/9/2022
RT-19	Not seamless operation	6	Making sure programs and code is compatible	3/9/2022
RT-20	Technical problems during demo	9	Preparing a video backup	5/6/2022

***Adjusted from (Old score)**

3.5 Risk reviews

Table 7: Risk revisions

No.	Description	Date
1	Initial review	01/27/22
2	First revision	02/09/22
3	New risks added	03/09/22
4	Final revision	05/06/22
5	Rewritten wording	05/10/22

When we started with the thecnical development we had to review the technical risks after some time since the scope of the project had been further adjusted. Some of the identified technical risks were no longer relevant, such as RT-15. After gaining insight about the technology the group also had a much better understanding about the technical limitations, so planning for and mitigating risks were a much simpler process. The latest review in table 7 also identified new risks that were not considered at the beginning. The problems were related to data handling with an external database, which was not considered in the early stages of the project. To make the project as good as possible we decided to go for a system divided into three; the HoloLens application, the Norbar torque wrench system and a server solution to maximise the system performance.

3.5.1 Non-identified risks

As discussed earlier, the involved technology presented many problems during the development of the technology. The torque wrench did not support direct communication on a hardware level, and the firewall at the school campus had intermittently blocked packages sent to and from the HoloLens. The NorTronic problems were not identified earlier because of a misunderstanding in the products manual. This particular problem had a huge impact on the implementation of a true real time system, since this meant we had no way to communicate without the proprietary software, and the software needs to be operated by a user to synchronize data. We managed to find a workaround to this problem, which ended up working very well. To further elaborate the unidentified risks involved in this project we made a list of known issues and bugs in section 11.3.

4 Requirements

Requirements for this project were prepared and set by the group with final approval from Semcon. The requirements can be seen in section 4. The parameters used in the table contains the requirement ID, the requirement description, when the requirement was approved and which priority it has. Requirements with priority A shall be completed during the project. Requirements with priority B are desirable if all A priority requirements have been completed.

Table 8: Requirements

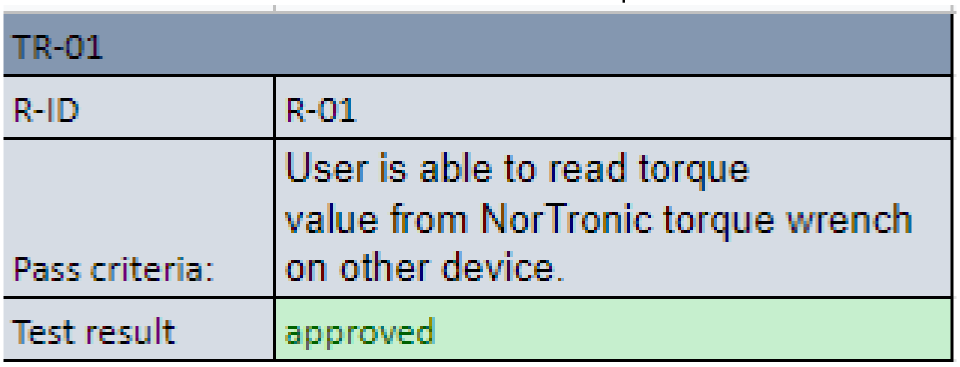
R-ID	**Description**	**Date up-dated**	**Requir-ement**
R-01	The torque wrench subsystem shall capture torque values from the torque wrench	27.01	A
R-02	The torque wrench subystem shall store data from the torque wrench in a data storage with wireless connection	27.01	A
R-03	The HoloLens subsystem shall have a stable build environment in Unity	14.01	A
R-04	The Hololens subsystem shall be able to build and run an application in the HoloLens	27.01	A
R-05	The HoloLens subsystem shall upload a CAD model into the application	27.01	A
R-06	The HoloLens subsystem shall visualize a CAD model	27.01	A
R-07	The HoloLens subsystem shall allow a specific part to be selected in the application	27.01	A
R-08	Hololens subsystem shall be able to locate an object in HoloLens application	30.03	A
R-09	The HoloLens and torque wrench should communicate as one system	27.01	B
R-10	The system should read the correct data from the torqued part selected inside the HoloLens 2 application	27.01	B
R-11	The system should upload a model from a database outside the system	27.01	B

4.1 Testing

The project group ran tests to see if the requirements were met. The test results, as well as the related requirements were placed in test cards. Each of the test cards contain a unique ID with the related requirement, as well as the criteria that had be satisfied for the test to be approved. An example of a test card can be seen in table 9, followed by the methods of how the requirements were tested.

Test example

Table 9: Test card example

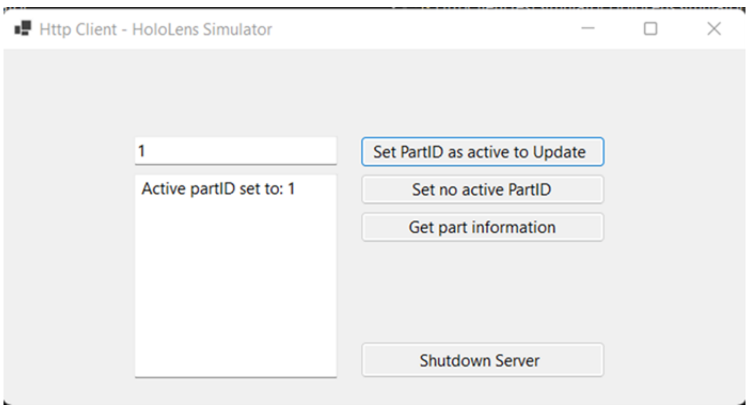


TR-01	
R-ID	R-01
Pass criteria:	User is able to read torque value from NorTronic torque wrench on other device.
Test result	approved

By running the server, the torque wrench application and the HTTP client simultaneously the criteria in TR-01 seen in table 9 can be validated. The steps to validate this test are further explained in the itemized list below.

- Entering partID "1" and click "Set PartID as active to Update" button as shown in the 5
- Torque a bolt on the test bench, and click store on the torque wrench, as shown on the 7
- After clicking the "get part information" button in the HTTP client GUI, the torque wrench values will show, see 6

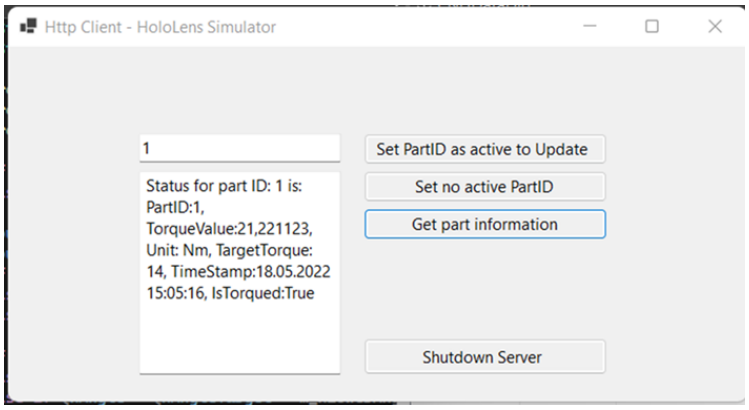
This test is approved.


Http Client - HoloLens Simulator

1

Active partID set to: 1

Figure 5: HTTP test client GUI for testing - setting partID to 1 to active


Http Client - HoloLens Simulator
1
Set PartID as active to Update
Set no active PartID
Get part information
Shutdown Server

Status for part ID: 1 is:
PartID:1,
TorqueValue:21,221123,
Unit: Nm, TargetTorque:
14, TimeStamp:18.05.2022
15:05:16, IsTorqued:True

Figure 6: HTTP test client GUI for testing - showing part ID information

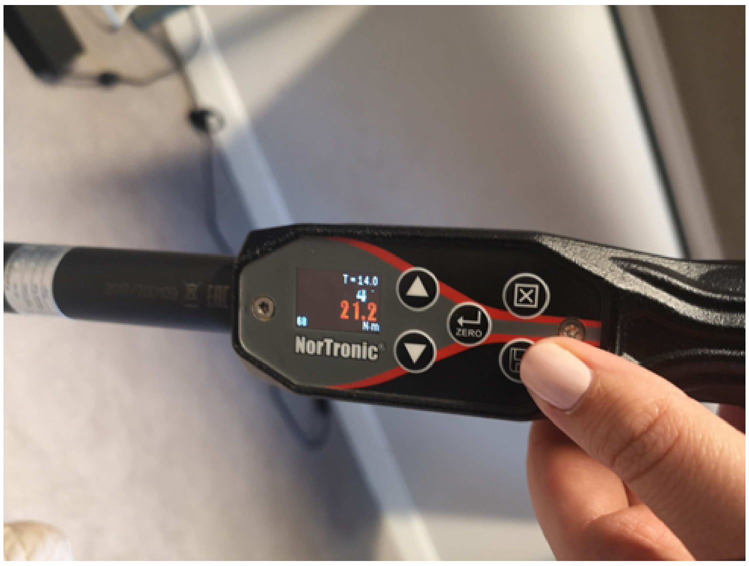


Figure 7: NorTronic torque wrench displaying a torque value torqued on a bolt

5 System Architecture

The diagram in fig. 8 is a basic overview of the system architecture. The HoloLens application receives torque values from the NorTronic torque wrench by communicating through a server. The server also handles the data storage. The torque wrench application communicates with the server via Named Pipes, while the HoloLens application communicates via HTTP requests.

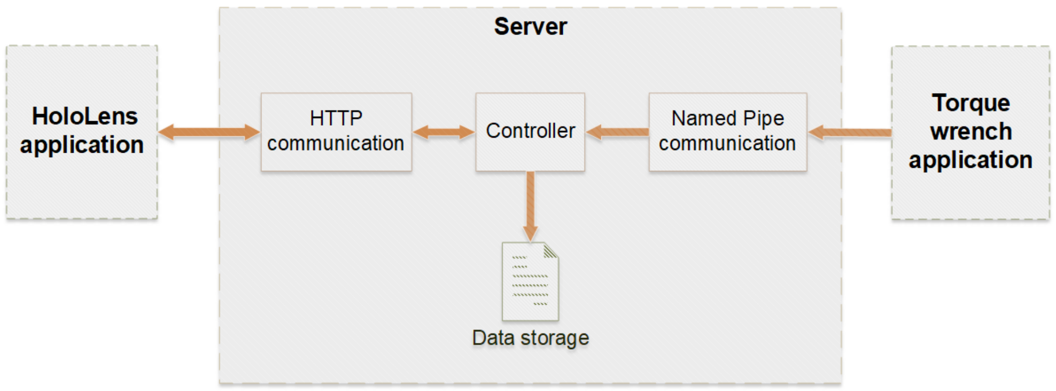


Figure 8: Block diagram of the overall system

5.1 System functions

Use case and sequence diagrams are created to visualize the functionality and communication within the system. The use case diagram seen in fig. 9 visualizes the functionality of the system by defining three actors, the HoloLens operator, the ARque server and the torque wrench application.

The operator using the HoloLens application can get part data from the server, or measure torque using the NorTronic torque wrench after activating a specific part. The torque wrench application sends data to the server. While the server can, based on the HoloLens application and the torque wrench application, update the part data. The server can also give an overview of the part data, and store it to an XML file.

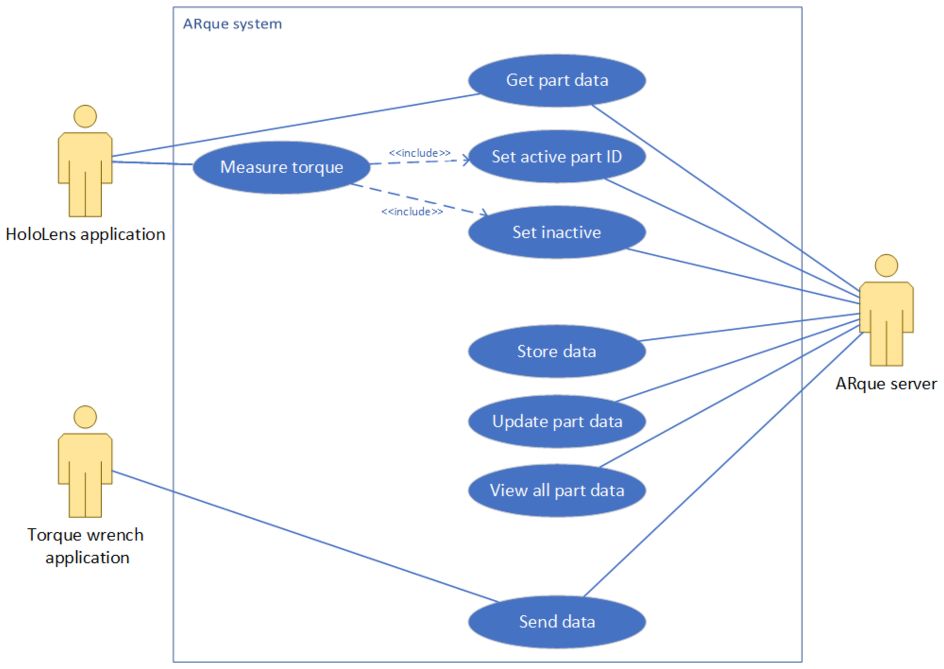


Figure 9: Use case diagram of system functionality

The torque wrench application gets connected to a RF transceiver, as shown in fig. 10. It retrieves strings of data coming from the transceiver which is sent to the server.

