Master's thesis

Hanna Aksetøy Aalmen

Bridging the Gap

Enabling the Use of Low-cost Industry 4.0 Prototypes in Small- and Medium-sized Enterprises

Master's thesis in Engineering and ICT
Supervisor: Federico Lozano
Co-supervisor: Martin Steinert
July 2022

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU
Norwegian University of Science and Technology

TrollLABS

Hanna Aksetøy Aalmen

Bridging the Gap

Enabling the Use of Low-cost Industry 4.0 Prototypes in Small- and Medium-sized Enterprises

Master’s thesis in Engineering and ICT
Supervisor: Federico Lozano
Co-supervisor: Martin Steinert
July 2022

Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

Summary

This Master's Thesis describes the work that has been done in order to develop and implement low-cost Industry 4.0 conceptual prototypes in a small- and medium-sized enterprise. The project has been divided into two main branches which have been worked on in parallel. The first branch concerns the development of sensors which track the assembly of a wooden product by monitoring the tools used in the process. The second branch concerns the development of a user friendly and easy-to-understand software which is capable of both teaching the user to set up the system of sensors on their own, while also serving as an interface which shows the readings from the sensor prototypes in real time.

This thesis attempts to answer three research questions through sensor protoyping in parallel with a software development process. The following three research questions are addressed:

1) Can data from low-cost Industry 4.0 solutions accurately represent the activity conducted at a wood product assembly station?

2) Is a production manager with minimal experience with IoT able to set up an Arduino and interpret its data when provided instructions and pre-written code?

3) Does a low-cost Industry 4.0 system generate valuable data for a wood product manufacturer where most processes are done manually?

The results from the prototyping activities conducted are promising and have resulted in valuable insight for further improvement of the proposed system.

Oppsummering

Denne Masteroppgaven omhandler arbeidet som er blitt gjort i forbindelse med utviklingen og implementasjonen av lavkostnads Industri 4.0-prototyper i små og mellomstore bedrifter. Prosjektet har blitt delt i to seksjoner som har blitt utviklet parallelt. Den første seksjonen omhandler utviklingen av et sensorsett som er i stand til å generere data om produksjonen av et trevareprodukt ved overvåkning av verktøy som inngår i prosessen. Den andre seksjonen omhandler utviklingen av brukervennlig programvare som både kan være et læremiddel for brukeren i å sette opp de aktuelle sensorprototypene, samt at det også viser avlesninger fra sensorene til brukeren i sanntid.

Denne oppgaven forsøker å svare på tre forskningsspørsmål gjennom prototyping og programvareutvikling. Spørsmålene lyder som følger:

1) Kan data fra lavkostnads Industri 4.0-løsninger nøyaktig vise hvilke prosesser som skjer på en produksjonsbenk for treverk?

2) Er en produksjonsleder med minimal erfaring innen IoT i stand til å sette opp og tolke data fra en Arduino når han eller hun får instruksjoner og ferdig kode?

3) Kan et lavkostnads Industri 4.0-system generere verdifull data for en trevarebedrift der de fleste prosesser er manuelle?

Resultatene fra prototypingen er lovende, og har ført til verdifull innsikt som videre kan bidra til forbedringer av det foreslåtte systemet.

Preface

The degree of Industry 4.0 implementation in large enterprises is often regarded as being connected to the enterprise's competitiveness i todays market. Small- and medium-sized enterprises lag behind this development, in large parts due to lack of resources. This thesis introduces two prototypes developed in parallel which together form an attempt at enabling users in small- and medium-sized enterprises to implement Industry 4.0 prototypes in their operations. This Master's Thesis in Engineering and ICT is written for TrollLABS at the Norwegian University of Science and Technology. The work has been supervised by Federico Lozano and Martin Steinert. Prototyping has been conducted both at TrollLABS and at Nasjonalparken Næringshage in Oppdal. Testing has been conducted at Otretek AS in Oppdal.

Hanna Aksetøy Aalmen

Trondheim, July 2022

Acknowledgements

I would like to thank my supervisors Federico Lozano and Martin Steinert for their guidance throughout this year. Their insight in the field has been extremely valuable, and they have provided flexibility and support while working on this thesis. To the people at TrollLABS, thank you for your help and input. I would also like to thank Nasjonalparken Næringshage for their hospitality and allowing me to work there during my visits to Oppdal, as well as Otretek AS for allowing me to test the prototypes in their production hall and for being part of the testing.

Finally, I would like to thank Tom-Are for his endless support, and my family for cheering me on. I would not have been able to do this without you.

List of Acronyms

API: Application Programming Interface

CPS: Cyber-Physical Systems

IDE: Integrated Development Eironment

IIoT: Industrial Internet of Things

IMU: Inertial Measurement Unit

IoT: Internet of Things

IT: Information Technology

MVVM: Model-View-ViewModel

PaaS: Platform-as-a-Service

RFID: Radio Frequency Identification

SaaS: Software-as-a-Service

SME: Small- and Medium-sized Enterprise

SUS: System Usability Scale

UI: User Interface

Overview

List of Figures

List of Tables

1 | Introduction

The fourth industrial revolution or Industry 4.0 is a term characterizing the current industrial technology paradigm. Industry 4.0 proposes the vision of Smart Factories, where Industrial IoT, cloud technolgies and artificial intelligence provide an enormous competitive advantage.

Small- and medium-sized enterprises lag behind in the implementation of Industry 4.0 technologies. This thesis researches the current state of Industry 4.0 implementation in a Norwegian small- and medium-sized enterprise. Furthermore, it suggests how the current degree of implementation can be improved in accordance with the requirements of the enterprise and its employees. Prototyping for the thesis was conducted at TrollLABS and the makerspace at Nasjonalparken Næringshage in Oppdal. Testing was conducted at Otretek AS, a wood product manufacturer also situated in Oppdal, Norway.

1.1 Scope

This thesis investigates whether it is possible to enable small- and medium-sized enterprises to implement low-cost Industry 4.0 prototypes in their operations. A prerequisite being that the users are provided guidance on how to implement sensors in order to achieve this. The guidance and sensor telemetry are shown to the user in a software which has been developed for the project. The thesis therefore also covers the software development process, including some remarks on further development of the software and the possible future use in SMEs. The following three research questions will be addressed in this thesis:

1) Can data from low-cost Industry 4.0 solutions accurately represent the activity conducted at a wood product assembly station?

2) Is a production manager with minimal experience with IoT able to set up an Arduino and interpret its data when provided instructions and pre-written code?

3) Does a low-cost Industry 4.0 system generate valuable data for a wood product manufacturer where most processes are done manually?

1.2 Motivation

A large motivation for the work conducted in this thesis is aiding small- and medium-sized enterprises in upping their competitive advantage in an increasingly digital world. As these enterprises make up over sixty percent of the world's working force, their importance cannot be overlooked.

While there seems to be a trend in differentiating between Industry 4.0 solution providers and solution users in research, the prototyping conducted in this thesis attempts to bring the two closer together. By introducing the employees of a small- and medium-sized enterprise to an open-source and free-to-use software that enables them to integrate low-cost sensors, they are hopefully inspired to explore this realm on their own.

1.3 Thesis Outline

Chapter 1 introduces the reader to the motivation and scope of this thesis. Chapter 2 covers the theory which forms the basis for the project. The prototyping activities conducted in order to monitor the Assembly Station at Otretek AS are covered in chapter 3, while chapter 4 introduces the software that was developed in parallel with the sensor prototypes. Chapter 5 contains a discussion of test results and suggestions for further work, and the last chapter concludes this thesis.

2 | Theory

This chapter details the theory which forms the basis for this thesis. The first section covers the definition of Industry 4.0, followed by a section on Industry 4.0 and its impact on small- and medium-sized enterprises (SMEs). Section 2.2 covers smart retrofitting and gives an insight into technologies similar to the ones being developed in relation to this thesis. Section 2.3 concerns prototyping and prototypes, and more specifically the design methodologies relevant for this thesis. The last section concerns software development and the open-source movement.

2.1 What is Industry 4.0?

The fourth industrial revolution is often referred to as Industry 4.0, and has been a widely used term since its introduction in German manufacturing industry in 2011 (Castelo-Branco et al., 2019). The term relates to the latest big shift in industry where digitization and automation have become vital factors in gaining competitive advantage (Bartodziej, 2017). In order to achieve such an advantage, the industry has incorporated solutions such as IoT, edge computing, cloud technologies and machine learning.

The industry is steadily increasing its use of Industry 4.0 technologies. The concept of Smart Factories also serve as inspiration for shaping the vision of the new industrial era (Castelo-Branco et al., 2019, p. 2). It becomes clear that the enterprises who are holding on to the technologies of former paradigms will experience tougher competition as the development continues.

2.2 The Status of Industry 4.0 in Small- and Medium-sized Enterprises

The Confederation of Norwegian Enterprise defines small- and medium-sized enterprises (SMEs) as having 100 employees or lower. Small enterprises are defined as having 1-20 employees, while medium-sized enterprises have 21-100 employees. In 2016 employees in small- and medium-sized enterprises made up 47 percent of the Norwegian workforce (NHO, 2018). Globally the differentiations between small, medium and large enterprises vary to some extent. While there exists no common definition in the industry, a majority of enterprises and organisations report employee numbers in the 1-250 range when asked what constitutes an SME (Berisha and Pula, 2015). A 2013 study by the OECD found that SMEs make up about 90 percent of enterprises globally, also constituting as much as 63 percent of the global workforce (Munro, 2013, as cited in Berisha and Pula, 2015).

While larger enterprises have come a long way in incorporating the cyber-physical systems characterizing Industry 4.0, small- and medium-sized enterprises lag behind. One of the issues can be traced to the lack of a comprehensive business strategy for evaluating and integrating cyber-physical systems. In a 2016 study, the researchers found that four out of ten German SMEs lacked a strategy for Industry 4.0 integration, while the case for larger enterprises was two out of ten (Schröder, 2016a, p. 4).

There are severeal causes for the low degree of Industry 4.0 integration in SMEs. The lack of a business strategy for the integration of appropriate technologies might be one of the most evident manifestations of them. Some of the issues come down to allocating enough financial resources, low degrees of standardization, the relatively small scale of the operation and in general a lack of understanding of the subject (Müller, 2019).

There is a high probability that some alterations must be made to the current IT systems in the SME in order to implement Industry 4.0 solutions. According to (Schröder, 2016a) this is especially relevant in cases where the SME's existing IT products have been acquired over time, tools and machinery come from different manufacturers and there is no designated IT department or IT knowledgeable employees in the enterprise. These points might cause an aversion among managers to get started with the transition, as it does not only seem to demand investments in equipment, but also time and training. In addition, (Schröder, 2016b) found that upper management was more cautious about possible implementations than production managers.

2.3 Smart Retrofitting

In order for an enterprise to achieve the ideal Industry 4.0 concept, the Smart Factory, all machinery must be able to incorporate or adapt to Industry 4.0 technologies. The case for many enterprises today is that the machine park consists of a variety of machines, both when it comes to manufacturer and age. Some machinery or tools might qualify as legacy equipment and have been used for several decades. This, together with a lack of knowledge on how to update current machinery, can make the implementation of industrial IoT difficult and sometimes technically unfeasable

(Guerreiro et al., 2018). Through smart retrofitting, one attempts to make said machinery Industry 4.0 compatible by incorporating elements from cyber-physical systems, such as sensors.

There exist several solutions for smart retrofitting within the realm of Internet-of-Things and the more specialized Industrial Internet-of-Things. Some of these solutions come in the form of SaaS or PaaS, or Software-as-a-Service and Platform-as-a-Service respectively. These services are remotely hosted and often tend to the cloud related aspects of Industry 4.0. They are mostly subscription-based and work in cooperation with IoT components that the user has acquired or can acquire via the service. Tulip is a subscription-based PaaS enabling digital transformation in workflows. The platform lets the user analyze data from a pre-defined set of integrated machinery, and is an extensive solution for IIoT (Tulip.co, 2022). Other solutions like Thingsboard.io and OpenRemote.io are two SaaS which specialize in smart retrofitting. The user acquires the embedded systems needed and the services help with the implementation and data processing. These two services are also subscription-based, but provide an open-source version of their source code with limited functionality (ThingsBoard.io, 2022, OpenRemote.io, 2022).

2.4 Prototyping and Prototypes

Ulrich and Eppinger (2012) define prototypes as ”an approximation of the product along one or more dimensions of interest” (Ulrich and Eppinger, 2016, p. 293). The act of prototyping is the development of such an approximation. Lim et al. (2008) states that not only do prototypes serve as concepts for evaluating an idea, they also have a ”generative role in enabling designers to reflect on their design activities in exploring a design space” (Lim et al., 2008, p. 1).

How many resources an organisation is willing to invest in prototyping varies. It also varies how encouraged the use of resources for prototyping is. (Schrage, 1996) classifies organisations as either spec-driven or prototype-driven based on such factors, and states that there are connections between the organisation’s culture and willingness to innovate.

In *What do Prototypes Prototype?* by (Houde and Hill, 1997), a prototype is defined as ”any representation of a design idea, regardless of the medium”. Furthermore, they place the issues designers are faced with when creating prototypes in one or more dimensions of their proposed model. Design issues can either be in the dimension *look and feel*, where they relate to physical perception, *role* which relates to the role the artifact has in a user's life, or *implementation*, which relates to how the artifact functions. A fourth dimension is also introduced, and presents *integration* prototypes. These prototypes cover the complete user experience of an artifact, and the designers are not restricted by the scope of a dimension.

2.4.1 Design Thinking

Design thinking is a method of developing design concepts which sprung out of the d.school at Stanford University. The methodology is based on five different modes which each serve a specific purpose. These modes are shown in figure 1.

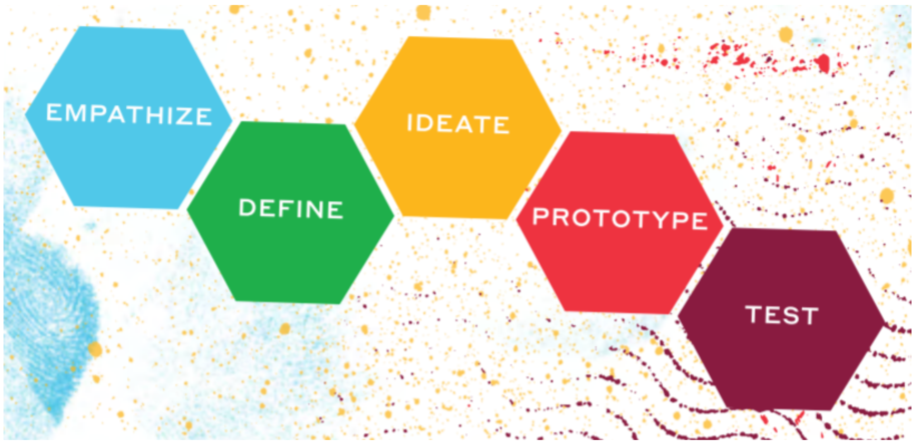


Figure 1: The different modes in the design thinking methodology (Doorley et al., 2018a, p. 2).

The five modes defined by (Doorley et al., 2018a) are *empathize*, *define*, *ideate*, *prototype* and *test*. In the *empathy* mode one should observe the user, engage with them and try to experience their point-of-view. This information will be used in the *define* mode, where it will be explored and used to define a problem space. This problem space serves as a basis for the *ideate* mode, where one should look at different designs which might solve the problems. The solution space gets defined once the ideation develops into prototypes in the *prototype* mode. The prototype is then tested in the *test* mode, and thus proves itself as viable or not.

The design thinking methodology often develops into an iterative process where the acts of needsfinding, solution exploration, prototyping and testing repeat themselves. As design thinking brings concepts from several disciplines together, it lays the grounds for "iterative learning cycles driven by rapid conceptual prototyping" (Leifer and Steinert, 2011). Working in high-performance diverse teams might therefore increase the learning outcome of the process.

While design thinking is best applied in teams, the concepts are adapted for individual use and then applied in this project. This is elaborated on in the next two chapters.

2.4.2 The Fuzzy Front-end and Wayfaring

The early stages of product development can be a challenging phase, often characterized by high levels of ambiguity. This is the Fuzzy Front-end of innovation, defined as the activities that take place before the conventional New Product and Process Development (Koen et al., 2002). In early-stage development product developers are tasked with needsfinding, defining a problem space and exploration of said space.

This process often presents itself as a cycle of divergent and convergent activities. Ideation and probing causes divergence, and the following findings lead to convergence towards more refined prototypes. The act of probing ideas and exploring the whole solution space through rapid prototyping is one of the main characteristics of the Wayfaring model. In the *Hunter-Gatherer Model* Leifer and Steinert (2012) compares the divergent activities with hunting and convergent activities with gathering. The goal is for one to hunt the next "big idea" (Steinert and Leifer, 2012). The Wayfaring Model is shown in figure 2.

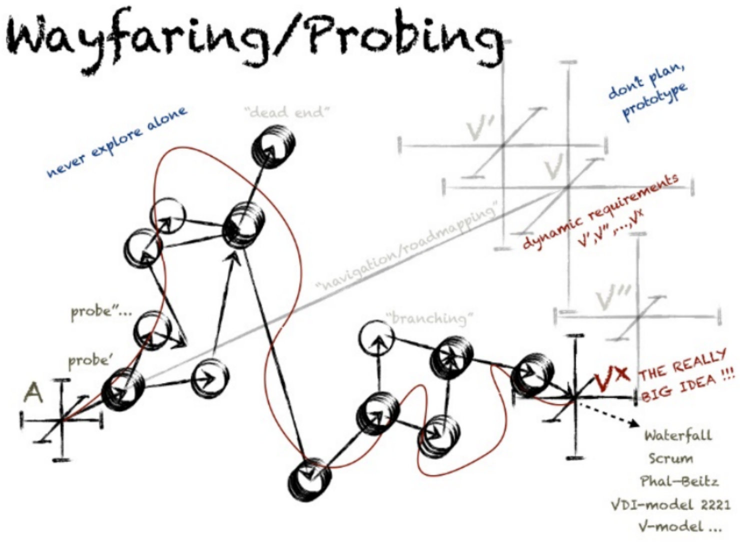


Figure 2: The Wayfaring Model by (Gerstenberg et al., 2015, p. 413)

2.5 Agile Software Development

Agile software development is a software development method which is particularly well suited for projects with changing requirements and high user-involvement. While requirements are documented at the beginning of the software development process, they are often subject to change as cycles of design, implementation and evaluation are executed, and new requirements emerge. During requirements engineering, prototypes in either physical or digital form can be developed in order to elicit system requirements. The same prototypes can be used to exlore options in the design process of the user interface (Sommerville, 2016).

Agile methods are often organized in frameworks, and the Scrum framework is one of the most used agile frameworks in software development. Scrum is organized in sprints, where the Scrum team works with a defined set of requirements from a Backlog over the course of the sprint. At the end of each sprint the work is reviewed and feedback from a product owner is taken into account. A ScrumMaster is in charge of the project, and makes sure that the team follows the principles of the framework (ScrumGuides.org, 2020). While following the principles of the Agile Manifesto (Beck et al., 2001), the Scrum framework is also relatively easy to implement in a company and imposes no restrictions on the choice of technology (Sommerville, 2016).

3 | Sensor Prototyping

This chapter presents the conceptual prototypes created in order to measure activity at the Assembly Station at Otretek AS. The medium-sized enterprise is a wood product manufacturer in Oppdal, Norway, which specializes in producing acoustic wooden lamellas. An example of the product can be seen in figure 3. As the prototypes were designed to provide data for production managers in the SME, they had to be small in size and non-invasive for the employees working at the Assembly Stations. A solution using Arduino Nano 33 IoT development boards is therefore presented in this chapter. The chapter also concerns the design thinking and Wayfaring process conducted, in-depth concept description and testing of the prototypes.

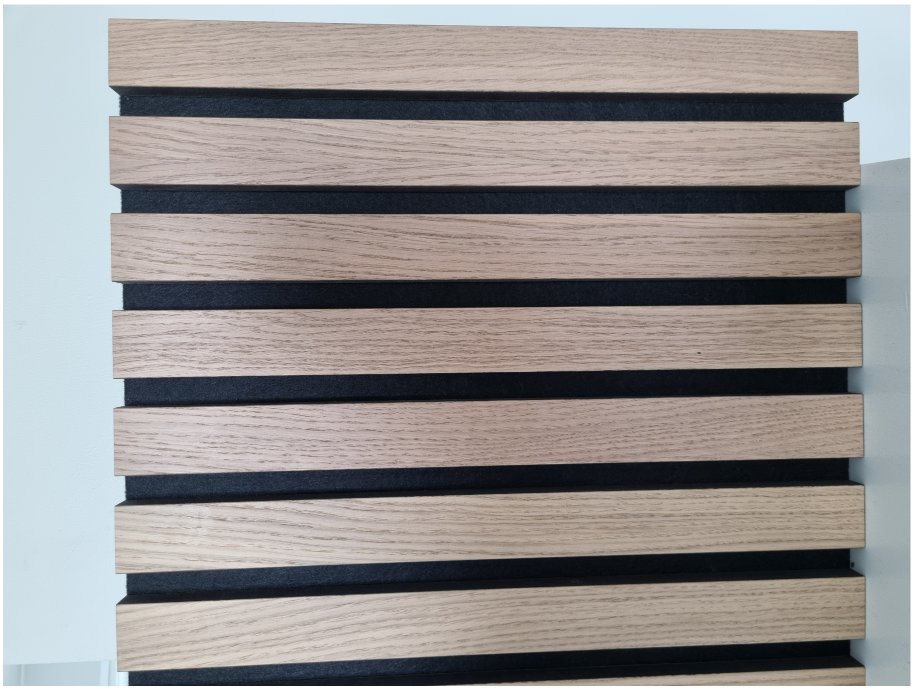


Figure 3: Acoustic wooden lamella produced by Otretek AS. The product consists of a backplate, lamellas and a combination of staples, adhesive or nails holding the two main components together.

3.1 Potential Solutions for a Wood Product Manufacturer

Otretek AS is a wood product manufacturer specializing in producing acoustic wooden lamellas. They also provide tailored solutions for larger interior projects such as schools and other public buildings. The production hall at Otretek AS consists of several zones, all of which were evaluated during needsfinding. In the next section the reasoning why the Assembly Station was chosen is elaborated on. This section shows some solutions that were ideated for each of the zones. Additionally, some existing solutions for similar environments are made known.

3.1.1 Material Delivery and Storage

The first step in the workflow at the production hall is the delivery of materials. This happens via the use of forklifts which transport the material to a designated storage location, before the material is transported further into the facility. Once the materials have been through the production line, the finished acoustic wooden lamellas are placed on pallets and transported to a new storage space. Either existing solutions for inventory tracking or automated delivery registration are options for this area of the production hall. The use of for example RFID for proximity readings (Yang and Yang, 2009) or scanning of ArUco markers (Guérin et al., 2016) are researched alternatives.

3.1.2 Sawing Station

The Sawing Station is where wood is cut into the right dimensions, so that they can be varnished or stained and then assembled into the finished product. In the case of Otretek AS, table saws which can be classified as legacy machinery is used. Possible data from the Sawing Station might include how much waste is produced or tracking of the state of machinery. While waste production for example can be monitored using weight sensors, the state of the machinery can be measured using vibration sensors. Using vibrations to monitor faults in machinery is covered in several studies, (Safizadeh and Latifi, 2014) being one example.

3.1.3 Wood Varnishing and Staining Station

Once cut, the wood is either varnished or stained according to the current order, and then set to dry. Otretek AS uses a relatiely new varnishing machine to varnish the wood, while staining (and sometimes spray painting) is done by hand. How much varnish or paint used is one measurable factor in this setting, which for example can be measured through weight measurements. It is also possible to monitor the number of planks treated if every plank must be placed at a specific location in order to be varnished or stained. This can for example be done using proximity sensors, which was attempted for the acoustic wooden lamella jig in the Project Thesis (Appendix A).

3.1.4 Assembly Station

The Assembly Station at Otretek AS consists of four workbenches, each equipped with pneumatic adhesive guns, staple guns and nail guns. Wooden jigs are placed on top of the workbenches and are produced for a specific product. Jigs might therefore be reused across orders. Next to the workbench pallets are placed with the wood and backplates needed to assemble acoustic wooden lamellas.

An attempt at tracking the production rate using photo cells was conducted in the Project Thesis (Appendix A), and a simple docker for tools was also made using an ultrasonic distance sensor. As these prototypes were implemented into a mock jig and would have to be either moved between the real jigs or implemented in all jigs, the solution might not have been ideal. New possible projects range from augmented reality aided assembly systems (Lai et al., 2020) to measuring movement of the tools using inertial measurement units (IMU).