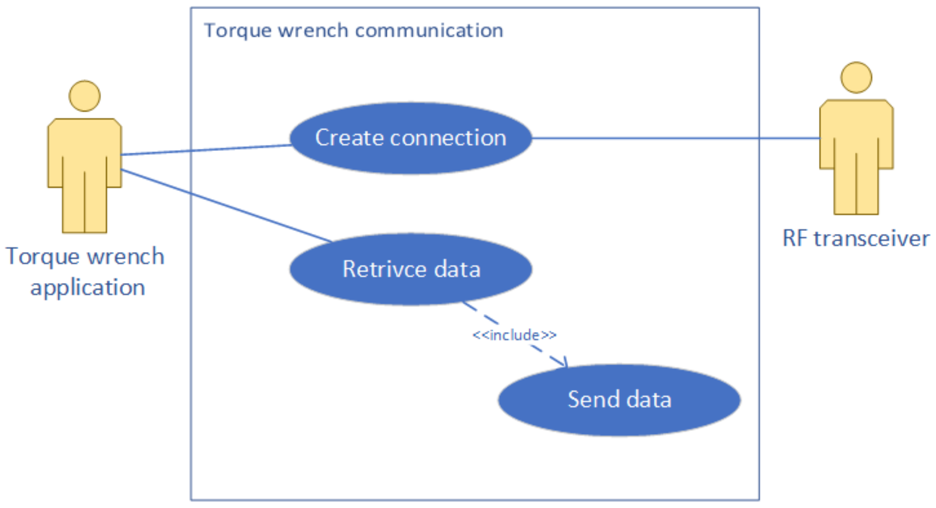


Figure 10: Use case diagram of torque wrench communication

6 Server

A server provides a service. It responds to requests from devices, called clients, connected over a network. A server can be hosted on any device that can store, process, and share data. There are a wide variety of servers that provides a variety of services. For example, a file server is a computer on a network with files which other computers can access. A print server is a host with a printer attached which other devices on the same network can access to be able to use the printer. Application servers are used to host applications. The clients do not need to download the data but get access to the application through a virtual server connection. A web server, used to host websites and runs web server software. HTTP is a commonly used protocol used for communication between the clients and a web server. A database server typically operates along with other types of servers. It uses a database application to provide the services or storing data in groups [4].

The server acts as the binding link between the systems HoloLens application and the torque wrench data retrieval system. The torque wrench with data storage and management, can be seen as the backend of the system, while the HoloLens application serves as the frontend. One of the requirements of the system, see section 4, and a big focus for the customer was to route data all the way from the torque wrench to the HoloLens, in real time. Along with the ability to log torque values for each bolt tightened, and notify of remaining bolts to be tightened. Our architectural philosophy was to have the server contain all data, and the HoloLens act as a stateless frontend, in terms of collected torque data. In this way any HoloLens with the frontend application installed can be connected to the server, and with that connection resume any previous session.

6.1 Communication

A web server can both store and deliver data to a web browser, typically through HTTP. HTTP is also used to communicate between software applications, in a communication form known as REpresentational State Transfer (REST) Application Programming Interface (API). We chose to use this architectural style for our communication between the HoloLens and server. HTTP uses various types of request methods, where some of them are described in table 10. With these HTTP requests, the HoloLens can communicate with the server, to retrieve information and to update the information in the server.

Table 10: HTTP methods

HTTP Methods	Description
GET	Request and retrieves data from a resource in the server.
POST	Sending data to the server to update or create a resource.
PUT	Sends data to the server to create or update
DELETE	Deletes resources that is indicated in the URL

REST is an architecture style which define a set of constraints for how a system should behave. REST API is usually based on HTTP methods to access resources via a Uniform Resource Locator (URL) encoded parameters. Requests made to a resource Uniform Resource Identifier (URI) result in a response containing a payload. This can be formatted in XML, JavaScript Object Notation (JSON), HTML or others. When using HTTP as we have decided to do, the HTTP methods provide the operations available as described in table 10. A large part of REST is statelessness in the protocol, which means the URIs do not change based on system state. Instead, the response is different [5].

Since we divided the project into multiple subsystems, REST makes it possible to work on each subsystem independently. REST is also easy to debug since the communication does not rely on a complicated state machine. This means trivial simulators, or even a regular web browser, can be used to test and debug the communication. Having this architectural style made the application less complex, and easier for the project group to achieve the communication between the subsystems without a long and difficult integration period.

When the client, the HoloLens, sends a request to the server, the servers response is a representation of the state of the resource that is requested. This means when the server gets a request to get information about a specific part ID, the server responds with the current state of that resource.

For the project the POST and GET requests are used. The GET method is to retrieve information about a specific part. The POST method is used to do a change to the servers resources, meaning when a POST request is sent to the server with a part ID, this part ID will be set as active. By setting the part ID as active, the next torque data sent to the server will update the active part IDs information.

The data that is sent in the HTTP request body, and can be formatted in many ways. Besides using raw text and parsing it in any manner one chooses, some industry standards are JSON, XML, or HTML format. For our project the format the representation of the state is sent as a string, though we have prepared the code to use JSON with the use of a C# DataContract in the PartData class. This allows C# library functions to quickly serialize and deserialize the class to or from a JSON string.

The web server designed for the project has one-way communication, from the client to server. As shown in the fig. 11, this implies that the HTTP client must first send a POST request to set an active part ID. New torque values from the torque wrench will then be stored under this part ID. In the mean time, the client must continuously send GET requests to get the updated part data. Once a new torque value has been received in the server, the HoloLens will see these changes as a change in the response from the GET request. This is a simple way of communicating, but has the disadvantage that the frontend client has to poll the server to get the data. Since the HoloLens only displays information about the selected part, this is considered acceptable.

The server has a state machine, while the frontend client can be seen as stateless in terms of total system function. The server state is determined by the integer activePartID and the flag inactivePartID. In the the sequence diagram in fig. 11 the communication flow is shown. The loop is showing that as long the server is in the state where a part ID is set and the flag inactivePartID is false, all new torqueData from the torque wrench application will overwrite and be stored with the part ID.

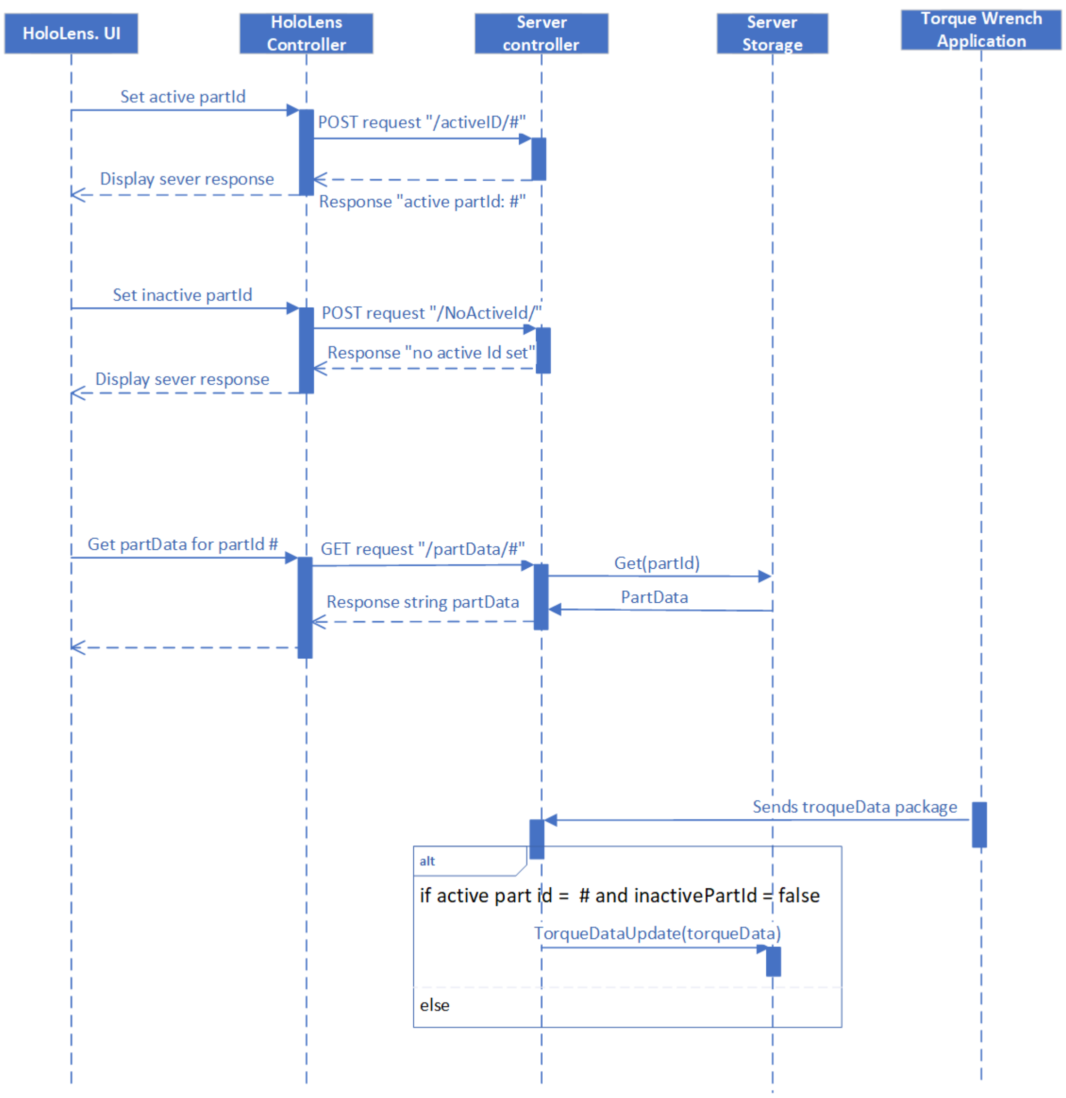

HoloLens. UI
HoloLens Controller
Server controller
Server Storage
Torque Wrench Application
Set active partId
POST request "/activeID/#"
Display sever response
Response "active partId: #"
Set inactive partId
POST request "/NoActiveId/"
Response "no active Id set"
Display sever response
Get partData for partId #
GET request "/partData/#"
Get(partId)
PartData
Response string partData
Sends troqueData package
alt
if active part id = # and inactivePartId = false
TorqueDataUpdate(torqueData)
else

Figure 11: Sequence diagram of the system interactions

6.2 Data storage

The costumer wants a system that can store the torque values that have been applied to a bolt. Meaning that when the operator of the system selects a bolt in the HoloLens the next torque value the operator torques the torque wrench with should be connected and stored on the selected bolt. To solve this we choose to have a list with the part and their torque data. So when the operator selects a part and stores a torque value, it will be added to the list. The list can be stored in an XML file. In the future as the system grows, a more scalable means of storage such as a database should be used instead of a list which is stored in the Random Access Memory (RAM). The part of the server code that handles storage of the data in a list, and converting to an XML file uses an interface. The same interface functions can be used to do the same work towards a database instead. In this way we have kept the server implementation independent of the exacted data storage used.

6.3 Server architecture

The server that was created for the project is written in C#. The reason for this is that C# is high level, simple to use, and we only need Windows support for the project. The fig. 12 shows a class diagram of the server system structure showing all the classes, the attributes, and methods in each class, and the relationship between the classes.