3.2 Prototyping Sensors for the Assembly Station

Design thinking played a large role during the needsfinding and ideation for this project. Furthermore, probing the solution space using Wayfaring aided in eliciting unknowns and pave the path towards prototyping with Arduinos for the Assembly Station at Otretek AS. This section covers the impact of these methodologies in detail.

3.2.1 The Effect of Design Thinking on Sensor Development

The design thinking methodology was used throughout the project, starting with the empathizing phase during the autumn of 2021 over several visits to Otretek AS. The preliminary work conducted during this period was in relation to the Project Thesis, which can be found in Appendix A. Therefore, a full iteration of empathizing, defining, ideation, prototyping and testing was carried out in this period. The findings made during the Project Thesis serve as material for the empathy phase in this thesis, in other words serving as an exploration of the problem space (Lindberg et al., 2011).

The first visits of autumn 2021 consisted of conversations with the people at Otretek AS and demonstrations of the workflow. Through conversations with employees working with material preparation and product assembly, as well as production managers and operation managers, it became clearer who the potential user group was. Areas of interest and a variety of problems also arose. There seemed to be low interest in knowing data about the production among the employees working in the production hall. The interest seemed to be higher among managers, who at this point in time expressed that they lacked insight on the production rate, how much material was used per finished product and how much waste was generated throughout the process (Appendix A, p. 21). They did however provide approximations on how fast they could finish any given order, they could make supplier orders to keep enough materials in stock, and there seemed to be no overall problems with the workflow.

With little insight into the details of the production, especially regarding the assembly station, it was unclear if there were any points of the workflow that could be improved. This became the main motivator for the Project Thesis, where the output was a suggested way to increase the production rate of acoustic wooden lamellas. Although a small increase came from the prototype of the proposed solution, a decision was made to pivot and have a new look at the problem space for this thesis.

It is worth noting that some prototypes from the Project thesis are shown in the Wayfaring map as they were of high relevance in the ideation mode.

Following the pivot, the Project Thesis' findings were reflected upon and were used for constructing a point-of-view during the definition mode of the design thinking process. The point-of-view should reflect who the user is, his or her needs and the reasoning why this need exists, also called an insight (Doorley et al., 2018b). The following point-of-view was defined for this project:

A manager in a small- or medium-sized enterprise needs to gather data from analog tools or machinery because he or she does not know which parts of the production have the potential of being optimized.

The subsequent ideation mode then uses the point-of-view and proposes solutions to the expressed problem. This iterative process of ideation, prototyping and testing should lead the product developer to a user-centered and refined design (Doorley et al., 2018a).

3.2.2 Wayfaring Map

The ideation, prototyping and testing was conducted through the more specified Wayfaring methodology, where iteratively probing ideas in the solution space, building prototypes and testing them are fundamental aspects (Steinert and Leifer, 2012).

As reflected by the previous section, the solution space was large when taking the whole production line into account. The Assembly Stations were observed to be the stations with the most inconsistencies, both regarding the variation in type of product assembled as well as the individual differences between the employees working the stations. The latter was also expressed to be a pain point in a previous attempt by Otretek AS to map production rate at the stations (M. Thomassen, personal communication, November 2021). As expressed in section 3.1, the Assembly Stations provide several tools and processes that can be measured using sensors. They are also the last checkpoint before the product gets stored and shipped to a customer. These were some of the reasons why the Assembly Stations were chosen as the ones to develop sensors for when working on this thesis.

Several prototypes were tested in order to find the solution that would best fit the Assembly Stations and provide data that gave a realistic representation of the process. The following figure shows the steps made in the Wayfaring process. The final sensor prototype is described in the next section.

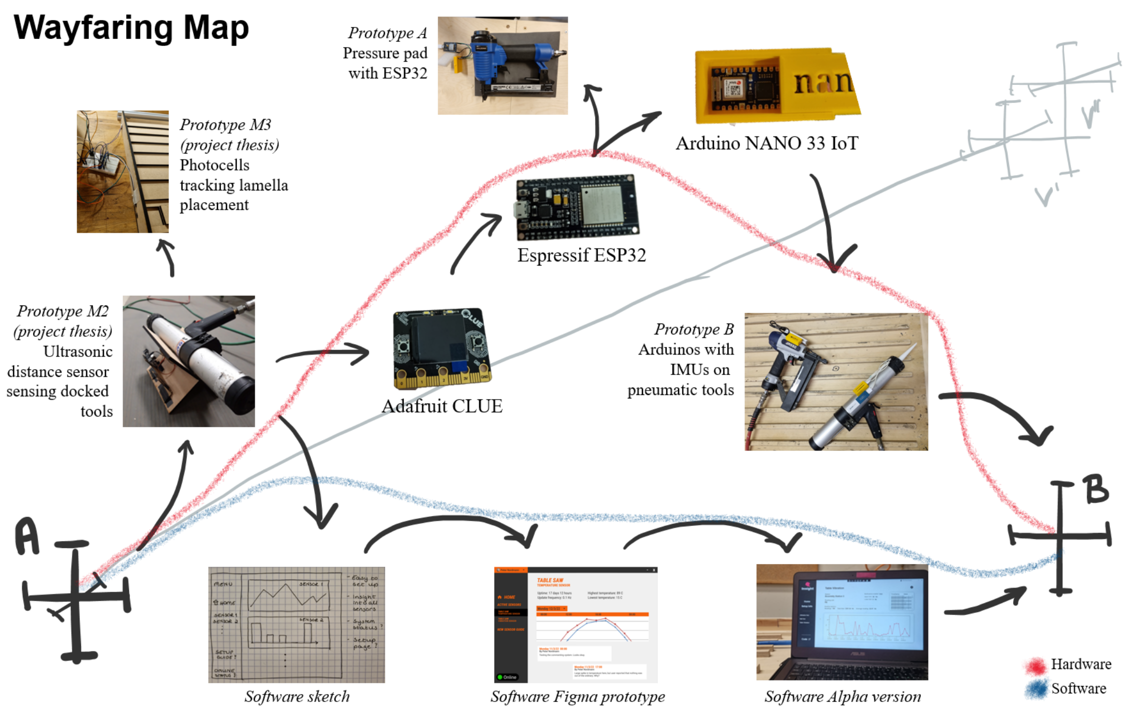


Figure 4: Wayfaring Map of the project. Prototypes M2 and M3 with figures retrieved from *Preliminary Work for Introducing Industry 4.0 in Acoustic Wooden Lamella Manufacturing* (Appendix A, p. 15-16)

3.3 Concept Description

The sensor solution for the Assembly Station consisted of an Arduino Nano 33 IoT development board with an integrated IMU, enclosed in a 3D printed casing for stability and protection. The development board was powered using power banks, and could alternatively receive power from electrical outlets if nearby. The integrated IMU, or inertial measurement unit, was used to capture acceleration data from the tools. The readings were then written to a real-time database hosted on Google's Firebase platform before being read by the software solution covered in the next chapter.

Employees working the Assembly Station at Otretek AS had several tools at their disposal. Which tools were being used varied according to the product being assembled. In the case of the acoustic wooden lamella, three types of pneumatic tools were used. An adhesive gun was used to apply adhesive to lamellas before adding a backplate, and a stapling or nailing gun was used to further fasten the backplate to the lamellas. The assembly was done on a metal workbench, and a wooden jig was placed on the workbench to fit the specific product being assembled. An Arduino Nano 33 IoT was mounted on an adhesive gun and a staple gun, as well as the metal frame of the workbench. The former two were programmed to measure movement via acceleration, and the latter measured acceleration data which was Fast Fourier Transformed into vibration data.

3.3.1 The Arduino Nano 33 IoT and Ease-of-Use

Several combinations of Internet-of-Things compatible boards and sensors were tested before arriving at the Arduino Nano 33 IoT. The requirements were that the solution was small in size, reliable and overall easy to understand. General ease-of-use would be highly valued. The Wayfaring map reflects the different ideas and prototypes considered, before arriving at a solution where a WiFi module and IMU were the essential components.

The tested alternatives with a low-cost wired IMU were primarily the Espressif ESP32 WiFi and Bluetooth MCU and the NodeMCU development board. As the user could potentially benefit from a small and reliable development board where no wiring is needed, the Arduino Nano 33 IoT seemed an appropriate alternative. The development board offers both a u-blox NINA-W102 WiFi and Bluetooth module and an LSM6DS3 module consisting of a 3D accelerometer and 3D gyroscope. It runs on a Cortex-M0 32-bit SAMD low power microcontroller unit and measures

45x18 mm (Arduino, 2022). While the price point of the Arduino is higher than the Espressif ESP32 and NodeMCU, it can be argued that no need for wires to get IMU readings, thorough documentation and support, and a small size outweigh the price difference. It is also possible to solder on pins on the Arduino Nano 33 IoT should the board be used for other purposes in future projects.

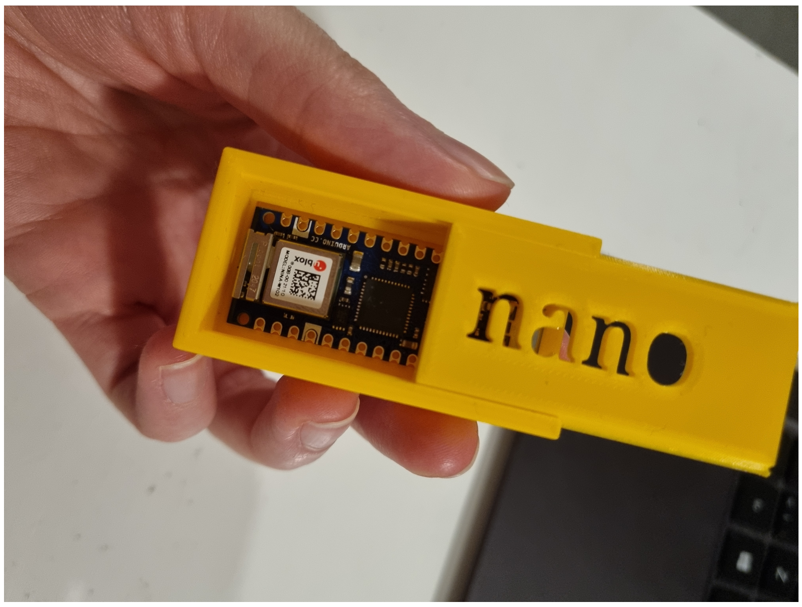


Figure 5: The Arduino Nano 33 IoT with a 3D printed case.

3.3.2 Measuring Movement

Movement was measured using the 3D accelerometer on the Arduino Nano 33 IoT's inertial measurement unit. During prototyping, accelerations in the x-, y- and z-directions were measured and analyzed to determine which readings resembled the movement pattern of a tool in use. As all of the pneumatic tools were laid on their side or hung on a wall when not in use, the only reading that should be registered at such a time was the standard gravity at roughly 9,81 m/s^2 or 1,0 g. There seemed to be a consistent pattern on each Assembly Station of either laying tools on their side or hanging them on hooks on the wall. This meant that the direction the Arduino was mounted on the tool would have to correspond to the decisive directions being used in the code logic. The code in Appendix B is adapted to tools laying on their side when not in use.

3.3.3 Measuring Vibration

Vibration was also captured in the form of acceleration readings which were transformed into vibration readings in the unit Hertz using Fast Fourier Transform. The Arduino FFT library was used for the transformations, providing built-in functions for frequency calculations of a sampled signal (Condes, 2022). The relevance of showing readings in the frequency domain when measuring potential vibrations in the workbench can be discussed. While there will likely be movement of the workbench and in the metal frame when performing the assembly, it is possible that the frequencies are higher than the maximum sampling rate of the accelerometer. An alternative is to plot the magnitude of the acceleration outputs, or to look at a one-dimensional acceleration reading to monitor movement like with the hand tools.

In the case of the LSM6DS3 the acceleration sampling rate is capped at 104 Hz, while the frequencies needed to for example detect faulty bearings in machinery lay in the range of 10-100 kHz (Safizadeh and Latifi, 2014). These vibration readings do however show how larger movements of the workbench appear in the frequency domain, which can be a useful insight in determining future use cases where a high-frequency sensor is implemented. An example is to mount a high-frequency accelerometer to the Sawing Station, where frequency readings can be used to diagnose the state of the tool.