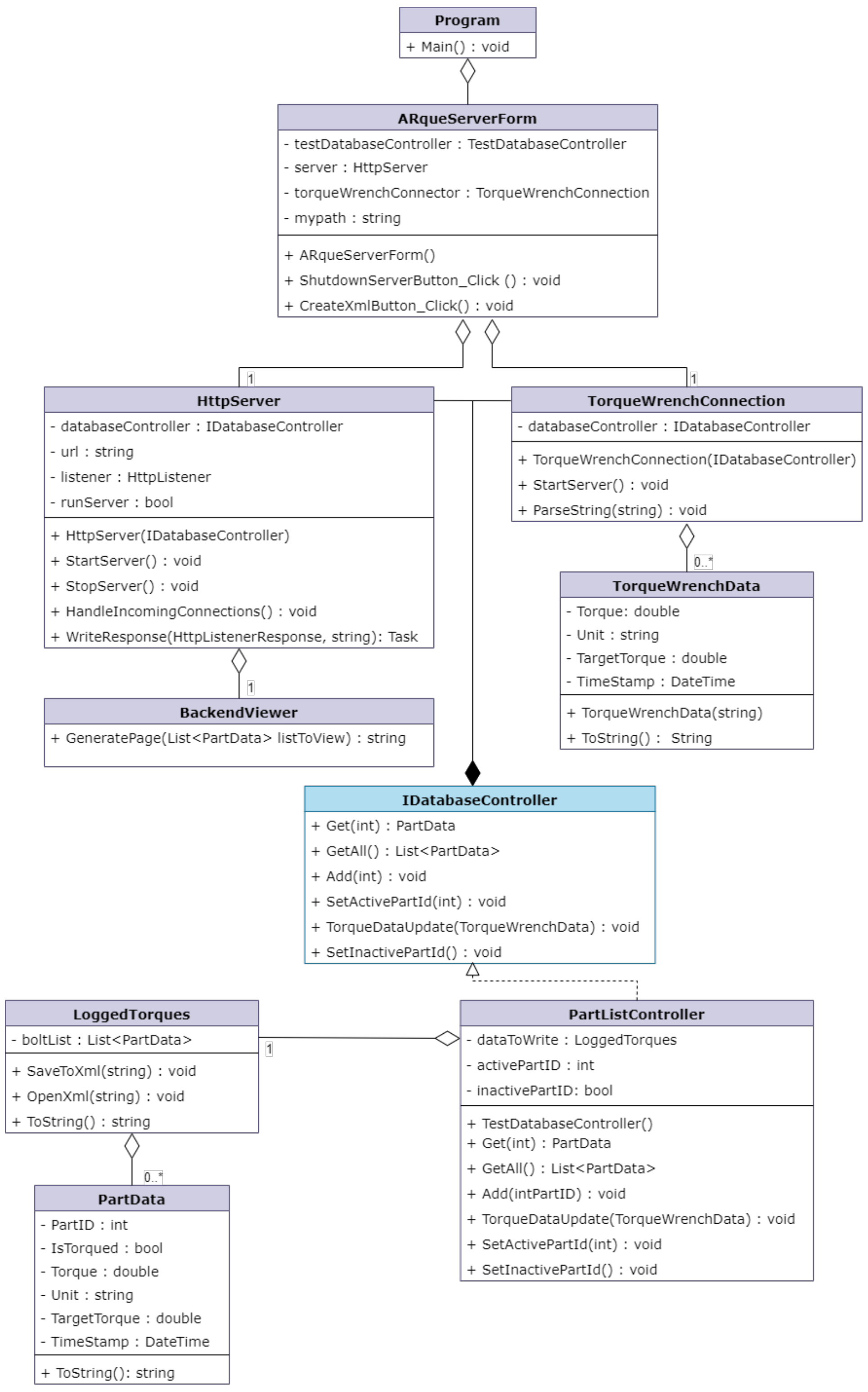

Program
+ Main() : void
ARqueServerForm
- testDatabaseController : TestDatabaseController
- server : HttpServer
- torqueWrenchConnector : TorqueWrenchConnection
- mypath : string
+ ARqueServerForm()
+ ShutdownServerButton_Click () : void
+ CreateXmlButton_Click() : void
1
1
HttpServer
- databaseController : IDatabaseController
- url : string
- listener : HttpListener
- runServer : bool
+ HttpServer(IDatabaseController)
+ StartServer() : void
+ StopServer() : void
+ HandleIncomingConnections() : void
+ WriteResponse(HttpListenerResponse, string): Task
TorqueWrenchConnection
- databaseController : IDatabaseController
+ TorqueWrenchConnection(IDatabaseController)
+ StartServer() : void
+ ParseString(string) : void
0..*
TorqueWrenchData
- Torque: double
- Unit : string
- TargetTorque : double
- TimeStamp : DateTime
+ TorqueWrenchData(string)
+ ToString() : String
1
BackendViewer
+ GeneratePage(List<PartData> listToView) : string
IDatabaseController
+ Get(int) : PartData
+ GetAll() : List<PartData>
+ Add(int) : void
+ SetActivePartId(int) : void
+ TorqueDataUpdate(TorqueWrenchData) : void
+ SetInactivePartId() : void
LoggedTorques
- boltList : List<PartData>
+ SaveToXml(string) : void
+ OpenXml(string) : void
+ ToString() : string
1
PartListController
- dataToWrite : LoggedTorques
- activePartID : int
- inactivePartID: bool
+ TestDatabaseController()
+ Get(int) : PartData
+ GetAll() : List<PartData>
+ Add(intPartID) : void
+ TorqueDataUpdate(TorqueWrenchData) : void
+ SetActivePartId(int) : void
+ SetInactivePartId() : void
0..*
PartData
- PartID : int
- IsTorqued : bool
- Torque : double
- Unit : string
- TargetTorque : double
- TimeStamp : DateTime
+ ToString(): string

Figure 12: Class diagram of the server

Program and ARqueServerForm

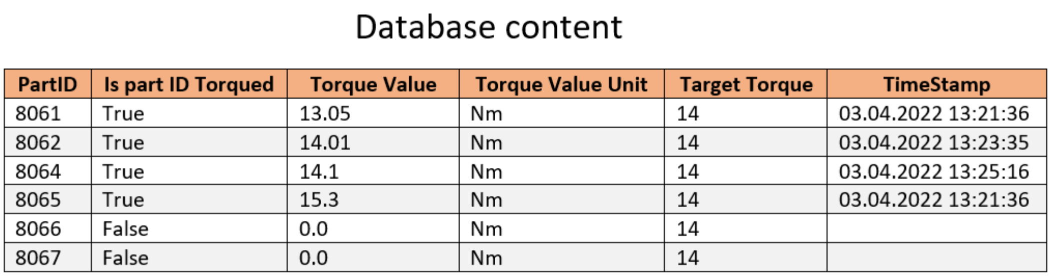
The program class has the main function which starts the ARqueServerForm class. The form class is a user interface which has the function to create and open the XML file with the torqued parts. It also contains a shutdown button to shutdown the server. The relationship between the program and the ARqueServerForm, is an aggregation. The program has an instance of the ARqueServerForm class.

HttpServer class

The HttpServer class provides the connection to the HoloLens through the HTTP methods POST and GET. When the server program begins to run, the HttpServer class is created and begins to listen for client connections. When a connection is made, the task HandleIncomingConnection will be called to see if a GET or a POST type method has been requested. If it is a GET request, HandleIncomingConnection will check if the URL path contains "/partData/" followed by an integer. If that integer is a valid part ID, the server will fetch the PartData object from the IDataBaseController, and respond to the client with a string formatted PartData object in the HTTP message body.

If the request is a POST request, HandleIncomingConnection will check if the part ID is present in IDataBaseController, if not a new PartData object will be added to IDataBaseController with default values. In both cases the part ID will be set to active, meaning that the next torque value package coming though the Named Pipes connection, will be connected to the active part ID on the server. This class is aggregated to the ARqueServerform class, since the ARqueServerform has an instance of the HttpSever created in the ARqueServerform constructor. The BackendViewer class' method GeneratePage() will be called if the GET request does not contain an URL that is defined in the server. The GeneratePage() method takes a list of PartData objects as an argument, and returns a string containing HTML formatted code. The HTML code visualizes all the parts and their data in a table, as shown in table 11.

Table 11: A representation of the data that the BackendViewer class function GeneratePage() returns


Database content

PartID	Is part ID Torqued	Torque Value	Torque Value Unit	Target Torque	TimeStamp
8061	True	13.05	Nm	14	03.04.2022 13:21:36
8062	True	14.01	Nm	14	03.04.2022 13:23:35
8064	True	14.1	Nm	14	03.04.2022 13:25:16
8065	True	15.3	Nm	14	03.04.2022 13:21:36
8066	False	0.0	Nm	14	
8067	False	0.0	Nm	14	

In listing 1, a part of the httpServer class is shown. The function StartServer() uses the HttpListener(), from the namespace System.Net, to listen for clients, which connects though the URL. In order to let other devices connect to the server the URL has an asterix, so the clients can fill this with the IP address that the server is hosted on. The function is asynchronous to prevent it from blocking.

```
public async void StartServer()
        {
            // Create a Http server and start listening for incoming connections
            listener = new HttpListener();
            listener.Prefixes.Add("http://*:8000/");
            try
            {
                listener.Start();
            }
            catch (HttpListenerException ex)
            {
                MessageBox.Show("Administrator rights are required to start a server! " + ex.Message,
                    "Unable to start",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
                return;
            }
            Console.WriteLine("Listening for connections on {0}", url);

            // Handle requests
            await HandleIncomingConnections();

            // Close the listener
            listener.Close();
        }
```

Listing 1: HttpServer StartServer()

If there is a connection the HandelingIncommingConnections() will be called. The method is async since it is a task to prevent it from blocking other functions. Listing 2 shows the essence of the function, that it uses a switch case, for the supported HTTP request POST and GET. The code shows a respond if a client sends a POST request with the URL path ending with "/shutdown", or if the client sends a GET request with the absolute path "/hello".

```
public async Task HandleIncomingConnections()
{
  // While a user hasn't visited the `shutdown` url, keep on
   handling requests
  while (runServer)
  {
    // Will wait here until we hear from a connection
    HttpListenerContext ctx = await listener.GetContextAsync();

    // Peel out the requests and response objects
    HttpListenerRequest request = ctx.Request;
    HttpListenerResponse response = ctx.Response;

    switch (request.HttpMethod)
    {
      case "POST":
        // If `shutdown` url requested w/ POST, then shutdown
   the server after serving the page
        if (request.Url.AbsolutePath == "/shutdown")
        {
          Console.WriteLine("Shutdown requested");
          await WriteResponse(response, "Server shutdown
   requested.!");

          runServer = false;
          continue;
        }

    ............................

```

```
        //GET request method request and retrives data from a
    resource in the server.
        case "GET":

          //await parseGet(response, request);
          if (request.Url.AbsolutePath == "/hello")
          {
            await WriteResponse(response, "Hello Hello!");
            continue;
          }

      ...........................

      }

      await WriteResponse(response, "invalid request");
    }
}
```

Listing 2: HttpServer HandleIncomingConnections()

TorqueWrenchConnection

Like the HttpServer class, the TorqueWrenchConnection class handles the data coming from a client. The client is the torque wrench application which sends data packages though a Named Pipe. The TorqueWrenchData class is a helper class which consists of variables that the data packages from the torque wrench sends over the Named Pipe connection. method ParseString() splits the string in an agreed format, so a new TorqueWrenchData object can be created from the string received.

In listing 3, a part of the StartServer() method for the TorqueWrenchConnection class is shown. The variable server is created of the NamedPipeServerStream which is from the namespace System.IO.Pipes. It will then wait for connections, and attempted to read string data by a new line character. This string is then parsed into a TorqueWrenchData object in the ParseString method.

```

          public void StartServer()
          {
              Task.Factory.StartNew(() =>
              {
                var server = new NamedPipeServerStream(@"arque_pipe");
                server.WaitForConnection();
                StreamReader reader = new StreamReader(server);
                Console.WriteLine("pipe connection made");
                while (true)
                {
                    string line = reader.ReadLine();
                    if (!string.IsNullOrEmpty(line))
                    {
                        ParseString(line);
                    }
                }
            });
        }
```

Listing 3: TorqueWrenchConnection to connect to client through Named Pipe

IdatabaseController and PartListController

The IDatabaseController is an interface, so the HttpServer and TorqueWrench-Connection do not need to know the exact implementation details of the database controller. PartListController is a class that inherits from the interface, implementing a simple list to simulate a database. A connection to a real database can be easily integrated into the code by implementing a new variant of the IdatabaseController interface, without needing to change the surrounding code.

The IdatabaseController and the PartListController is the heart of the server since they connect the data coming from the HttpServer connection and the TorqueWrench-Connection. Since the PartListController inherits from the IdatabaseController interface, the class can use the methods defined in the interface rather than PartList-Controller. To update a PartData object an active part ID must be set from the request from the HttpServer. Then once the TorqueWrenchConnection receives new data, this information will used to update the PartData object with the active part ID. LoggedTorques implements a SaveToFile function this converts the LoggedTorqued class of an XML representation of the list. The purpose is that the file can be reopened by the program, so the session can continue. This class is kept external to IDatabaseController, so saving to file is independent from how the data is stored internally in the system.

The code in listing 4, shows the PartData class, which structures the data that will be stored for each bolt in the system. In addition to an ID, each bolt also has a flag to indicate whether it has been torqued, an actual torque value, the unit for torque value, and the time stamp. The class itself, and each data member is tagged with a DataContract attribute or DataMember attribute, which comes from the namespace System.Runtime.Serialization. These allow the class to be automatically serialized or deserialized, into either JSON for sending via the HTTP server, or into an XML format, which is used to save the list of PartData objects to an XML file.

```
    [DataContract]
    public class PartData
    {
        [DataMember, XmlAttribute]
        public int partID;

        [DataMember, XmlAttribute]
        public bool isTorqued = false;

        [DataMember, XmlAttribute]
        public double torque;

        [DataMember, XmlAttribute]
        public string unit;

        [DataMember, XmlAttribute]
        public double targetTorque;

        [DataMember, XmlAttribute]
        public DateTime timeStamp;
    }
```

Listing 4: Part of the PartData class

7 NorTronic torque wrench

This project uses the NorTronic digital torque wrench from Norbar to capture torque data. The torque wrench has a digital display, equipped with dual color OLED displays, which can be seen in fig. 13. The display is used to display torque values while measuring torque and can give off warnings when the applied torque is higher than the torque target. The torque wrench is also equipped with different buttons, allowing the user to save measured values and navigate between them. It is also possible to define a torque target and other variables such as 'snug'. The wrench has an accuracy of $\pm$ 2% over the full measurement range. For small values this can be neglected, but for larger values, especially for large equipment installed for a long duration of time, the value has to be more precise. [6].

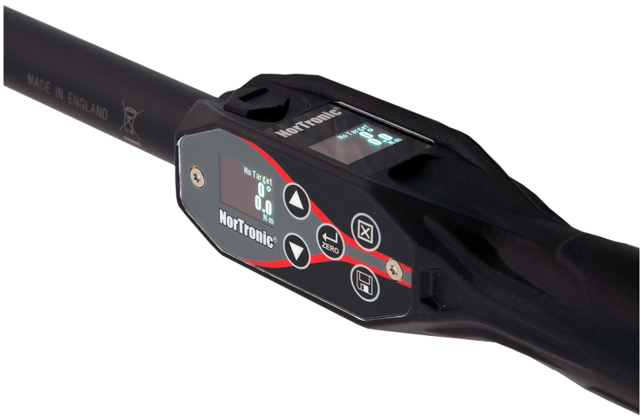
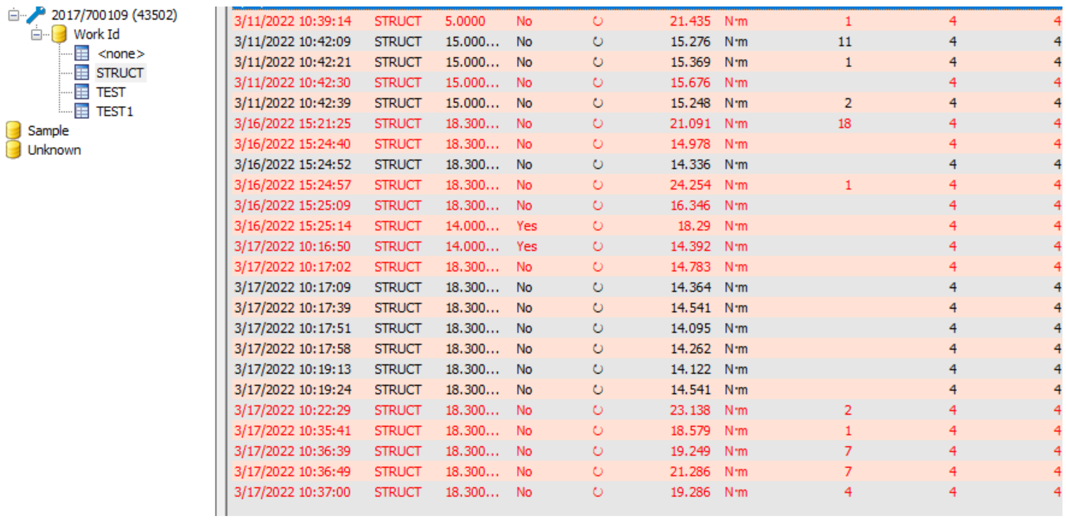


Figure 13: NorTronic torque wrench from Norbar

Figure 14: RF transceiver from Microchip technology

The wrench comes with a wireless RF receiver shown in fig. 14. The data transfer is done over RF at 868 MHz [7]. The RF transceiver is ultra low power, in order to meet European regulation. The receiver makes it possible to connect the torque wrench to Norbars integrated program for data storage, called Torque Data System (TDS). This program displays torque data captured by the torque wrench and saves it to an SQLite database. An example of logged values in TDS can be seen in fig. 15.



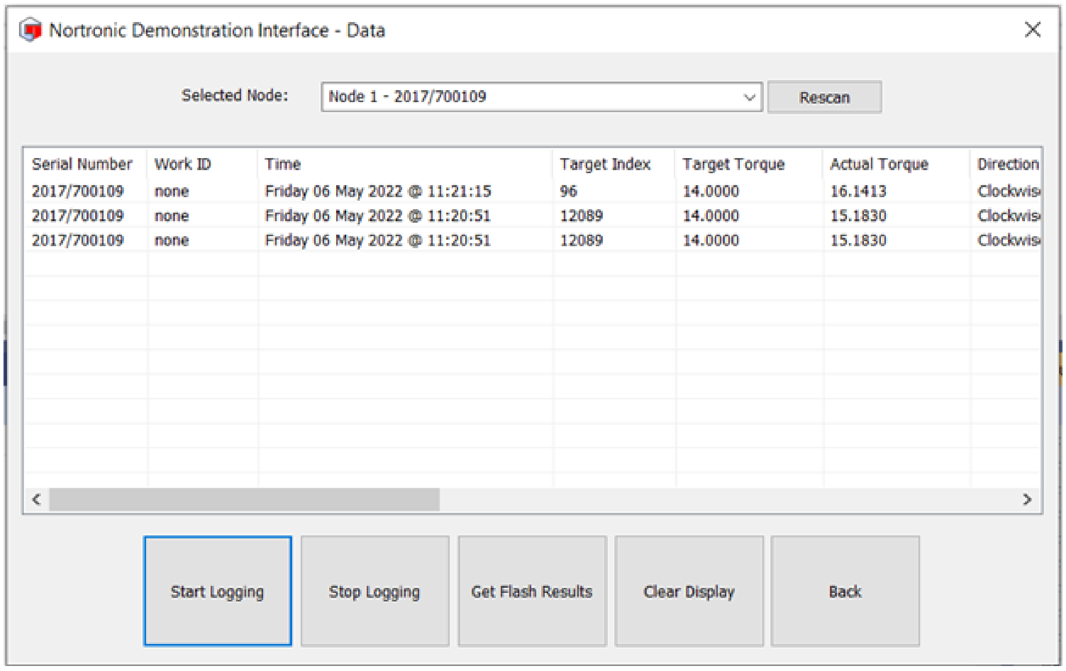
3/11/2022 10:39:14	STRUCT	5.0000	No	↻	21.435	N·m	1	4	4
3/11/2022 10:42:09	STRUCT	15.000...	No	↻	15.276	N·m	11	4	4
3/11/2022 10:42:21	STRUCT	15.000...	No	↻	15.369	N·m	1	4	4
3/11/2022 10:42:30	STRUCT	15.000...	No	↻	15.676	N·m		4	4
3/11/2022 10:42:39	STRUCT	15.000...	No	↻	15.248	N·m	2	4	4
3/16/2022 15:21:25	STRUCT	18.300...	No	↻	21.091	N·m	18	4	4
3/16/2022 15:24:40	STRUCT	18.300...	No	↻	14.978	N·m		4	4
3/16/2022 15:24:52	STRUCT	18.300...	No	↻	14.336	N·m		4	4
3/16/2022 15:24:57	STRUCT	18.300...	No	↻	24.254	N·m	1	4	4
3/16/2022 15:25:09	STRUCT	18.300...	No	↻	16.346	N·m		4	4
3/16/2022 15:25:14	STRUCT	14.000...	Yes	↻	18.29	N·m		4	4
3/17/2022 10:16:50	STRUCT	14.000...	Yes	↻	14.392	N·m		4	4
3/17/2022 10:17:02	STRUCT	18.300...	No	↻	14.783	N·m		4	4
3/17/2022 10:17:09	STRUCT	18.300...	No	↻	14.364	N·m		4	4
3/17/2022 10:17:39	STRUCT	18.300...	No	↻	14.541	N·m		4	4
3/17/2022 10:17:51	STRUCT	18.300...	No	↻	14.095	N·m		4	4
3/17/2022 10:17:58	STRUCT	18.300...	No	↻	14.262	N·m		4	4
3/17/2022 10:19:13	STRUCT	18.300...	No	↻	14.122	N·m		4	4
3/17/2022 10:19:24	STRUCT	18.300...	No	↻	14.541	N·m		4	4
3/17/2022 10:22:29	STRUCT	18.300...	No	↻	23.138	N·m	2	4	4
3/17/2022 10:35:41	STRUCT	18.300...	No	↻	18.579	N·m	1	4	4
3/17/2022 10:36:39	STRUCT	18.300...	No	↻	19.249	N·m	7	4	4
3/17/2022 10:36:49	STRUCT	18.300...	No	↻	21.286	N·m	7	4	4
3/17/2022 10:37:00	STRUCT	18.300...	No	↻	19.286	N·m	4	4	4

Figure 15: Logged values retrieved by TDS

7.1 Data Retrieval and Communication

Part of the requirements in this project was to capture torque values from a digital torque wrench, as well as sending these values to an external device for data storage. This was to make it possible for an application to retrieve the data and display it in the HoloLens, as well as connecting the data to a unique part ID. At the beginning of our project we planned to use American Standard Code for Information Interchange (ASCII) commands, provided in the operators handbook, to retrieve data. However, as the torque wrench we were given was of an earlier model, ASCII communication had not yet been integrated. This meant we had to look at other options for data retrieval. First, we decided to look at possibilities using the TDS program. In order for the torque data to be sent to an external device, the data would have to be retrieved from the SQLite database, containing all the logged torque values. Different test scripts were created which would retrieve and send data in various ways. Either by converting the database to an Comma Separated Values (CSV) file or by printing the values in a terminal window. However, the outcome was not satisfactory. The goal was also to achieve real time logging, which proved to be a challenge when collecting data from the SQLite database.

The group contacted Norbar, the manufacturer of the NorTronic torque wrench, as well as VASI, the Norwegian Norbar distributor to portray the problem of the missing ASCII communication, and by the beginning of April we were sent a "workaround" code from Norbar. This was a comprehensive project file which consisted of several thousand lines of code. The project created an application which would retrieve logged data from the torque wrench in real time, and display them on a graphical user interface as seen in fig. 16. Although the functionality was similar to the TDS program, being given the source code to the application, we were able to modify and further develop the code to implement our wanted functions. The application was modified to send the retrieved values to the ARque server, which distributes the data by making it available for the HoloLens application, as well as storing the data in an XML format. Once the server receives the measured torque values, the HoloLens is able to retrieve them and display them in the glasses, along with the appropriate part ID.


Nortronic Demonstration Interface - Data

Selected Node: Node 1 - 2017/700109

Serial Number	Work ID	Time	Target Index	Target Torque	Actual Torque	Direction
2017/700109	none	Friday 06 May 2022 @ 11:21:15	96	14.0000	16.1413	Clockwis
2017/700109	none	Friday 06 May 2022 @ 11:20:51	12089	14.0000	15.1830	Clockwis
2017/700109	none	Friday 06 May 2022 @ 11:20:51	12089	14.0000	15.1830	Clockwis

Figure 16: Graphical user interface in work around program from Norbar

In order to develop the functions we wanted, we first needed to dive into the code and understand all the functions it had. Norbar provided us with a document which explained several of the functions concerned with torque wrench connection, logging and retrieval. The application first establishes a connection with the torque wrench by using the yellow Radio Frequency (RF) dongle mentioned previously. Once a torque wrench is connected, its serial number and node name will show in the 'selected node' box, as can be seen in fig. 16. After a connection is established, the user will be able to retrieve values from the torque wrench by entering logging mode. This is done by clicking the 'start logging' button. Once in logging mode the measured torque will be displayed in the programs GUI, as seen in fig. 16. This is were the code has been modified, so that once logging mode is entered, the program establishes a connection to a server via Named Pipes communication and sends the measured torque values to the server, as well as displaying it on the application GUI. Each time the user saves a torque measurement using the torque wrench it will be sent to the server, until the user exits logging mode where the connection will be broken. Before creating a Named Pipe client to send values to the server, we tested the retrieval of values by sending them to a text file. For each logging session, the text file would be overwritten with new values, but it gave us a way of testing that the correct values were retrieved and sent by the Norbar program. Even though we have established a connection to the server, the values are still being sent to a text file to help when troubleshooting. At the beginning some of the sent values weren't received by the server, reading the text file helped us see that the correct values were in fact being sent, but an error had recurred with the server. Listing 5 is an excerpt from where the Norbar code has been modified to send values to a text file for testing, as well as creating a string with values to send to the Named Pipe server.

```

  //Prints values in a textfile called "textfile.txt"
  TestFile << "Torque: " << dTorque << "\n"
  << "Target Torque: " << dTorqueTarget << "\n"
  << "Snug: " << dSnug << "\n"
  << "Angle: " << nAngle << "\n";
  TestFile.close();

  m_IDC_LIST1.UpdateWindow();

//Sends string with values to the named pipe server when torque
    data is retrieved
  if (new_torque != dTorque) {
    string result_string =
    "Torque; " + to_string(dTorque)
    + ", TargetTorque; " + to_string(dTorqueTarget)
```

```
        + ", Timestamp; \n";

        const char* message = result_string.c_str();
        send(message);
    }
    else {
        string result_string = "No new value\n";
        const char* message = result_string.c_str();
        send(message);
    }
    new_torque = dTorque;
}
```

Listing 5: Function to send values in Norbar code

7.2 Interprocess Communication using Named Pipes

A Named Pipe is a one way or duplex form of communication between a pipe server and one or more pipe clients [8]. Named pipes can be accessed by any process, making it an easy form of communication between related or unrelated processes. To distribute the retrieved torque values, we decided to use a server written in C#. This server is able to communicate via HTTP requests, but because the Norbar code is written in C++, we found it easier to implement Named Pipe communication for Inter Process Communication (IPC).

Other methods that were researched was the use of sockets, Common Language infrastructure (CLI) and Common Language Runtime (CLR). TCP sockets handles IPC like the named pipes, but we estimated that it would be too time consuming to program the C++ sockets in a robust way. The CLI would wrap the torque wrench application and merge it in the server application[9]. To do so we would have to continue with the new project for the torque wrench application, but we estimated that it would have taken to long time to finish. The named pipes solution was the simplest way of getting the functions that we needed to prove the functionality of the system. This was done by first creating a test setup with a simple Named Pipe client and server to ensure the communication worked, before integrating a client in the Norbar code to communicate with the C# server. Listing 6 contains an excerpt from the Norbar code with the functions to create a named pipe client and establish a connection with the server, as well as the function used to send strings containing data.

```
BOOL result;
HANDLE pipe;

//Function to create client and connect to server
int connection(){
    wcout << "Connecting to pipe..." << endl;
  pipe = CreateFile(
    L"\\\\.\\pipe\\arque_pipe",
    GENERIC_READ | GENERIC_WRITE,
    FILE_SHARE_READ | FILE_SHARE_WRITE,
    NULL,
    OPEN_EXISTING,
    FILE_ATTRIBUTE_NORMAL,
    NULL
  );

  if (pipe == INVALID_HANDLE_VALUE) {
    wcout << "Failed to connect to pipe.\nError: " <<
   GetLastError() << endl;
    return 1;
  }
}
//Function to send string of values to server
void send(const char* message) {
  wcout << "Sending data to pipe..." << endl;
  DWORD numBytesWritten = 0;
  DWORD lengthofdata = static_cast<DWORD>(strlen(message) *
   sizeof(char));

  result = WriteFile(
    pipe,
    message,
    lengthofdata,
    &numBytesWritten,
    NULL
  );

  if (result) {
    wcout << "Number of bytes sent: " << numBytesWritten <<
   endl;
  }
  else {
    wcout << "Failed to send data."
    << GetLastError() << endl;
  }
}
//function to close pipe handle
int close() {
  CloseHandle(pipe);
  wcout << "Done." << endl;
  return 0;
  }
```

Listing 6: Functions to communicate with a Named Pipe server

8 Augmented Reality

Augmented Reality is a growing technology and involves overlaying virtual elements onto a real world environment to enhance the user experience. It is also closely related to mixed reality, which merges real and virtual worlds to produce new environments and visualisations. Well known applications like Instagram and Snapchat use AR in their filters to add facial features and information. It is also used in Google Maps for navigation and online shopping as a 'try before you buy' option. For comprehensive use in enterprises and institutions, like in fig. 17 [10] and military training [11], AR glasses and headsets might prove to be more relevant.