3.4 Testing the Prototypes

This section shows the output from testing the Arduino Nano 33 IoT with integrated IMU on the tools at the Assembly Station at Otretek AS. The objective was to test the hypothesis that the readings gathered through the sensors can reflect the work conducted at the workbench. The plots of this chapter stem from the software described in chapter 4. The datapoints have also been compared to the ones in the database to ensure that the readings are correctly shown in the plots.

3.4.1 Procedure

The test was conducted at one of the four workbenches at the Assembly Station. Continuous sensor readings were made over the course of four hours. For consistency, one of the most experienced employees (M. Thomassen, personal communication, October 2021) were working the station during the test period so that the production rate would be high and potential problems would be resolved quickly. It was considered important to show and test a process with high ecological validity, therefore the test was conducted during normal production hours on real customer orders. A total of about 20 acoustic wooden lamellas were produced. A workbench used for the assembly is shown in figure 6.

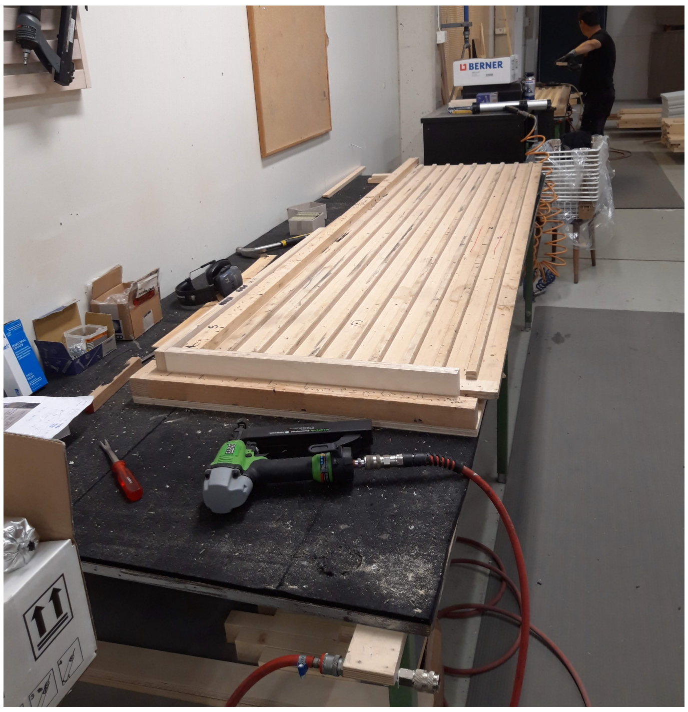


Figure 6: One of the Assembly Station workbenchs at Otretek AS.

There are different processes for different products. The one being produced during testing had both adhesive and staples binding the wood to the backplate of the product. During assembly, a jig which had been created for a specific order was used to guarantee consistency between the products, as well as to make the assembly easier for the employee. The first step of the assembly was to place wooden planks in the jig. The planks had been cut and either varnished or stained according to the order specification before being placed in the jig. Adhesive was then applied on the full length of each plank using a pneumatic adhesive gun. The backplate was placed on top and pushed down onto each plank to make sure the adhesive had spread. As the last step, a pneumatic staple gun was then used to staple the backplate to the planks. The finished product was removed from the jig and placed on a pallet, ready to be stored and then shipped off to the customer.

One Arduino was placed on the pneumatic adhesive gun and one on the pneumatic staple gun. Both tools were placed on their side when not in use, thus the Arduino was placed in a horizontal position so that the decisive direction for movement detection would be vertically along the z-axis. A third Arduino was placed on the metal frame of the workbench and set to detect vibration. The assembly was physically observed from a distance and at the same time observed via the software.

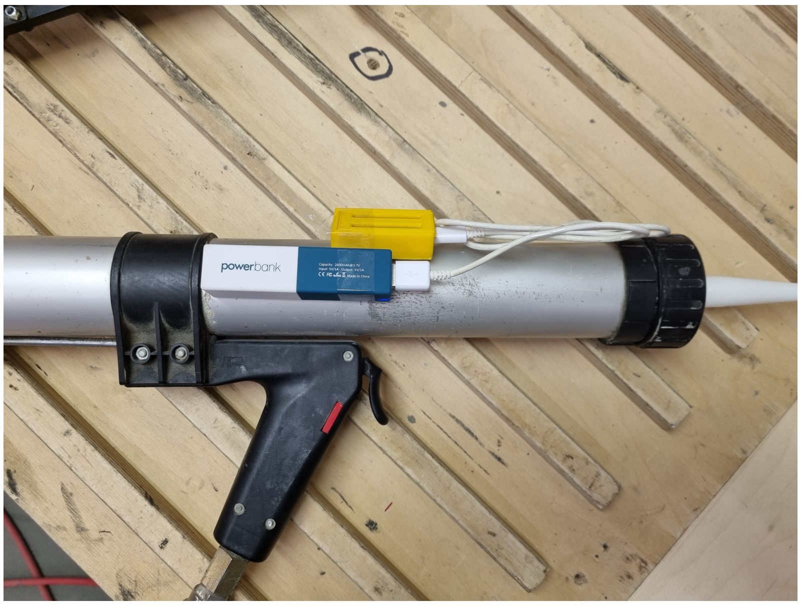


Figure 7: The pneumatic adhesive gun with the movement measuring Arduino. The Arduino slid towards the table during testing and became tilted in the y-direction, a possible error in the readings.

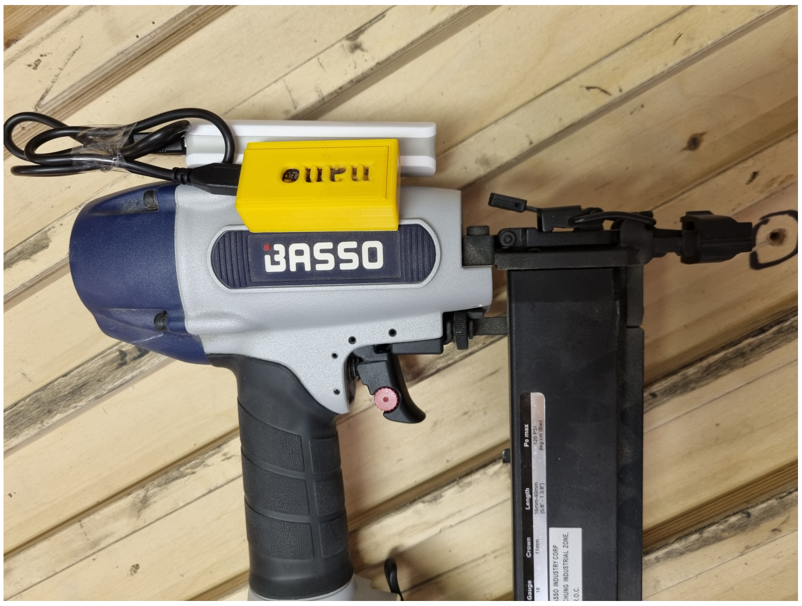


Figure 8: The pneumatic staple gun with the movement measuring Arduino.

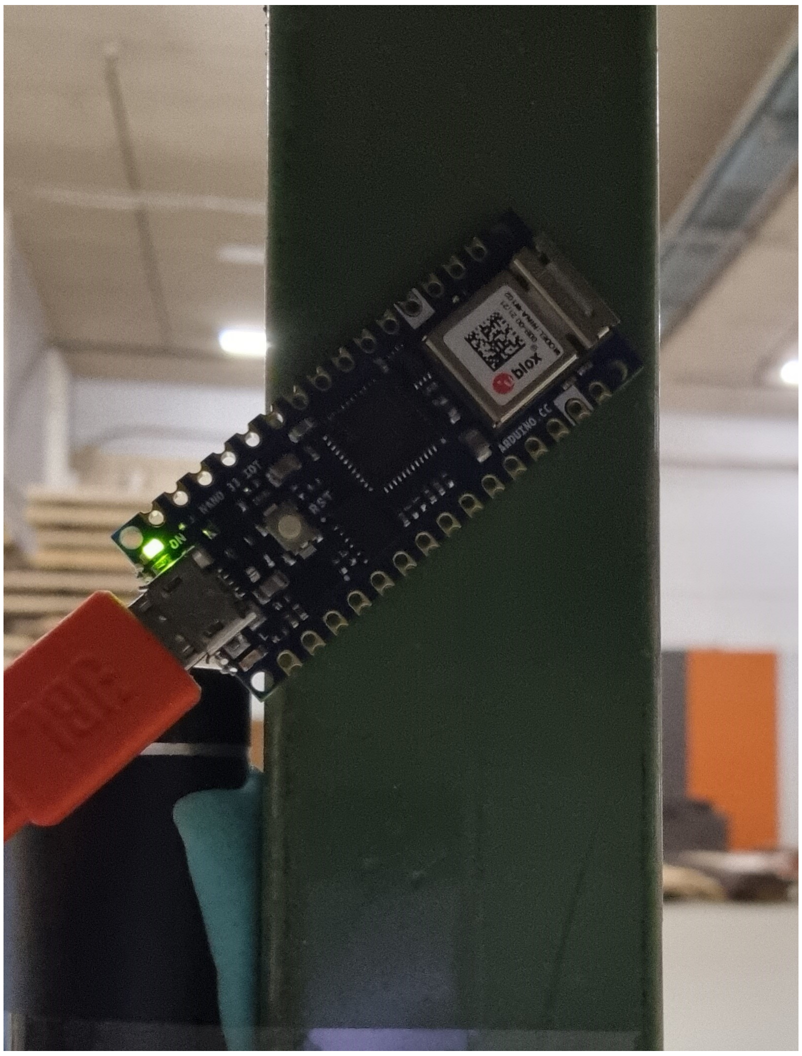


Figure 9: The vibration measuring Arduino on the metal frame of the workbench.

3.4.2 Results

The results are shown in the form of plots from three active sensors. As explained, an Arduino Nano 33 IoT was placed on three different locations at one of the Assembly Stations. Additionally, the station was physically observed and notes were taken at points which could be of significance for the readings. Some of these observations are referred to in this section.

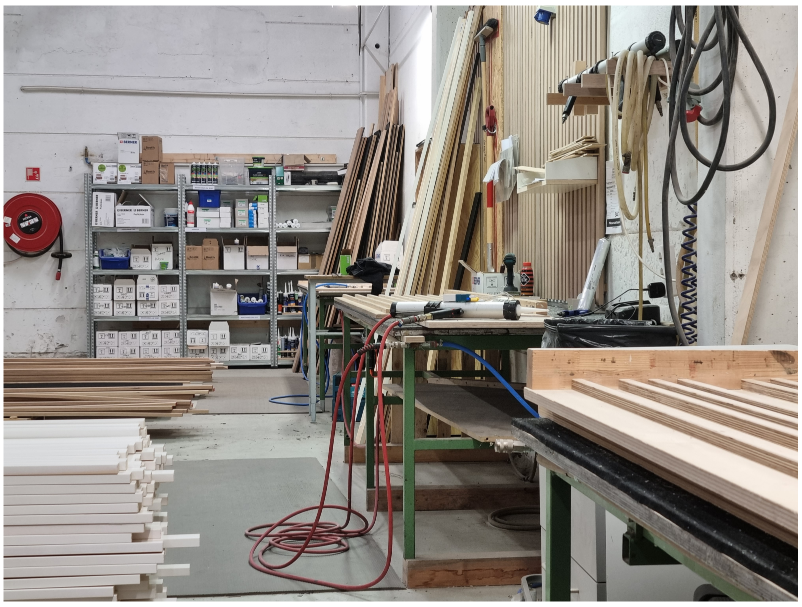


Figure 10: The point-of-view for test observation.

The first plot shows readings from the Arduino adhered to the side of the pneumatic adhesive gun, as previously seen in figure 11. Blue arrows along the x-axis indicate one "use" of the tool. The reading's timestamp is also seen on the x-axis. The y-axis show the duration of the "use" in seconds.

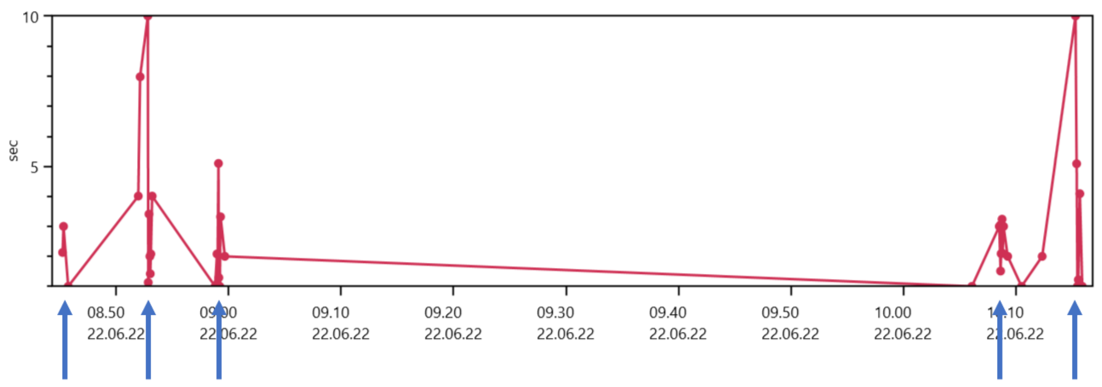


Figure 11: Use of the Adhesive gun with duration-based readings in seconds. The blue arrows each mark when the tool is used to assemble one Acoustic Wooden Lamella. Note the hour long lunch break between 09:00 and 10:00, displayed through the lack of reading points.

The code running on the devices measuring motion used vertical shifts, or shifts along the z-axis, to start a movement. The threshold for terminating a started movement was during the testing set to a low value, which caused one "use" of the tool to be split into several readings. This can clearly be observed in the plots for both the adhesive gun and the staple gun.

Whereas the readings clearly show when the adhesive and staple guns were in use, they are also prone to erroneous readings which were not filtered out at any point in the process. One of these points is clearly shown in figure 12 at the timestamp 08:32. This reading is registered as close to 60 seconds long, far longer than the average usage of the tool and it is also registered as one continuous movement. In the case of movement tracking, this type of erroneous reading differs from the valid readings to such an extent that it should be possible to filter them out.

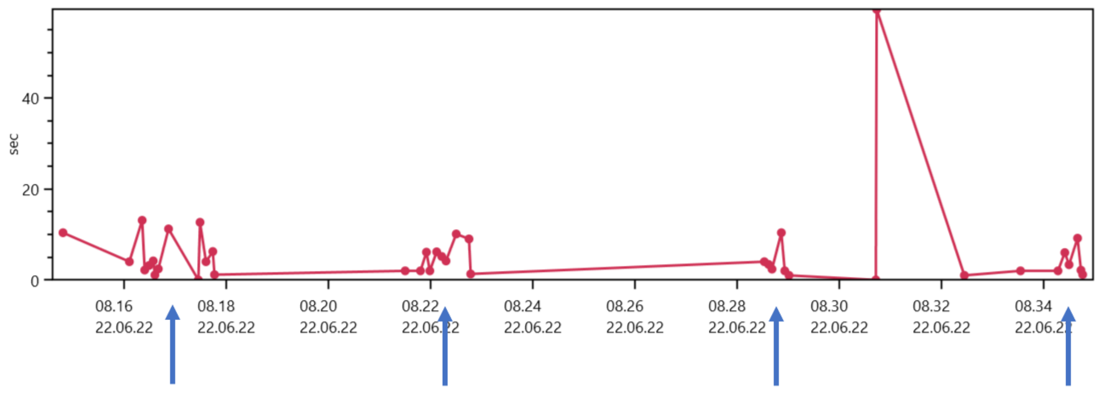


Figure 12: Use of the Staple gun with duration-based readings in seconds. The blue arrows each mark when the tool is used to assemble one Acoustic Wooden Lamella.

When comparing the readings from the Arduino on the pneumatic adhesive gun to the readings from the one on the pneumatic staple gun, some small differences appear. As shown in figure 13, the movements of the staple gun (top) causes relatively even readings during one ”use”. The adhesive gun (bottom) tends to have a longer first reading, followed by several shorter readings. The blue arrows in figure 13 show when the staple gun is used compared to the timeline of the adhesive gun. These two plots clearly show ”uses” which are defined enough to use in for example registration of the number of acoustic wooden lamellas produced.

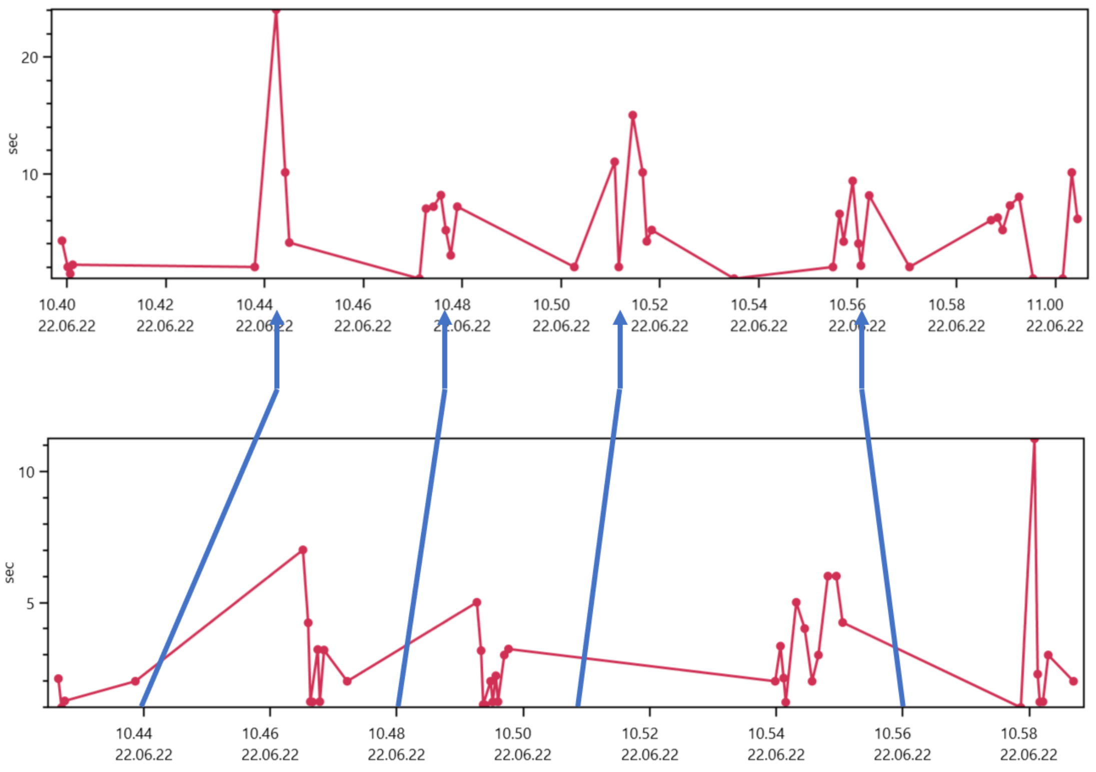


Figure 13: Comparison of the use of the two tools based on the reading timestamps. The staple gun is in the top plot and the adhesive gun in the bottom plot.

The next three plots show the output from the Arduino measuring vibration. There are several readings that stand out and that can resemble patterns for certain parts of the production. Figure 14 is an example, clearly showing level readings in Hertz when using the staple gun compared to the adhesive gun. Despite being a recurring pattern which showed up for most of the staple gun uses, the staple gun was also used in the first part of figure 15, which has more irregular spikes. The even readings from figure 14 are not present, and there seems to be no clear connection between the staple gun patterns of the two plots.

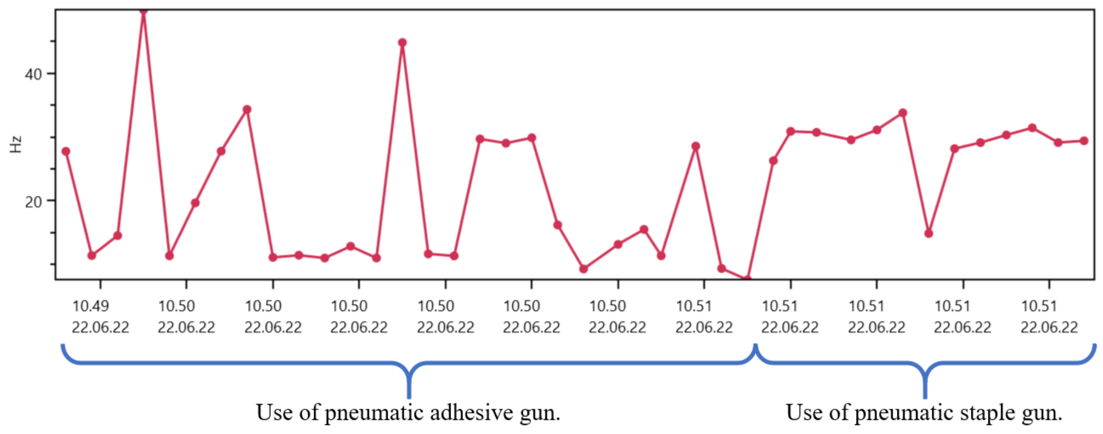


Figure 14: Vibrations registered for the pneumatic adhesive gun and pneumatic staple gun.

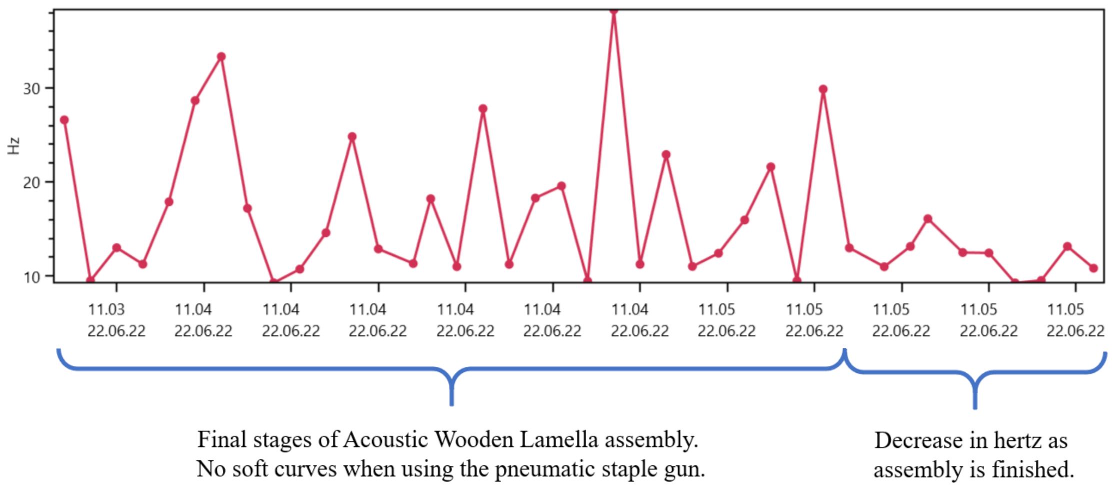


Figure 15: Drop in hertz at the end of assembling one Acoustic Wooden Lamella (11:05:20).

The last part of figure 15 shows how the readings reduce in hertz when work at an Assembly Station terminates. During observations, no work done at the Assembly Station mainly gave readings in the 10-15 Hertz range. However, figure 16 shows what might be one of the consequences of being in a production hall with a lot of heavy machinery. While the middle section of the readings give the expected output for no work, the outer sections of the plot show random spikes peaking at around 40 Hertz. This could possibly be the effect of running heavy machinery or material transport in close proximity to the Assembly Station.

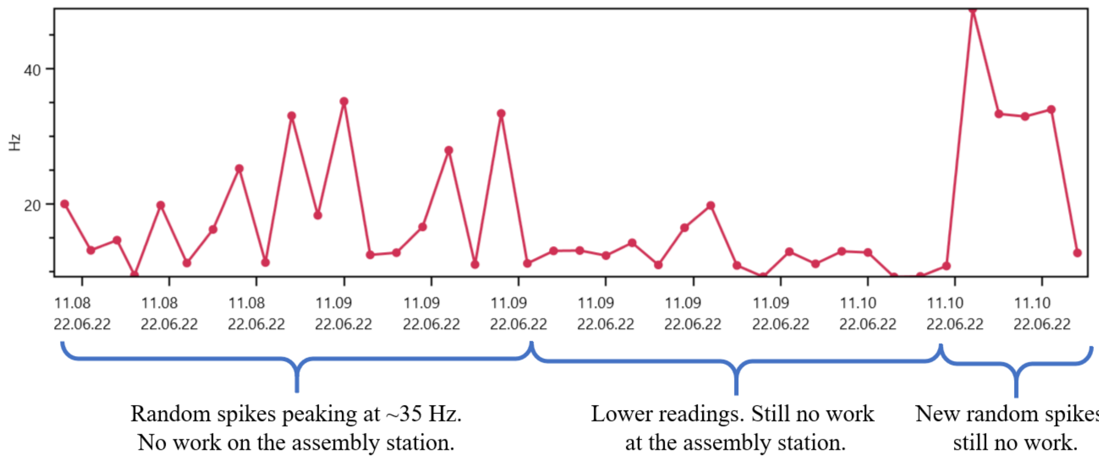


Figure 16: Vibrations registered from the production hall while no work is done at the Assembly station.