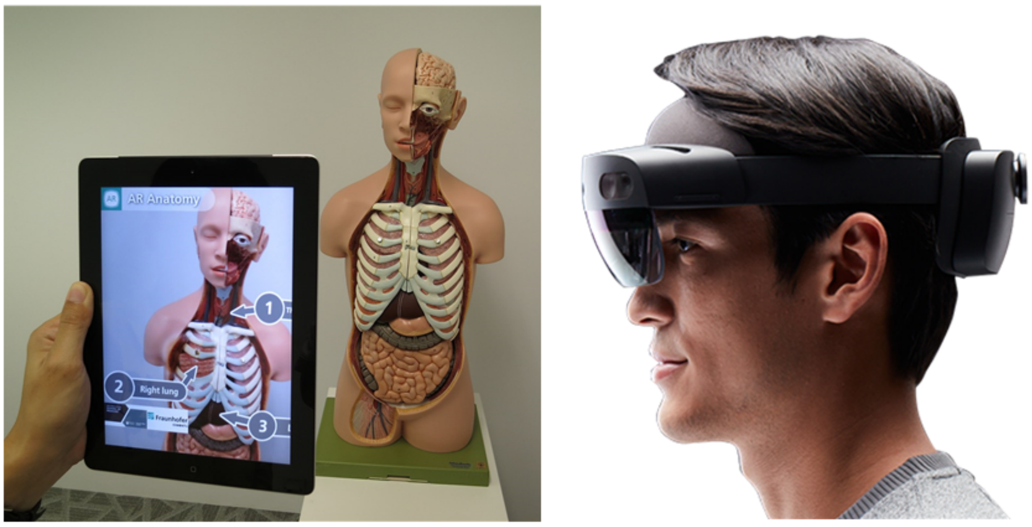


Figure 17: AR used for education (Wikimedia) and HoloLens 2 (Microsoft)

Holograms

Virtual objects that are rendered in the AR headset are called holograms. Holograms can be static or dynamic, and will appear as a 3D object in the real world that one can walk around to see from all angles. A hologram is an advanced form of photography where the motive is three dimensional. Just like with motion pictures, a hologram must be rendered at a rate of greater than 24 frames per second to create the visual experience of motion. A hologram with motion is referred to as a dynamic hologram [12]. The holographic equivalent of a pixel is called a polygon. A polygon is any shape that can be drawn on a paper in one stroke where the start and stop position of the stroke is the same, and the line is not crossed over by itself. Many polygons together is what makes the hologram.

8.1 HoloLens 2

The HoloLens is an industrial product supposed to be customisable for several fields, such as installations, which made it fitting for the ARque project. The HoloLens uses spatial sensors to map the users environment, and has special trackers focusing on the hands of the user. Controllers can be used to interact with the holograms, but the hand trackers are reliable enough to make the experience rely solely on the users own hands, making it possible for our application to run without the need for extra equipment.

Development framework

The framework used is .NET 5. This is a unified platform developed by Microsoft to build any type of .NET application with a single base class library, and provides us with the additional features needed to build and develop an application running on the HoloLens.

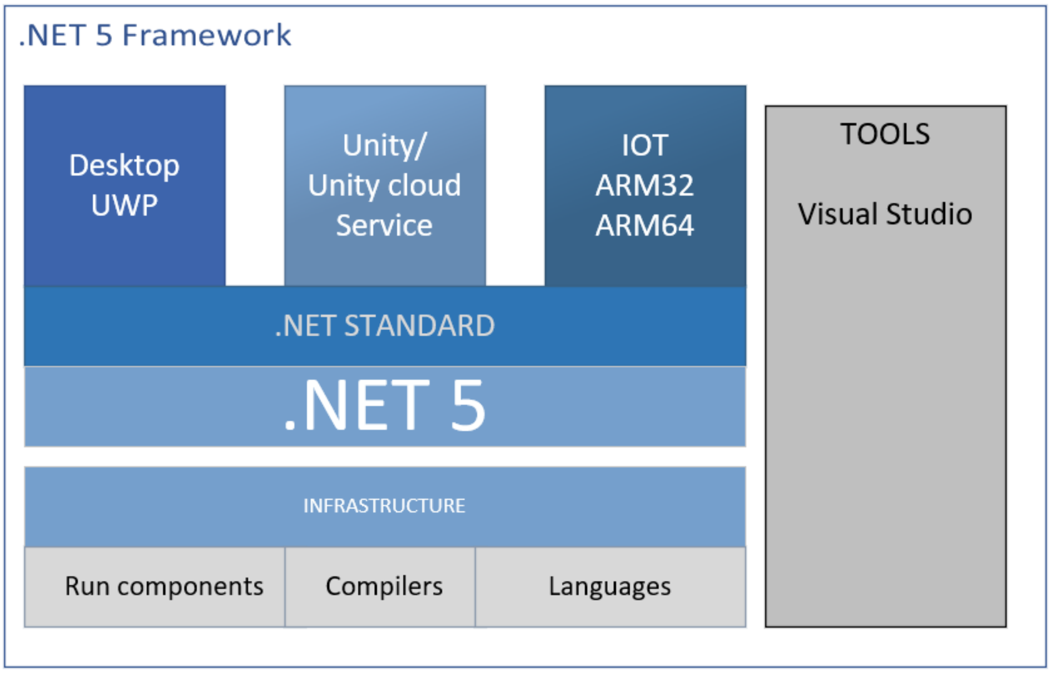


Figure 18: Framework for HoloLens devlopment

The .NET 5 Framework contains a lot of accessories in a wide scope of software development [13] . Some of the technologies included in the .NET 5 Framework that we use to develop our AR application are shown in fig. 18. The .NET library must be included to be able to build the application. The parts of the .NET framework allow user interaction and user interface in the mixed reality experience. Universal Windows Platform (UWP) is a platform that runs on all devices that has a Windows Operating System (OS). UWP has the same API on all devices running Windows OS. The HoloLens runs a modified version of Windows 10 called Windows Holographic, and falls under the UWP umbrella. The the HoloLens is an ARM64 Internet Of Things (IOT) device.

8.2 The ARque application

The flowchart seen in fig. 19 shows the application activation process when the user loads the ARque application. The first view the user will see is a menu where the user can click on a start QR scan button. When the user has clicked the button to activate the QR scanning the application will create an instance of a QRtracker object which will be responsible for detecting QR codes. When the application detects a QR code the assembly handling communications with the server will be overlaid in a known location relative to the origin of the QR code. This means we can place the holographic overlay over a physical object correctly every time.

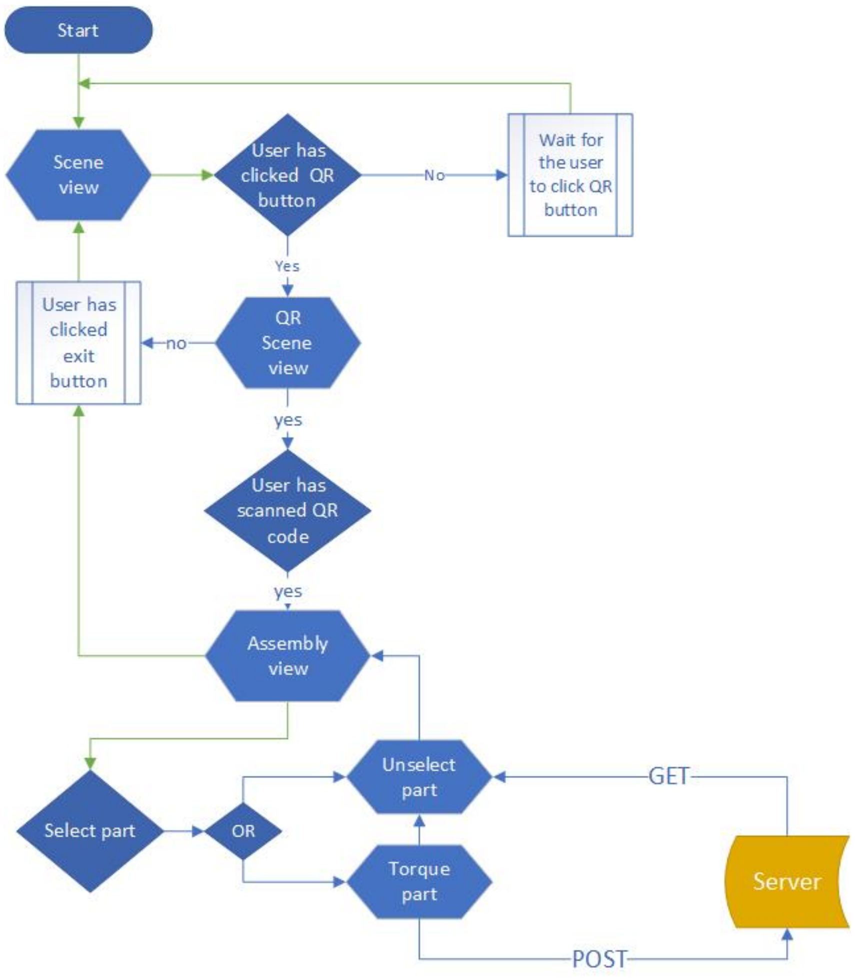


Figure 19: Flowchart of the activation process when loading the application

8.2.1 Communication with the Server

Part of all B requirements, see section 4, was to connect the two subsystem HoloLens and NorTronic together. So, one of the main features to the AR application that we developed was a communication line between the server and the HoloLens.

The first method that we tested was using Web Real-Time-Communication (WebRTC). WebRTC is used to transfer data packets and video transmissions, so the client can communicate inside the web and is supported by native apps and web browsers. WebRTC uses POST and GET but this method did not quite fit this task when we were going to retrieve torque values in clear text from the server. So, we decided under the development for test if we can get the torque value to use HTTP requests from the HoloLens to server.

Testing the application in Visual Studio

In order to communicate with the server we sat up a test application in a Visual Studio UWP project that has a user interface in 2D Extensible Application Markup Language (XAML) before it was going over to Unity. Listing 7 shows the test code that first succeeded to communicate with the server. The code is a part of the Windows.Web.Http namespaces [14] and the classes that are used is

- HttpResponseMessage
- ReadAsStringAsync
- HttpClient
- TryParseAdd
- GetAsync.

```
private async void Get_Event(object sender, RoutedEventArgs e)
{

    HttpClient httpClient = new HttpClient();
    var headers = httpClient.DefaultRequestHeaders;
    //error handlers
    if (statment)
    {
        throw new Exception(statment)
    }

    Uri requestUri = new Uri("http:/ localhost :8000/ ");
    HttpResponseMessage httpResponse = new HttpResponseMessage
   ();
    string httpResponseBody = "";
    try
    {
        httpResponse = await httpClient.GetAsync(new Uri("http
://localhost:8000/"));
        httpResponse.EnsureSuccessStatusCode();
        httpResponseBody = await httpResponse.Content.
ReadAsStringAsync();
    }
    catch (Exception ex)
    {
    httpResponseBody = "Error: " + ex.HResult.ToString("X") + "
Message: " + ex.Message;
    }
}
```

Listing 7: Event function that sends HTTP request from HoloLens to server

Listing 7 is the setup of events that are waiting for an input to send the requests to the server.

Importing a UWP project from Visual Studio into Unity

Figure 18 shows the relationship between the .NET Framework and Unity. From the UWP project we could not use the Unity API scripts without adding the reference to the DLL files that Unity used. And to do that we need to add the references to the dependency so that we can use UnityEngine DLL to use UnityEngine namespace.

HTTP Connection Integratet in Unity Enviorment

Listing 8 contains the code we used in Unity to connect to the server.

```
using UnityEngine.UI;
using UnityEngine;
namespace httpClientConnect
{
    public class PostRequest : MonoBehaviour
    {
        InputField Output;
        void Start()
        {
            Output = GameObject.Find("Output").GetComponent<
InputField>();
            GameObject.Find("PostButton").GetComponent<Button
>().onClick.AddLi       stener(PostData);
12          }
13          public void PostData() => StartCoroutine(
    PostWithErorhandler());
14          IEnumerator PostWithErorhandler()
15          {
16              try
17              {throw statement;}
18              catch(Exception e)
19              { catch statement;}
20              yield return null;
21          }
22      }}
```

Listing 8: Using the UnityEngine with HTTP request

Listing 8 shows an event that is initialized when the application is running, shown on line 10. MonoBehaviour was used to access UnityEngine so we could use the start function that is a part of the UnityEngine DLL. Namespace httpClientConnect was used to wrap up the classes so we could access the member inside of it using UnityEngine and UnityEngine.UI outside of Unity.

Line 13 shows StartCoroutine() which is a method built into the Unity scripting API. This method returns a yield statement. Inside PstWithErrorHandler we have entered different statements ranging from error handler to sending a POST request to the server. This feature is triggered by one of the buttons.

8.2.2 QR Tracking

There are several methods to place a hologram on a surface. They can be called on by tags, exist in the room by default or be placed using buttons and commands. We used tags to call and place holograms in this project.

There are also multiple types of tags that can be placed on a physical object so it can be scanned by a camera to extract information. A common tag type is the Quick Response (QR) code. QR codes are commonly used to be scanned by cellphones to direct the phone to a website. QR codes are used for this purpose because they are created for conveying information. In our case, the data we want to receive using a tag/code is location, angular orientation and distance. Preferably as fast and precise as possible. The team considered AprilTags for object tracking, but they are not officially supported by the HoloLens. Using AprilTags we would also have to use the web camera built into the HoloLens. The web camera is only available to one process at a time, so if the user has to take photos or use video, the camera would be unavailable. Because of the possibility of tying QR codes to other functions and events we decided this was the preferable type of tag to use in this project. The sensors used for spatial mapping also supports QR tracking, so the web camera is available to other processes during runtime. We based our application on the ability to scan a QR code to place a holographic object in the three dimensional space as seen in fig. 20, and afterwards track the code if the object should move.