4 | Software Development

The software prototype is an attempt at making it easier for production managers in small- and medium sized enterprises to incorporate low-cost sensors in their operations. It should therefore serve a purpose in teaching the user to connect a set of defined microcontrollers to the software over WiFi via the use of pre-written code. Thus, one can define the software's mission (IEEE, 2000, p. 4) as "to enable the user to get an overview of the telemetry being read off multiple active sensors". As the project has no defined product owner (ScrumGuides.org, 2020) or acquirer (IEEE, 2000, p. 12), the user involvement has been limited to the individuals in the SME who have been available at the time of testing the software. This chapter delves into the ideation, prototyping and testing of the software. The last part of this chapter covers usability test results, while discussions of the results are covered in the next chapter.

4.1 The Software Development Process

An agile approach was used to plan and develop the software. As this was done by one person, only specific elements from selected agile frameworks were used. The main focus was to maintain the agile principles of being open and responsive to change and to prioritize working software over extensive documentation (Beck et al., 2001). This section covers the findings from the design thinking activities, as well as the produced backlog and information about it.

4.1.1 The Effect of Design Thinking on Software Development

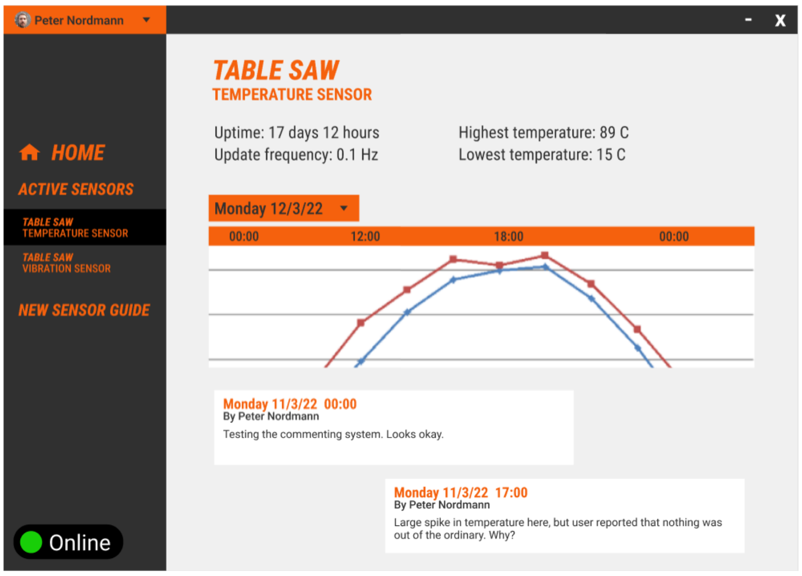
Design thinking was an integral part of the development process, both for the physical prototyping and the software development. As shown in the Wayfaring map in figure 4, the software itself was a prototype developed in parallel with the sensor prototypes for the Assembly Station.

The most impactful takeaway from the design thinking process when it comes to the software development, was the use of the defined point-of-view. However, some elements were added based on the feedback of the prototype from the Project Thesis, as well as discussions with employees from other SMEs during the spring of 2022. It was communicated that the SMEs had limited resources when it came to internal research and development and in developing new technical skills (M.

Løfaldli, personal communication, April 2022). It also became clear that while the technical knowledge on sensors, Internet-of-Things and programming among managers could be considered low, the interest was high. In addition, all of the SMEs in question already had or would soon gain access to a local makerspace meant to stimulate small-town innovation. The extended point-of-view can be seen below, where the added points are shown in boldface.

A manager in a small- or medium-sized enterprise needs to gather ***and view*** *data from analog tools or machinery* ***using an inexpensive and easy to understand system****, because he or she does not know which parts of the production have the potential of being optimized.*

A separate ideation process was conducted for the software development after the point-of-view was constructed. Mainly due to the fact that the software would be the larger contributor in enabling the user to setup and run the combined software and sensor system. Thus, the software would be in charge of providing a setup guide for sensors in addition to showing the sensor readings. When moving on to prototyping a solution, paper sketches were produced before moving on to digital prototypes in the design tool Figma (Figma, 2022). A backlog was also produced, which is covered in the next section.



Uptime: 17 days 12 hours
Update frequency: 0.1 Hz
Highest temperature: 89 C
Lowest temperature: 15 C

Monday 11/3/22 00:00
By Peter Nordmann
Testing the commenting system. Looks okay.

Monday 11/3/22 17:00
By Peter Nordmann
Large spike in temperature here, but user reported that nothing was out of the ordinary. Why?

Figure 17: Prototype created in Figma.

4.1.2 Agile Development and Creating a Backlog

Two principles from the agile Scrum framework were utilized. Weekly Scrum meetings with other engineering students generated useful insight from an outside perspective. These students were from the same academic environment but were not part of this project. The second principle was to set up a Backlog featuring user stories based on the design thinking activities conducted prior to the software development.

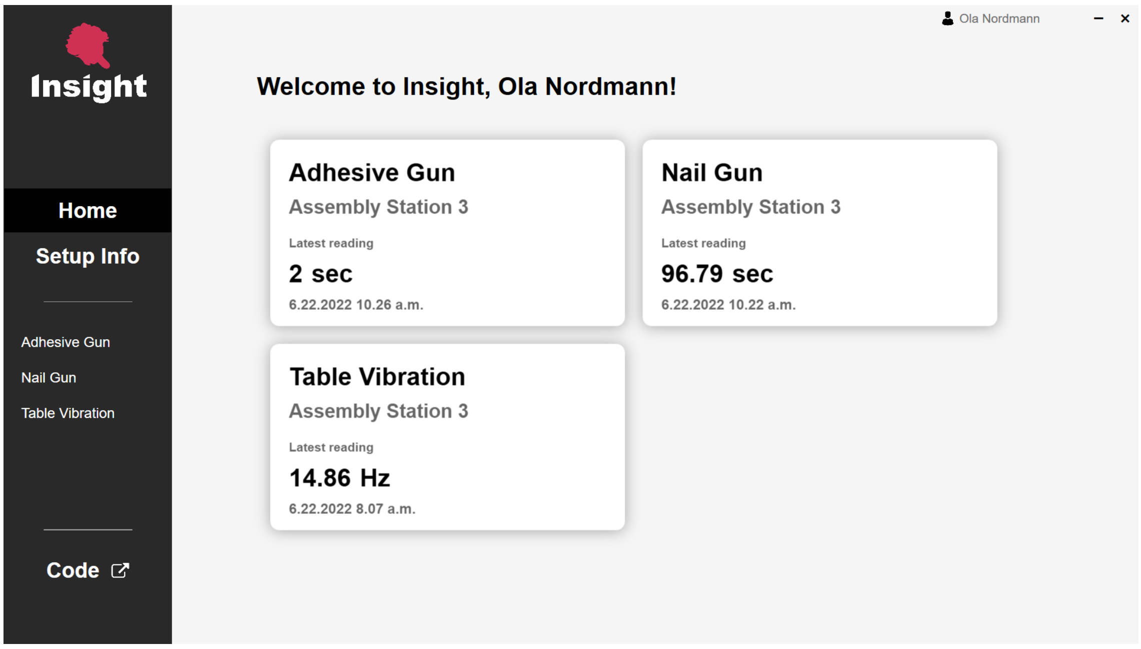
According to the official Scrum Guide "The Product Backlog is an emergent, ordered list of what is needed to improve the product" (ScrumGuides.org, 2020). The Backlog for this project was constructed using user stories, defined as natural language descriptions of situations to easier explain user needs and how user interactions happen (Sommerville, 2016, p. 774). They should also show the role of the user in relation to the software or product if relevant. User stories should therefore be easier to understand from a non-technical perspective than a Backlog created using purely technical requirements. Each user story was given an ID for development purposes, and they were sorted so that the user stories with low value ID's were the ones prioritized during development.

ID	User Story
1	As a user I wish to have a program that can be easily opened on my computer
2	As a maintainer I want to be able to add new sensors in less than 30 minutes
3	As a user I want to see all sensors at the same time to get a clear overview
4	As a user I want each sensor to have its own page with more detail
5	As a user I want to get a notification when sensor readings reach a given threshold
6	As a maintainer I want to be able to set thresholds for when I get a notification
7	As a user I want clarification on what a sensor does before I set it up
8	As a maintainer I want the program to have finished code for sensors so that I don't have to learn programming to set them up
9	As a user I want to log in from anywhere so that I see sensors from a remote location
10	As an administrator I want to control which users get to access my sensors

Table 1: The backlog created for the software development.

4.2 Concept Description

The software has been named "Insight" and is at the time of writing this thesis in an alpha state. The software being in an alpha state relates to early stage testing for software that is still in the development phase. It also reflects the importance of user-involvement and feedback in agile development methodologies which apply to the entire project (Sommerville, 2016, p. 249). The following sections address the software's mission and stakeholders, the technology being used and the architecture of the software.



Welcome to Insight, Ola Nordmann!

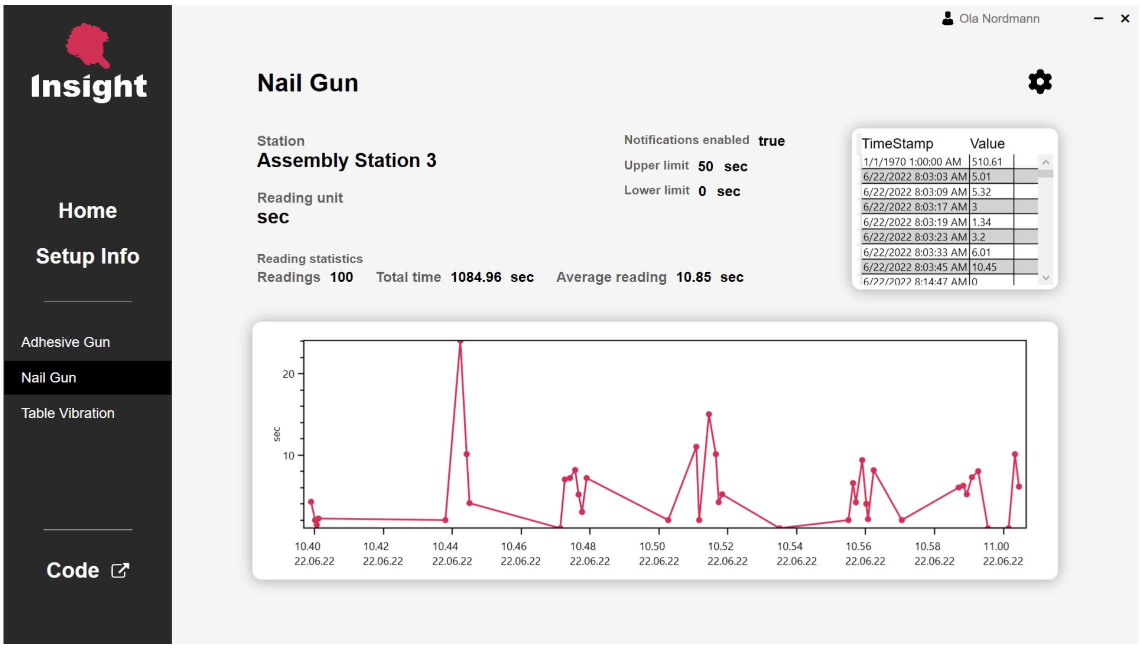
Adhesive Gun
Assembly Station 3
Latest reading
2 sec
6.22.2022 10.26 a.m.

Nail Gun
Assembly Station 3
Latest reading
96.79 sec
6.22.2022 10.22 a.m.

Table Vibration
Assembly Station 3
Latest reading
14.86 Hz
6.22.2022 8.07 a.m.

Figure 18: Insight's home page, currently with three sensors registered.

It is possible to navigate between three types of pages in "Insight", all shown in the side bar. The home page is the one welcoming you once you open the application. The setup info page contains info on how you find code and upload it on new microcontroller units. The sensor pages show info about each sensor registered in the database, and gets loaded into the side bar with the sensor name decided by the user. The button at the bottom of the side bar leads to a repository on GitHub.com where microcontroller code is hosted. The figures 18, 19 and 20 show the home page of the software, a sensor page and the setup page, respectively.



Nail Gun

Station
Assembly Station 3

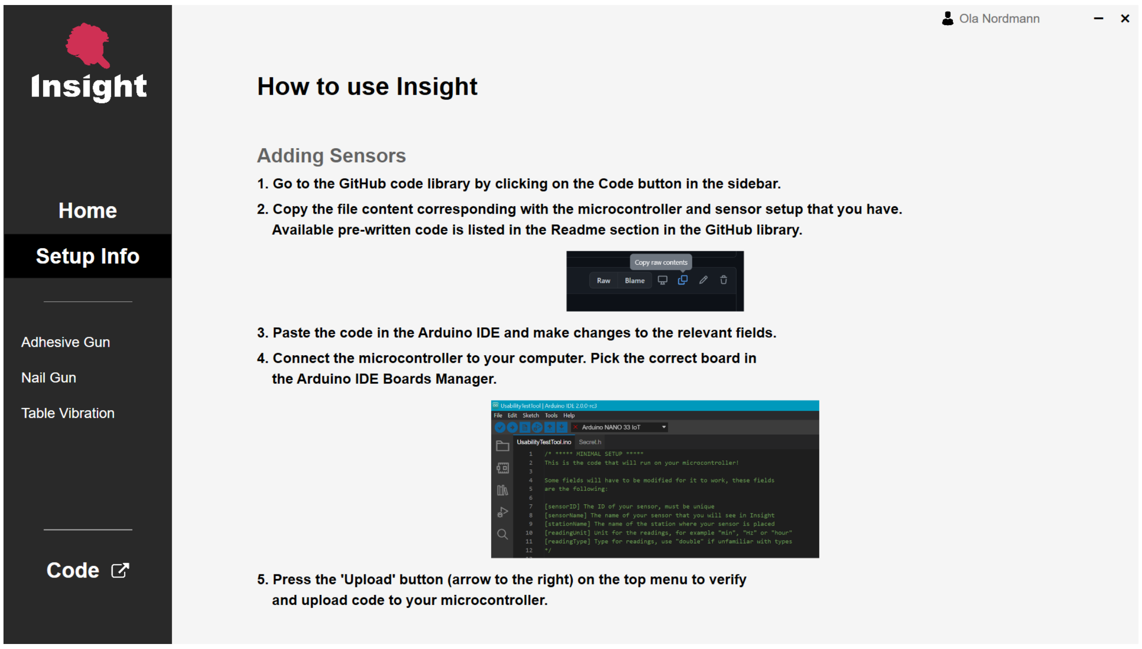
Reading unit
sec

Reading statistics
Readings 100 Total time 1084.96 sec Average reading 10.85 sec

Notifications enabled true
Upper limit 50 sec
Lower limit 0 sec

TimeStamp	Value
1/1/1970 1:00:00 AM	510.61
6/22/2022 8:03:03 AM	5.01
6/22/2022 8:03:09 AM	5.32
6/22/2022 8:03:17 AM	3
6/22/2022 8:03:19 AM	1.34
6/22/2022 8:03:23 AM	3.2
6/22/2022 8:03:33 AM	6.01
6/22/2022 8:03:45 AM	10.45
6/22/2022 8:14:47 AM	0

Figure 19: The page for the sensor currently sending data from the pneumatic nail gun.



How to use Insight

Adding Sensors

1. Go to the GitHub code library by clicking on the Code button in the sidebar.
2. Copy the file content corresponding with the microcontroller and sensor setup that you have. Available pre-written code is listed in the Readme section in the GitHub library.
3. Paste the code in the Arduino IDE and make changes to the relevant fields.
4. Connect the microcontroller to your computer. Pick the correct board in the Arduino IDE Boards Manager.

```
/* ***** MINIMAL SETUP *****
This is the code that will run on your microcontroller!

Some fields will have to be modified for it to work, these fields
are the following:

[sensorID] The ID of your sensor, must be unique
[sensorName] The name of your sensor that you will see in Insight
[stationName] The name of the station where your sensor is placed
[readingUnit] Unit for the readings, for example "min", "Hz" or "hour"
[readingType] Type for readings, use "double" if unfamiliar with types
*/
```

5. Press the 'Upload' button (arrow to the right) on the top menu to verify and upload code to your microcontroller.

Figure 20: The page containing setup info for microcontroller units.

Notifications have also been implemented to show when readings surpass a threshold set by the user. The upper and lower limits are changed on the settings subpage of a sensor page. Figure 21 shows one of the notification types implemented.

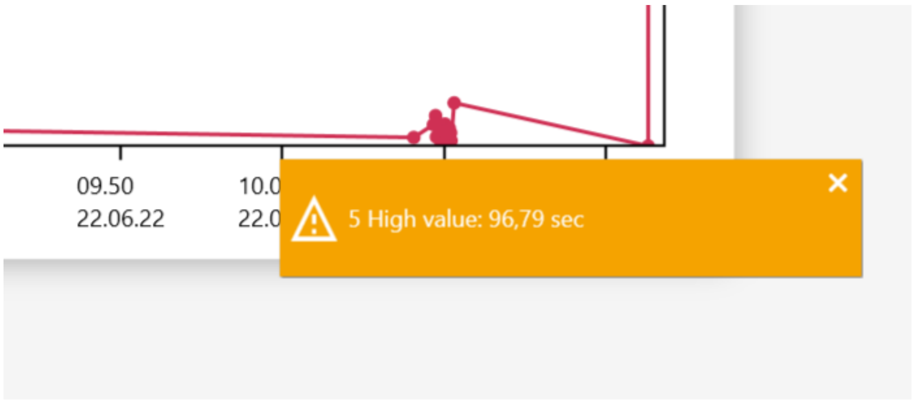


Figure 21: Notification showing when readings go over or under a set threshold.

4.2.1 Mission and Stakeholders

The software's mission (IEEE, 2000, p. 4) is to enable the user to get an overview of the telemetry being read off multiple active sensors. It is also meant to serve a purpose in teaching the user to connect a set of defined microcontrollers to the software over WiFi via the use of pre-written code. Ideally it is an entry-point for users to develop their own logic for Arduinos or other microcontrollers, suitable to other parts of production than the Assembly Station covered in this thesis.

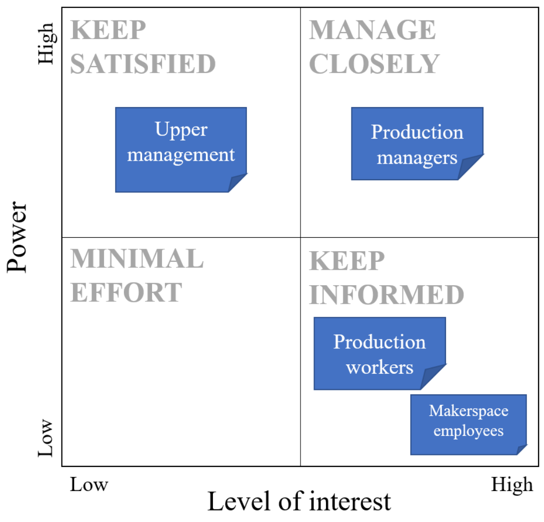


Figure 22: The power-interest matrix by Johnson and Scholes (2008).

The most important stakeholders of the software project are the employees at Otretek AS, more specifically the production workers, production managers and upper management. One can also incorporate employees at the local makerspace in the mapping. As a medium-sized enterprise, the complete stakeholder map is relatively compact compared to larger organizations. It is also not relevant to show the complete map when defining the stakeholders of the software. Placing the stakeholders in the power-interest matrix by (Johnson et al., 2008, p.156) paints a clearer picture of who the software is designed for, and which relationship the other stakeholders have to the software. Based on the information gathered, it is likely that the production managers will be the sole end-users of the software, yet it might still be of relevance to test with other employees as well.

4.2.2 Choice of Technology

"Insight" is designed as a desktop client application which aims to ease the transition to low-cost Industry 4.0 solutions for SMEs. As the need for a low-cost solution is essential to the project, "Insight" is intended as Free software (Williams, 2011, p. 15). The project will also be made publicly available on the code-sharing and version control provider GitHub.com. As it is intended to be open-source and free-to-use, it differs from remotely hosted software running on a subscription model. This business model called Software-as-a-Service (SaaS) is widely used in today's market (Oliveira et al., 2019). The intention of "Insight" being Free software ultimately affects the choice of technology used, as for example all API's and services must also be free-to-use and preferably open-source in the project.

Based on the initial visits to Otretek AS it is clear that in their case, all computers run on the Windows operating system. It is also assumed that having a desktop client application and not a remotely hosted service will be easier to handle for the end-users. The UI framework Windows Presentation Foundation (WPF) has therefore been chosen. This is a framework that supports a backend written in the programming language C# and frontend written in the markup language XAML. All of these technologies have comprehensive documentation, making them suitable for open-source projects. Ideally, further development happens via contributions from other programmers who can then also request that they become part of the official project on GitHub.com.

4.2.3 Patterns in Software Architecture

The software is built with some selected architectural patterns. According to Sommerville, architectural patterns are "stylized, abstract descriptions of good practice, which have been tried and tested in different systems and environments" (Sommerville, 2016, p. 176).

The Model-View-ViewModel architecture pattern is the most prominent pattern used and works particularly well with Windows Presentation Foundation (WPF) applications. The pattern has officially been incorporated in the WPF framework (Smith, 2009). Its main feature is that the pattern clearly separates the presentation layer, called the View, from behavior and data. The business logic which handles for example database operations and connects parts of the code, resides primarily in the ViewModel and Models. Other patterns can be used to further partition the business logic appropriately, for example by moving much of the business logic to services. This is the case for "Insight", as seen in figure 23. The separation of the View, which is written in markup language, enables rapid prototyping as a whole new graphical user interface can be written without changing the C# code handling the logic.

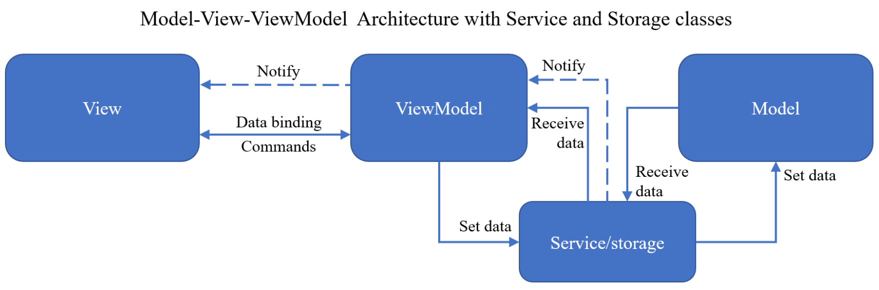


Figure 23: Visualization of the Model-View-ViewModel pattern which also incorporates services.

The Mediator design pattern is used to communicate specific changes throughout the business logic or across ViewModels. It consists of messengers who publish messages and subscribe to them. These can be implemented where needed, and promotes loose coupling between ViewModels (Gamma et al., 1995, p. 273)

The Singleton design pattern is used to construct singular instances of classes, for example the project's Firebase storage class which continuously updates with readings from the database. The service and storage classes in the project are all Singletons. New instances of these classes can only be retrieved through a single access point (Gamma et al., 1995, p.127).

4.2.4 The "4+1" View Model of Software Architecture

The architectural description of the software and its relations is based on the IEEE's Recommended Practice for Architectural Description of Software-Intensive Systems

(IEEE, 2000). However, it does not cover the full extent of the recommendation due to the project being relatively small and not yet completed. The architectural description is thus limited to a set of views based on The "4+1" View Model of Software Architecture (Kruchten, 1995). The views of the "4+1" View Model pertain to a specified set of stakeholders, and it provides a more specified version of the earlier mentioned recommended practices.