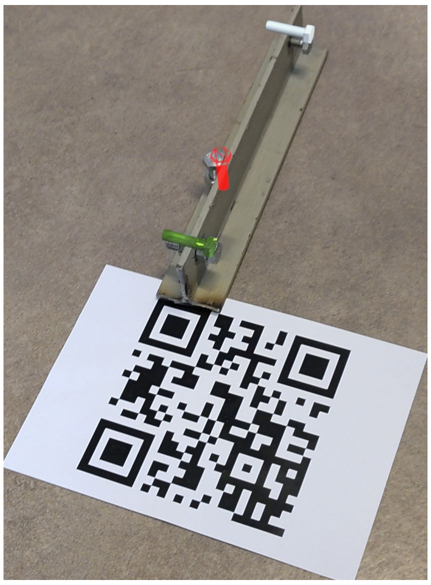


Figure 20: Correctly placed holographic bolts over the physical test bench

Enabling QR tracking

To enable QR tracking on HoloLens 2 we needed access to the system level drivers responsible for detecting the QR codes. The spatial cameras used to map the user environment has built in support for QR-tracking, and is the resource we used to detect codes and project the holograms correctly. Using the spatial cameras we needed a translator mirroring the Left Hand Side (LHS) matrixes used to project the hologram, to HoloLens' Right Hand Side (RHS) detection matrixes [15].

To start we used the QRCode sample project provided by Microsoft, and further developed the project from there. The QRTracking project from Microsoft's own repository is the foundation of all the projects we identified during the project. The repository consists of scripts accessing, reading, storing and processing the codes read on the OS level in the HoloLens. From the Git repository the important scripts are

- QRCodesManager.cs
- QRCode.cs
- QRCodesSetup.cs
- QRCodesVisualizer.cs
- SpatialGraphCoordinateSystem.cs

QRCode.cs is the code that manages displaying the QR data on a Hologram inside the application. The script is attached to a QRCode object. The QRCode object is the 3D model that is attached to the physical QR code and serves as the visual representation of the tracker. Having the QR tracking implemented, the next step was to create a scene which contained the test bench model we built. Changing the model itself was easy enough, but we needed to make sure we could be sure where the model was placed every time the QR code was scanned. We used the origin in the coordinate system as our basis, as covering the QR code with the physical object would make the HoloLens unable to scan it properly. The result can be seen in fig. 20. The QRCodesManager.cs script is responsible for the plugin itself and does a check to make sure the device supports QR tracking as a feature. When access has been granted and all the checks completed, the script is responsible for handling the events through callback, starting and stopping the tracker, and maintains a local list of QR codes scanned. As mentioned before the actual detection and tracking of the codes happens at the system level. QRCodesSetup.cs calls the QRCodesManager to start the tracking functionality in the application. The setup script is attached to a game object that acts like a trigger for the actual tracking to start. The QRCodesSetup is called in the QRCodesManager game object. The spatial coordinate system script is where the interpretation of the real world location happens and it also handles the translation into the spatial understanding used by the HoloLens. There is a webcam built into the glasses, but this camera can only be accessed by a single application at a time, so if we used this camera for QR tracking the camera would be unavailable to other processes.

Prefabs is a very special type of game object in Unity. A prefab is a user defined game object that is locked in a current state, but can be added to one or multiple scenes where the parameters can be changed, but attached scripts and other prerequisites remain locked. A prefab is therefore the preferred type to be placed using QR scanning. Aditionally a prefab is stored in the files system and can be added to scenes, whereas normal game objects only exist in the scene. The prefab that is to be displayed must contain the scripts QRcode.cs and SpatialGraphNodeTracker.cs to be positioned relative to the physical QR code and tracked. These scripts must be attached at the root of the prefab folder only.

8.2.3 Application user interface with POST and GET

For the user of the application, the user interface looks something like the setup in fig. 21. It was the first design created before it was going to be combined with the QR application.

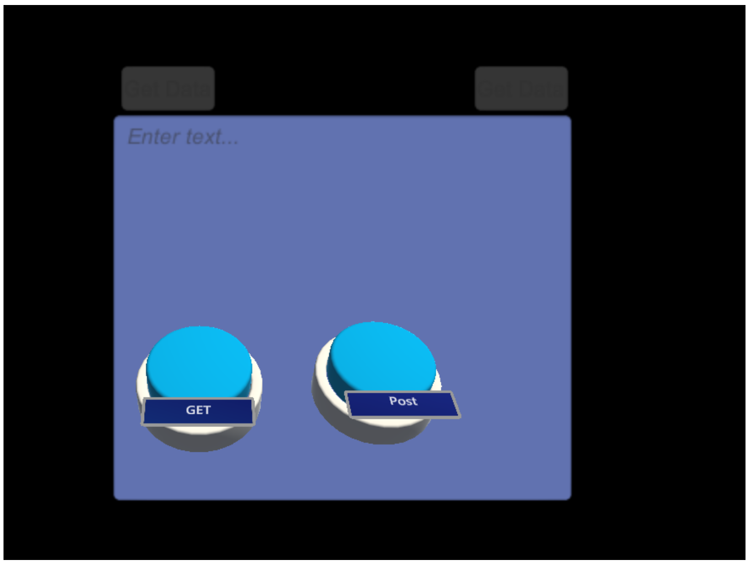


Figure 21: Application GUI GET and POST buttons and output textbox

In fig. 21 the two buttons POST and GET sends requests to the server when the buttons are pressed. The response from the server is written in the canvas as shown in fig. 22.

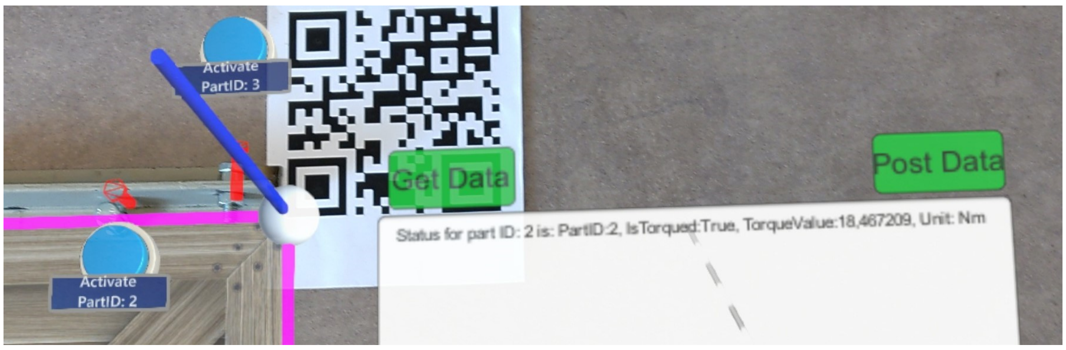
The canvas also acts as an output for the error handler to be able to take on different error messages. One example of an error we have used is if the IP to the server does not match the one that the user of the application enters. If so there will be thrown an "invalid url" exception in the canvas. This also ensures that the application does not crash every time there is a user error. One needs to add the CanvasUtility.cs script for the output to be shown. CanvasUtility.cs is part of Microsoft.MixedReality.Toolkit.Input.Utilities that allows the user to use the API of HoloLens.

8.2.4 Activate and deactivate buttons

When the activate button shown in fig. 22 is pressed, a couple of events are triggered. These are:

- Hide the part itself
- Display a sphere where the physical part is
- Hide the activate button
- Display the deactivate button of that specific part.

The deactivate button is displayed some distance from the actual part, to prohibit unwanted triggering while actually working on that part. Additionally there are some events called to let the server know what is going on. Each button has the script Interactable.cs attached to itself, which has an Event property that handles events triggered by the users interactions. The activate button uses the PostRequest.cs script, accessible from the Canvas.



Status for part ID: 2 is: PartID:2, IsTorqued:True, TorqueValue:18,467209, Unit: Nm

Figure 22: ARque system seen through HoloLens showing activation buttons for part 2 and 3, holographic bolts and output on canvas after deactivating bolt 2

Both the PostRequest.cs and GetRequest.cs scripts are attached to the Canvas, and accessed by the bolt objects. This is to provide better user experience, by getting instant visual feedback that confirms the actions that are done in the application. Unwanted events triggered by connection malfunctions with the server are likely to happen if not. A POST request is used to send information to the server. In this case, we want to send which part ID the server should write the next value to. When a specific part is activated, a dim sphere is displayed around the physical object, to indicate that this is the part to be torqued. No other holographic model is rendered over the active part to avoid any visual disturbance while working. After activating a specific part, we torque it, and the torque value is sent to the server when saving it on the torque wrench. After torquing the selected part, we press the deactivate button to see the new value.

The deactivate button reverses the four actions listed in the bullets above. To get confirmation about the actions we have done, the Interaction Event uses the GetRequest.cs script to acquire the desired information about the torqued part, and display it in the Canvas fig. 22 in the application. The information is of course also stored for each part separately in the database, and is easily checked in the web browser GUI.

8.3 Further development

Scenes

During development we had trouble with the QR tracking breaking after loading and reloading scenes, with the solution being to reset the HoloLens. We identified the fault as a Null Reference Exeption, created by the instantiated instance of the QR watcher. The QRCodesManager in the QR project is instantiated as a singleton, which should on paper be available to other processes across scenes. However we did not manage to successfully recover the instance after scene changes, or rewrite the scripts to avoid using a Singleton. Instead we focused on a workaround that proved to have other benefits.

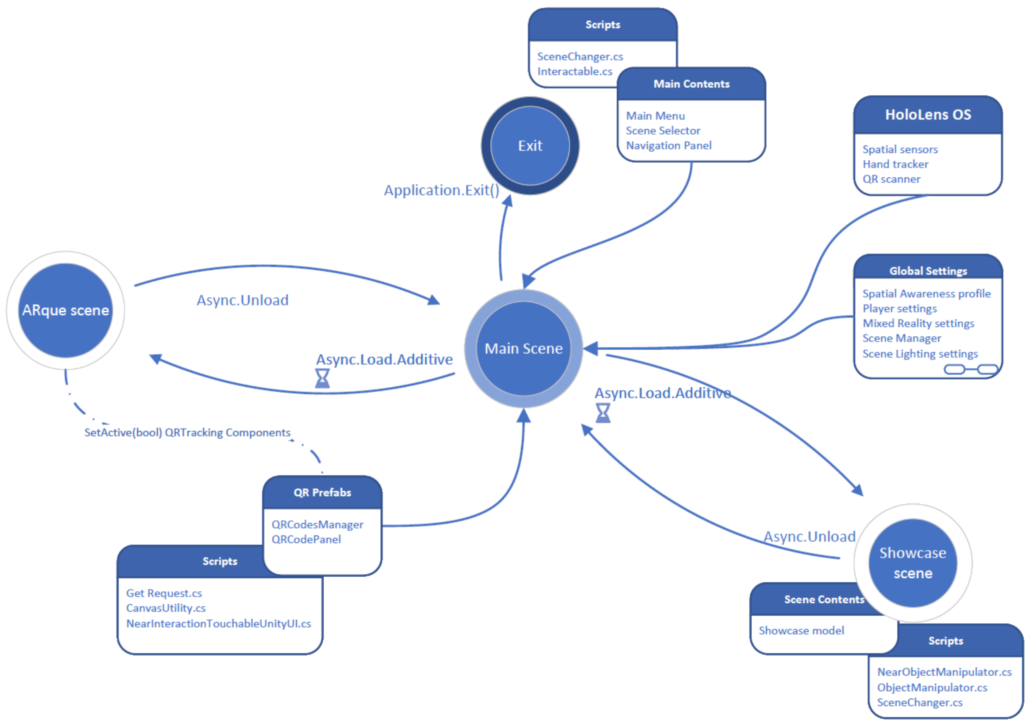


Figure 23: Scene hierarchy

Since we are using Unity as the main development tool for the AR application, we made a setup inspired by game development. Our final setup shown in fig. 23 consists of a main scene where all the AR settings and features are placed. This is to minimize the possibility of introducing duplicate players, which would make the user unable to interact with holograms. In the same scene we have placed the QR tracking capabilities, and the associated Prefabs.

When we load a new scene the scenes are always added in addition to the main scene, so the player can exist in the same space at all times, with different models shown in the room depending on what scene is loaded. During development we wanted to make a foundation that is easily customisable. With the current setup a developer can easily tailor the application to their needs, as long as the Main Scene is kept intact. As long as the scene is added to the SceneChanger script shown in Listing 9, and included in the build the user should be able to load and unload the assets as they want.

```
public void ReturnHome()
    {
        GetActive();
        if (SetActive)
        {
            ToggleActive();
        }
        SceneManager.UnloadSceneAsync(scene);
    }
    public void ARqueScene()
    {
        SceneManager.LoadSceneAsync("ARque_1.3", LoadSceneMode.
    Additive);
        ToggleActive();
    }
```

Listing 9: Example from the scene changer

The example from Listing 9 shows two important functions for this setup to work. The ToggleActive function is tied to the QR tracker, and only activates/deactivates the visibility of it according to what scene is loaded. The ReturnHome function is a global return to Main scene used in all scenes. If the QRtracker is active, it is then disabled after before returning. This solution made us able to create an environment with smooth transitions and good customisability in our opinion.

9 Proposed future work

This project was aimed at implementing a specific scenario into an AR application to see if the specific use case could be streamlined with the use of AR. As discussed earlier the technology is rapidly advancing, and it can be difficult to identify real world use cases for AR outside niche products and "showcase" applications.

9.1 AR application

The use of QR codes in this project is mainly limited to object placement, but the benefits of QR codes could be much bigger. Differentiating the codes and using events and triggers in Unity would let the user choose scenes based on QR codes. This means one could have multiple scenes with different models, and using known QR codes to choose what model should appear. We decided to keep using the instance of the QR tracking for stability, and relied on correct object placement with the player as a reference for the other scenes without the use of QR codes. When already using QR codes, we suggest looking into developing a web app to store information about models, scenes and everything else. This primarily does two things:

- The developer does not need to deploy to the HoloLens for testing.
- The scenes can easily be shared between devices.
- Secure connection

This can also make the general development a lot easier, and could make the application easier to implement across device families such as phones, tablets and VR headsets.

9.2 Data capture / torque wrench

The connection between the NorTronic wrench and the program receiving the data has been unstable at times. There seems to be a problem in the program when it comes to detecting the tool's presence, and it requires restarting the application to work. This should be rectified when moving forwards.

The data transfer between the HoloLens and the server should also be encrypted, since the project focuses on demonstrating the functionality, security was not prioritized. Also, a simple implementation is to instead of having a manually parse string function between the applications, JSON deserialize/serialize provided by .NET should added. For the server a method that opens a XML file would be a next step, so the operator can continue on a previous session.

For reliability there should be an option to create and store data captured from the wrench locally on the HoloLens. This could be useful for creating a quick LookUp Table (LUT) within the application for visualizing progress on what has been worked on or not. The server handling the communication between the wrench and the HoloLens should also get a database for managing the information. At the same time the server / torque wrench application should be able to run as a stand alone solution so the operator does not have to use the HoloLens for documenting the work being done.

Another feature to add is the ability for the HoloLens to send the target torque value along with the active part ID. The server can then store this and send it down to the torque wrench. This ensures that the operator tightens the bolt to the correct value, without needing to manually set the torque wrench setting. As it is now the target value is set on the torque wrench, and is sent to the server.

10 Conclusion

The goal of our project was to develop a system that would enable communication between a digital torque wrench and a HoloLens application. This was mainly to ensure data storage of critical values and to minimize the need of bringing extra documentation into industrial environments.

During the project we tested and created several ways of retrieving data from the torque wrench. We also created different applications in the HoloLens to display and update data, as well as a server to connect the two subsystems and distribute information between the two.

We therefore conclude that the system we have developed fulfills the goal of the assignment. We have created a solution that conveys information from the real world to the virtual world in real time. The project has proved that there is a possibility of creating environments combining real world operations with AR technology, given the proper use of development resources. Using AR to ease specific tasks is something that will be useful in the future, as long as the use is seamless and does not interfere with the tasks themselves. We believe that AR can make a strong contribution towards streamlining certain operations.

References

[1] Fortune Business Insights. *Augmented reality market size, share and COVID-19 analysis*. 2022. URL: https://www.fortunebusinessinsights.com/augmented-reality-ar-market-102553.

[2] Ken Schwaber and Jeff Sutherland. *The definitive guide to Scrum: the rules of the game*. White paper. Creative Commons, Nov. 2020.

[3] Brian. Vanderjack. *The agile edge : managing projects effectively using agile Scrum*. First edition. Business Expert Press, 2015.

[4] Mark Michaelis. *What is a server?* TechTarget, 18 Aug 2021. URL: https://www.techtarget.com/whatis/definition/server.

[5] Li Li and Wu Chou. "Design and describe REST API without violating REST: a Petri net based approach". In: *2011 IEEE international conference on web services*. Washington, DC, USA, 4-9 July 2011, pp. 508–515.

[6] Ioannis Giannopoulos, Damian Doroni-Dawes, Kyriakos Kourousis, and Mehdi Yasaee. "Effects of bolt torque tightening on the strength and fatigue life of airframe FRP Laminate Bolted Joints". In: *Composites Part B: Engineering* vol 125 (May 2017), pp. 19–26.

[7] Norbar Torque Tools Ltd. *Operators manual 34399 Nortronic, pp. 11*. Version 3. 2014.

[8] Steven White, Kent Sharkey, David Coulter, Drew Batchelor, Mike Jacobs, and Michael Satran. *Named Pipes*. 2021. URL: https://docs.microsoft.com/en-us/windows/win32/ipc/named-pipes.

[9] James S. Miller and Susann Ragsdale. *The common language infrastructure annotated standard*. Addison-Wesley Professional, 2004.

[10] Carolien Kamphuis, Esther Barsom, Marlies Schijven, and Christoph Noor. "Augmented reality in medical education?" In: *Perspectives on medical education* (Jan. 25, 2014), pp. 304–306.

[11] Ahmet Ejder, Isa Haskologglu, and Ali Kemal Düsgün. "Augmented reality in military training and education". In: *Defense resources management in the 21st century* (Nov. 15, 2012), pp. 1–2.

[12] Andrew Bañasa and Jesper Glückstad. "Holo-GPC: holographic generalized phase contrast". In: *ScienceDirect* 392.1 (2017), pp. 190–195.

[13] Mark Michaelis. *.Net Reunified: Microsoft's Plans for .Net 5*. 2019. URL: https://docs.microsoft.com/en-us/archive/msdn-magazine/

2019/july/csharp-net-reunified-microsoft%5C%E2%80%99s-plans-for-net-5.

[14] Steven White, David Coulter, Mike Jacobs, Maira Wenzel, Mauricio de los Santos, Eliot Cowley, Michael Satran, and Alexander Koren. *HttpClient*. 2021. URL: https://docs.microsoft.com/en-us/windows/uwp/networking/httpclient.

[15] Qian Wen, Tyler Pride Milligan, Vinnie Tieto, and Christopher McClister. *QR code tracking Overview - mixed reality*. May 2022. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/qr-code-tracking-overview (visited on 04/28/2022).

11 Appendices

11.1 Appendix 1: Technical Project Contribution

Julie		
Technical contribution	**In the report**	**When was it done?**
I modified the workaround code from Norbar to retrieve values from the torque wrench and send them to a server. This includes researching interprocess communication and setting up a named pipes client and test server.	Nortronic torque wrench section, 7	Research and development happen mainly in the end of April and beginning of May
Code projects when researching and testing retrieval of values from the torque wrench SQLite-database.	Nortronic torque wrench section, 7.1	Week 9
Discussed and created system architecture with Marie	System Architecture section, 5	Throughout the project as diagrams were updated throughout the project, but especially week 10-12 and week 18-20.

Marie		
Technical contribution	**In the report**	**When was it done?**
I worked independently on creating the server and the matching HTTP test application.	Server section, 6, and fig. 7 shows the test application.	Research started in week 10, the project code that is used for the project was developed in beginning of April and worked on until mid of May.
Created system architecture in collaboration with Julie.	System Architecture section 5	Throughout the project. Mainly in week 10 and 19.
Researched the torque wrench API and began on a C++ project	In 7, and the code project is mentioned in a paragraph in section 7.2.	Research began in week 8. The C++ project in Week 14, when we got the demo software from Norbar.
Created 3D model of the test bench in Solid Works.	Mentioned in section 8.2.2.	week 14.

Rikard		
Technical contribution	**In the report**	**When was it done?**
I have been working on communication between HoloLens app to the server, and have worked server and created the scripts in Visual Studio and Unity for HoloLens	8.2.1	end April to mid May.
Have created Flowchart and researched .NET framework and setup.	Section 8.2	In February the framework and the Flowchart may.
Created canvas in HoloLens application that can output text from server, post and get buttons.	Section section 8.2.3	In May it was implemented in Unity Enviroment .
Collaborated with Sondre to implement communication app with QR app (Sondre/Sigurd).	section 8.2.2	May.

Sigurd		
Technical contribution	**In the report**	**When was it done?**
Collaborated with Marie and Julie on the preliminary work on the torque wrench, with most of the early time spent on the ASCII communication, and data extraction from the SQLite database. Switched to Unity development after Julie succesfully implemented the current solution.	section 7	February-March.
Reasearch, debugging and implementation of AprilTags and QR tracking together with Sondre.	section 8.2.2	Research happened in February, development in early March to the end of April.
Designed the scene hierarchy and set up the scene management and contents in the final application.	section 8.3	May

Sondre		
Technical contribution	**In the report**	**When was it done?**
Researched and debugging on QR tracking compilation together with Sigurd.	section 8.2.2	March and April
Correct placement of holograms according to the position of the QR codes, creating a GUI in the application	section 8.2.2	May
Activate and deactivate buttons with functionality, and collaboration on integrating server functionality with Rikard.	section 8.2.2 and section 8.2.4	May

11.2 Appendix 2: Epics

Epic 1	Torque wrench and data storage
Story	**As a** user of the subsystem, **I want** a wireless transfer from the torque wrench to a data storage, **So** it can be checked what torque value a part have been tightened with.
Epic ID	ARQ-182

Epic 2	HoloLens
Story	**As a** user of the subsystem, **I want** to see a virtual 3D model in the HoloLens overlapping with the physical model and pick a part of the virtual assembly, **So** the selected part are ready to be torqued.
Epic-ID	ARQ-176

Epic 3	Documentation and presentation
Story	**As a** project member of ARque, **I want** to have an overview of the documentation and presentation, **So** it is separated from the technical work.
Epic-ID	ARQ-61

Epic 4	Sprint admin
Story	**As a** project memeber of ARque, **I want** to have an overview of all administrative task to be done for each sprint, **So** it is separated from the technical work, and I am sure it gets done.
Epic-ID	ARQ-111

11.3 Appendix 3: Known issues

Table 12: List of known issues

1	Using different versions of Unity and the packages in Unity could lead to severe errors in the editor.
2	In a stable build environment in Unity we recommend postponing updates unless it is absolutely necessary.
3	The QR tracking on Hololens 2 happens on the OS level, so direct control over the stored codes is not possible through Unity. To work around this, a developer should set the program to ignore codes with a timestamp that is xxx old.
4	Unstable connection between the Norbar torque wrench and the receiving application. This problem affects the example software we were given, so it might be a bug in the DLL provided by Norbar. It may also be a hardware fault, a low level connection loss, or a bug in the Norbar application, but not in the DLL itself.
5	The bolts are currently not implemented in an object oriented way, both with regards to the HoloLens application and the server, so the application in its current state is not particularily adaptable to other assemblies.

11.4 Appendix 4: Gantt Diagram

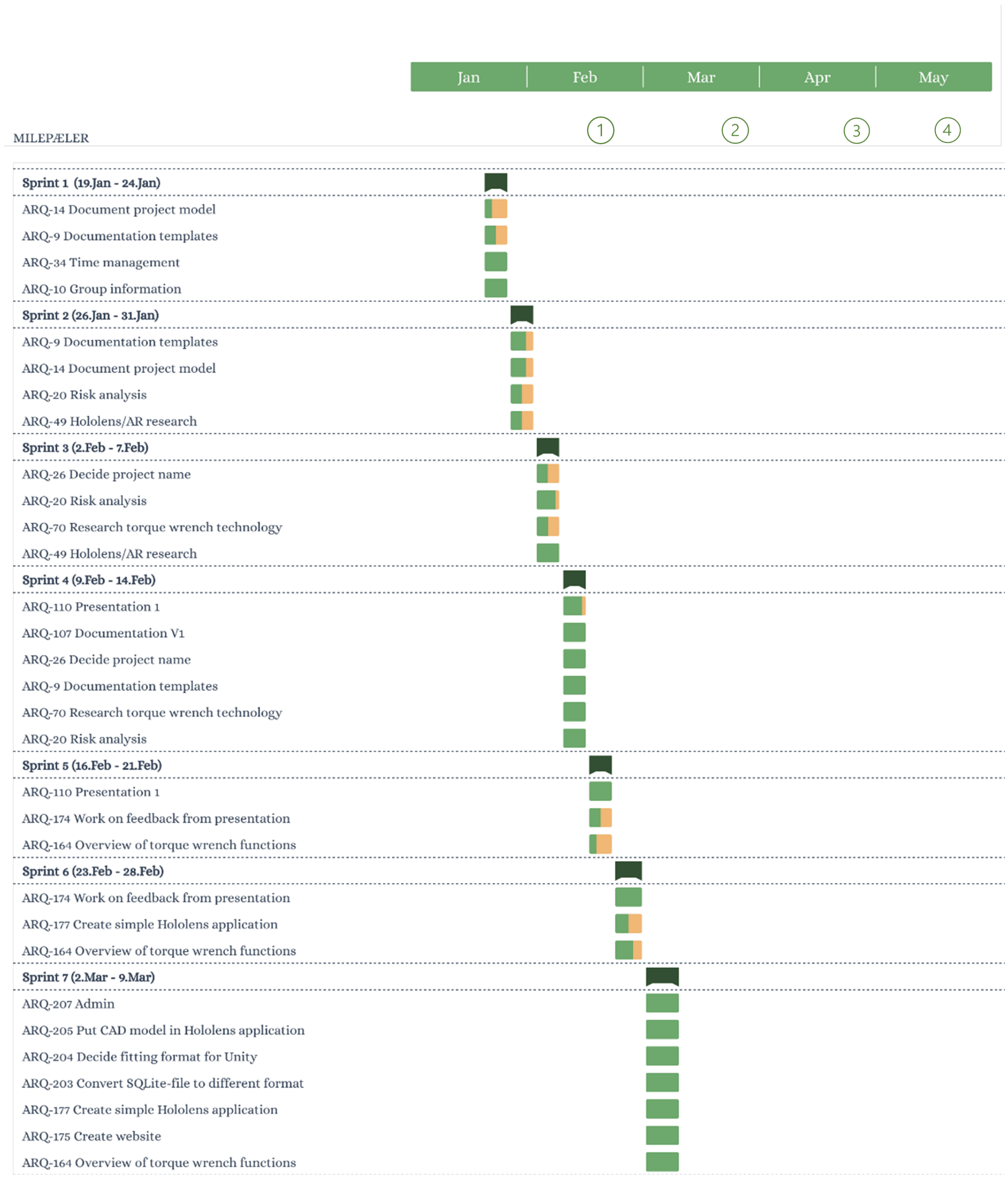


Figure 24: Gantt diagram part 1

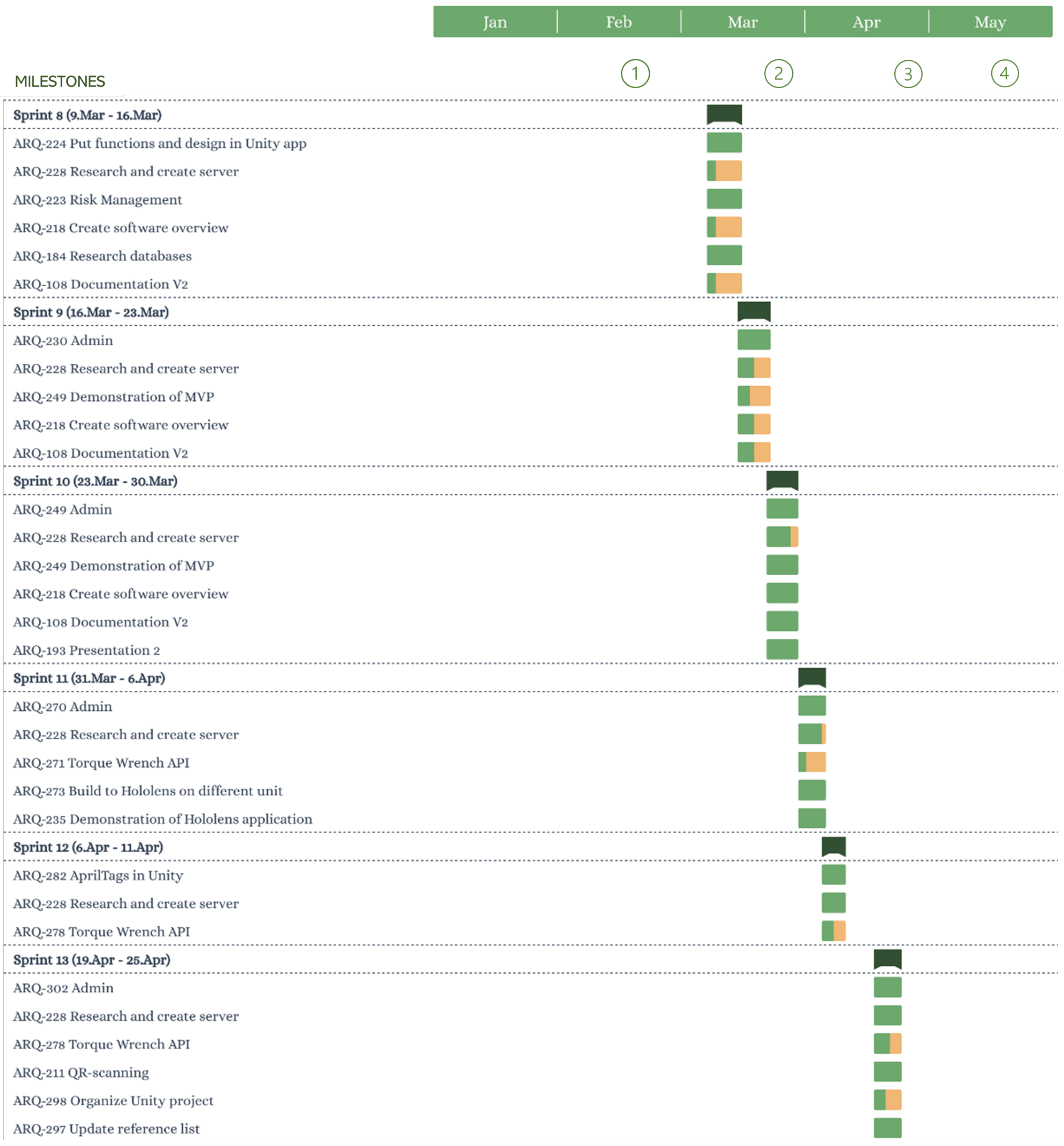


Figure 25: Gantt diagram part 2

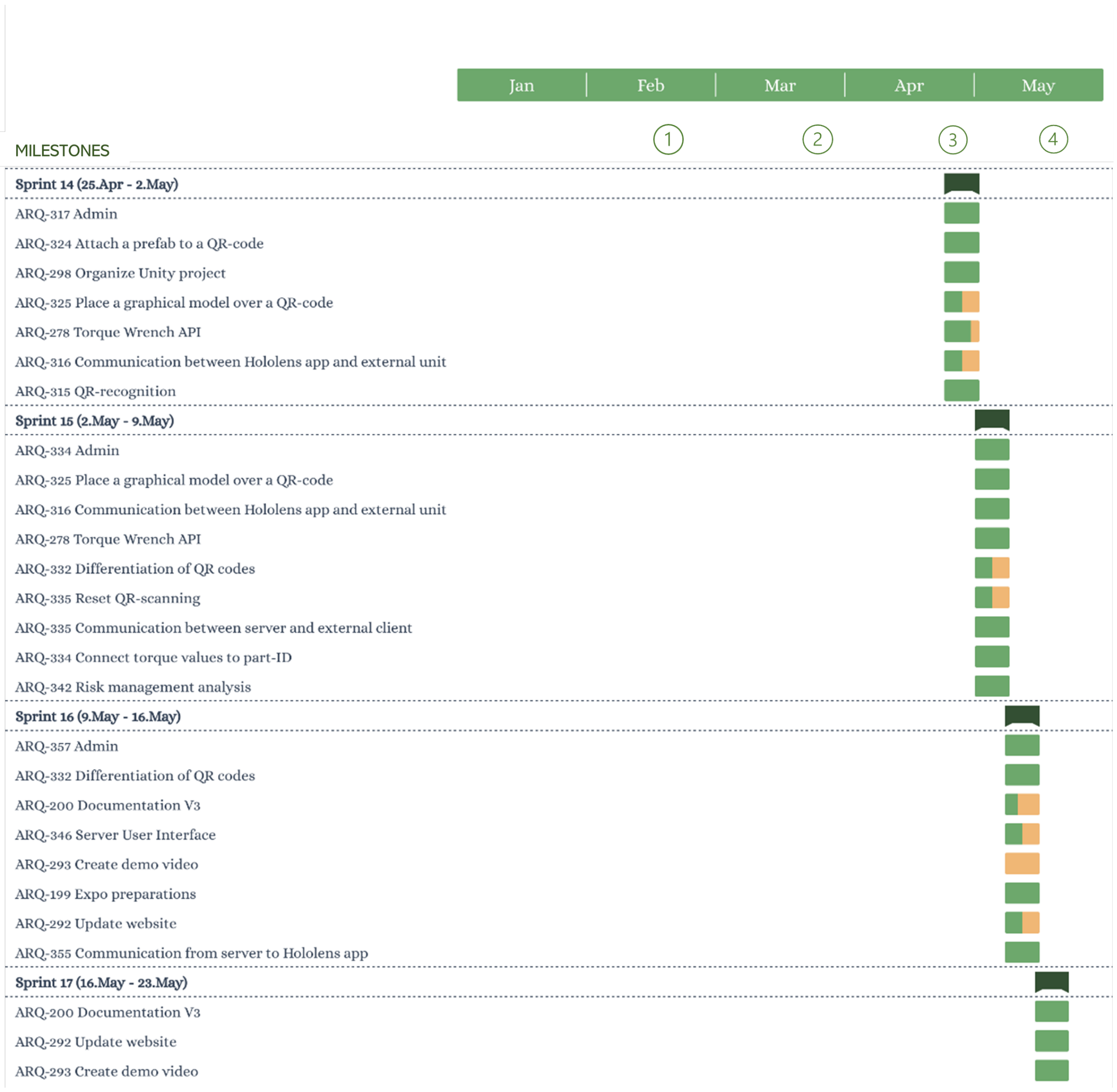
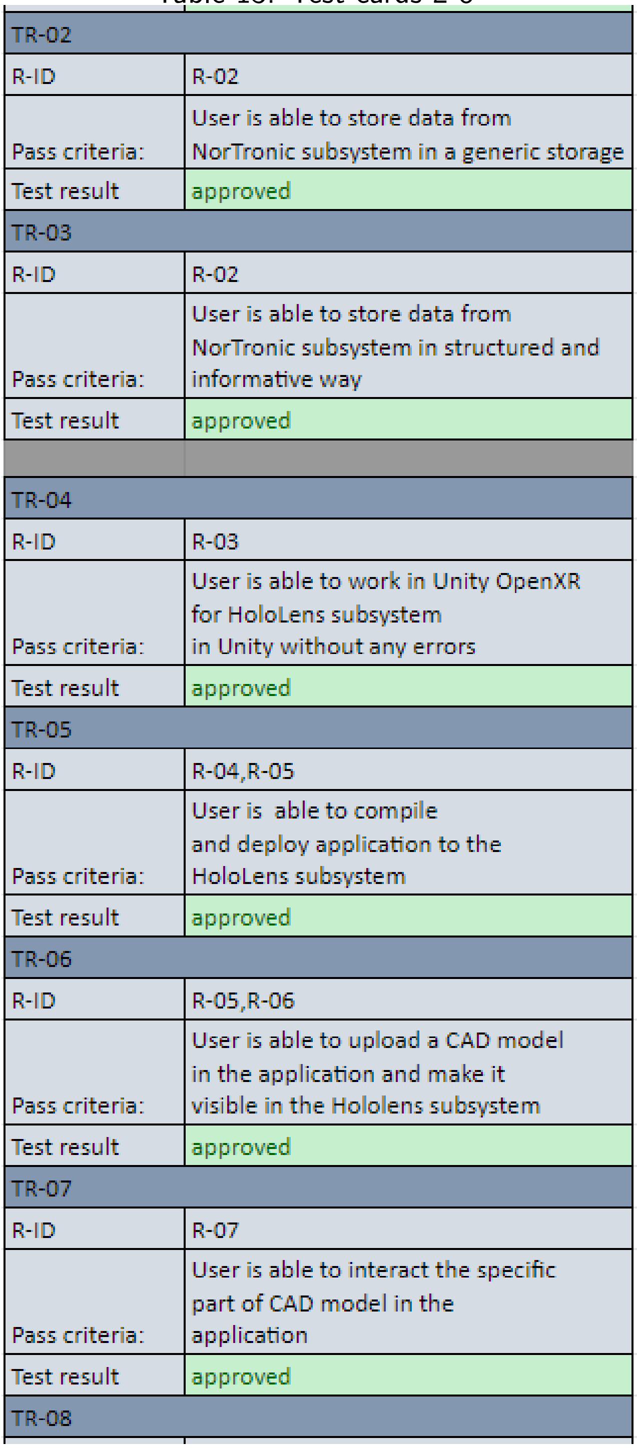


Figure 26: Gantt diagram part 3

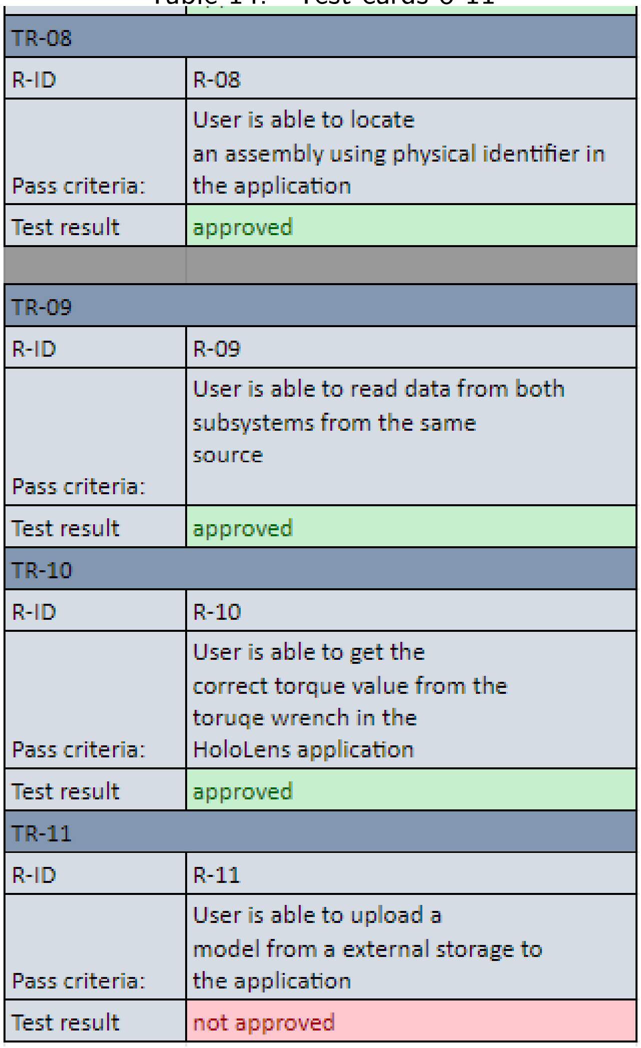
11.5 Appendix 5: Test Requirements

Table 13: Test cards 2-8



TR-02	
R-ID	R-02
Pass criteria:	User is able to store data from NorTronic subsystem in a generic storage
Test result	approved
TR-03	
R-ID	R-02
Pass criteria:	User is able to store data from NorTronic subsystem in structured and informative way
Test result	approved
TR-04	
R-ID	R-03
Pass criteria:	User is able to work in Unity OpenXR for HoloLens subsystem in Unity without any errors
Test result	approved
TR-05	
R-ID	R-04,R-05
Pass criteria:	User is able to compile and deploy application to the HoloLens subsystem
Test result	approved
TR-06	
R-ID	R-05,R-06
Pass criteria:	User is able to upload a CAD model in the application and make it visible in the Hololens subsystem
Test result	approved
TR-07	
R-ID	R-07
Pass criteria:	User is able to interact the specific part of CAD model in the application
Test result	approved
TR-08	

Table 14: Test cards 8-11



TR-08	
R-ID	R-08
Pass criteria:	User is able to locate an assembly using physical identifier in the application
Test result	approved
TR-09	
R-ID	R-09
Pass criteria:	User is able to read data from both subsystems from the same source
Test result	approved
TR-10	
R-ID	R-10
Pass criteria:	User is able to get the correct torque value from the toruqe wrench in the HoloLens application
Test result	approved
TR-11	
R-ID	R-11
Pass criteria:	User is able to upload a model from a external storage to the application
Test result	not approved