4.2.4.1 *The Logical Architecture (Class diagram)*

The logical view reflects the functional requirements that can be extracted from the user stories in the Backlog (Sommerville, 2016, p.149). In the case of Otretek AS the end-user associated with the user stories would be a production or operations manager. Making a complete class diagram is outside the scope of this thesis, and the class diagram therefore comprises of the classes that properly reflect the functional requirements. It should be reiterated that these requirements cover the desktop client application, and not the entire process of setting up the system with both microcontroller units and software. As a complete class diagram would be too extensive for this section, a class diagram showing the most essential classes has been created.

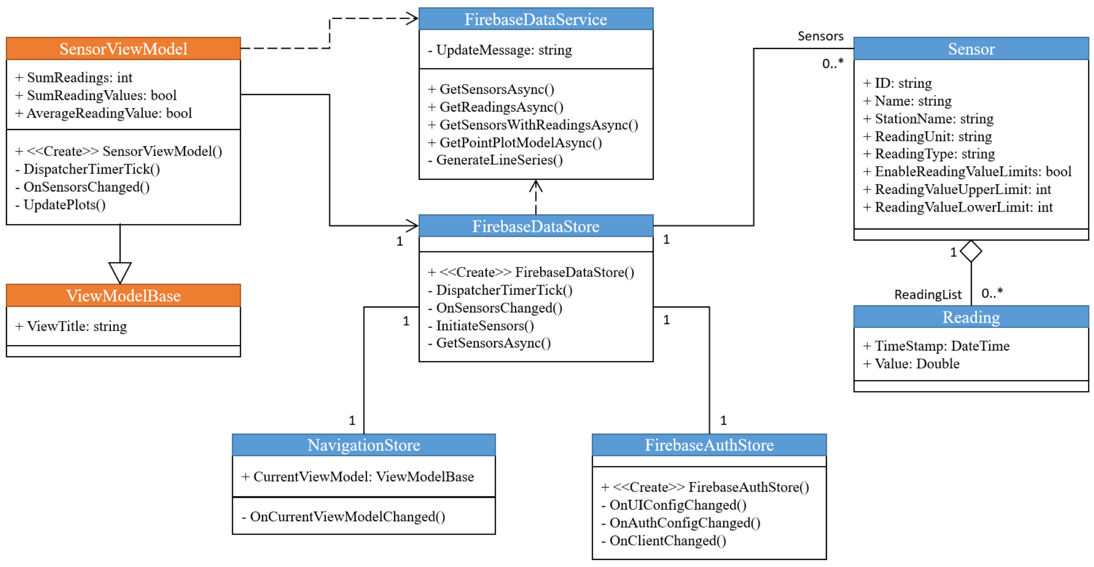


Figure 24: The most essential parts of the logical architecture shown in a class diagram.

4.2.4.2 *The Process Architecture (Activity diagram)*

The process view concerns the system's components at runtime, and can be useful when looking at non-functional requirements (Sommerville, 2016, p.174). In this case it is shown in the form of an activity diagram. The activity diagram shows the behavior of the system in a set of tasks on a high-level, which at the current development stage is more appropriate than going into detail on the processes as they might be altered.

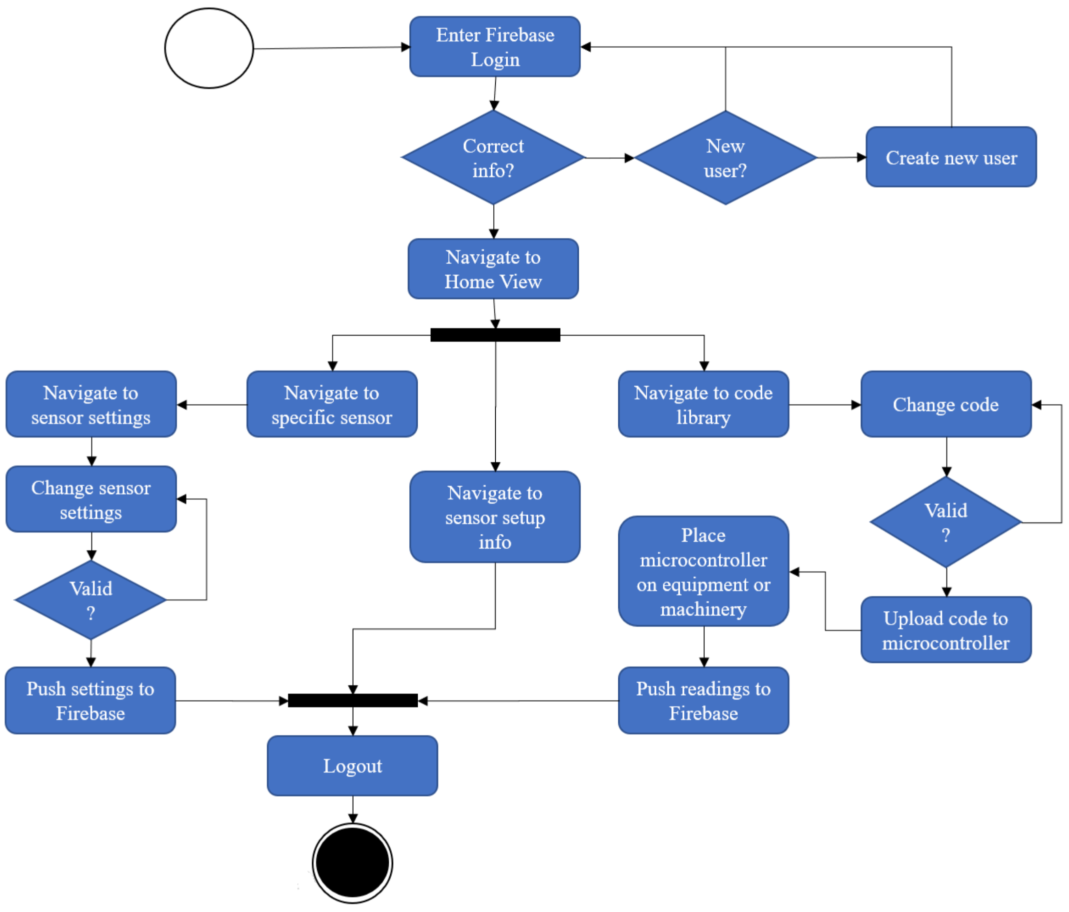

Enter Firebase Login
Correct info?
New user?
Create new user
Navigate to Home View
Navigate to sensor settings
Navigate to specific sensor
Navigate to code library
Change code
Change sensor settings
Navigate to sensor setup info
Valid ?
Place microcontroller on equipment or machinery
Valid ?
Upload code to microcontroller
Push settings to Firebase
Push readings to Firebase
Logout

Figure 25: The process architecture shown in an activity diagram.

4.2.4.3 *The Development Architecture (Package diagram)*

The package diagram is a structural diagram meant for developers. It shows elements of architectural significance, specifically related to packages or components that can be tackled by a development team (Sommerville, 2016, p. 174). The presentation layer created using the UI framework can for example be created by a designated team of frontend developers, working with only that package.

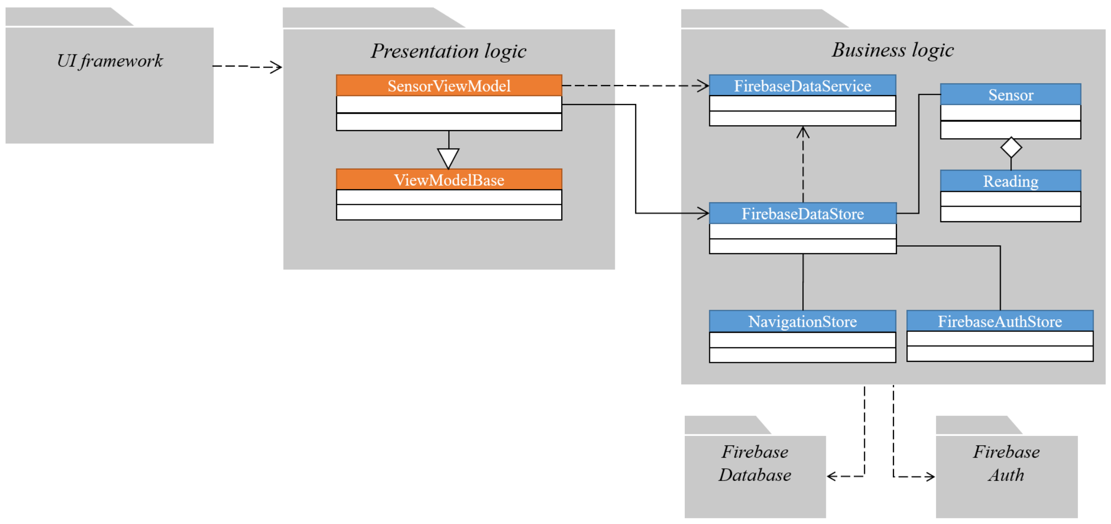


Figure 26: The development architecture shown in a package diagram.

4.2.4.4 ***The Physical Architecture (Deployment diagram)***

The physical architecture is shown through a deployment diagram, a type of structural diagram which also stems from non-functional requirements. These diagrams show the components needed to get the system running, which at the time of writing this thesis might be subject to change. Note that the microcontroller unit is included in this diagram. The current deployment diagram is shown in figure 27.

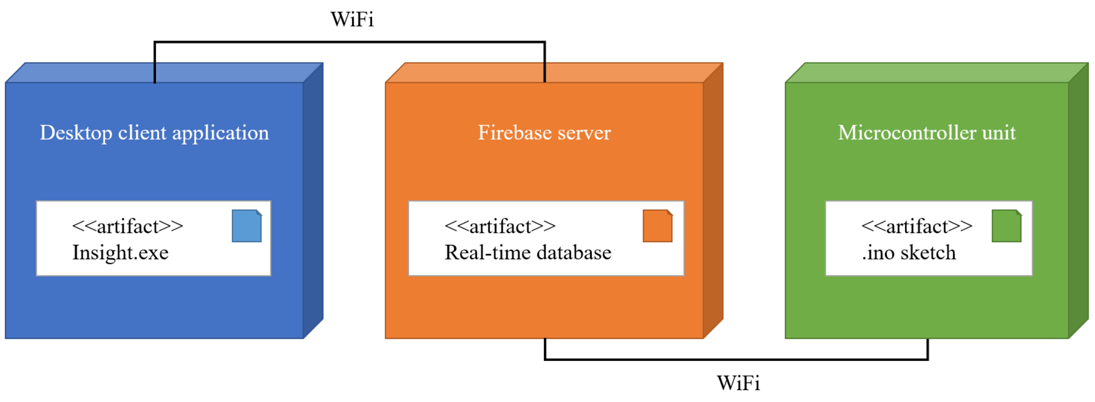


Figure 27: The physical architecture shown in a deployment diagram.

4.3 Testing the Software

A test which produced both quantitative and qualitative data was conducted in order to assess the software in its current state. The participants were individuals representing either Otretek AS or the makerspace at Nasjonalparken Næringshage in Oppdal. This section covers information about the participants, the test procedure and results from the testing. Discussions of the results are covered in the next chapter.

4.3.1 Participants

When testing "Insight", six individuals were asked to participate in the usability test. Participants in the test were either employees at Otretek AS or individuals involved with the maker space at Nasjonalparken Næringshage. According to (Virzi, 1992), 80 percent of usability problems are uncovered with four or five participants in a usability test. Additionally, the most severe problems will be uncovered after testing with just the very first subjects.

Following the General Data Protection Regulation's definition (ProtonTechnologies, 2022), no personal data was collected during the testing of the prototype. Verbal consent for participation was deemed sufficient for the tests. The pre-test survey and System Usability Scale can be found in Appendix D.

4.3.2 Procedure

The participants were first given a set of screening questions for documenting their IT knowledge and interest in IoT concepts. They were then given a short introduction to the purpose of the project before being presented with the software. Each participant was asked to solve nine tasks while giving oral feedback to the test supervisor. The participants were carefully observed, and was allowed to ask questions along the way. However they were encouraged beforehand to try and solve the tasks on their own before asking for help. The primary goal of the procedure was to collect data on user performance, as a central component of usability testing within Human-Computer Interaction (HCI) (Preece et al., 2015, p. 655).

Figure 28 shows the setup for the usability test. As there were limited resources available at both Otretek AS and Nasjonalparken Næringshage for setting up a designated room for usability tests, the setup was kept as simple and free of distraction as possible.

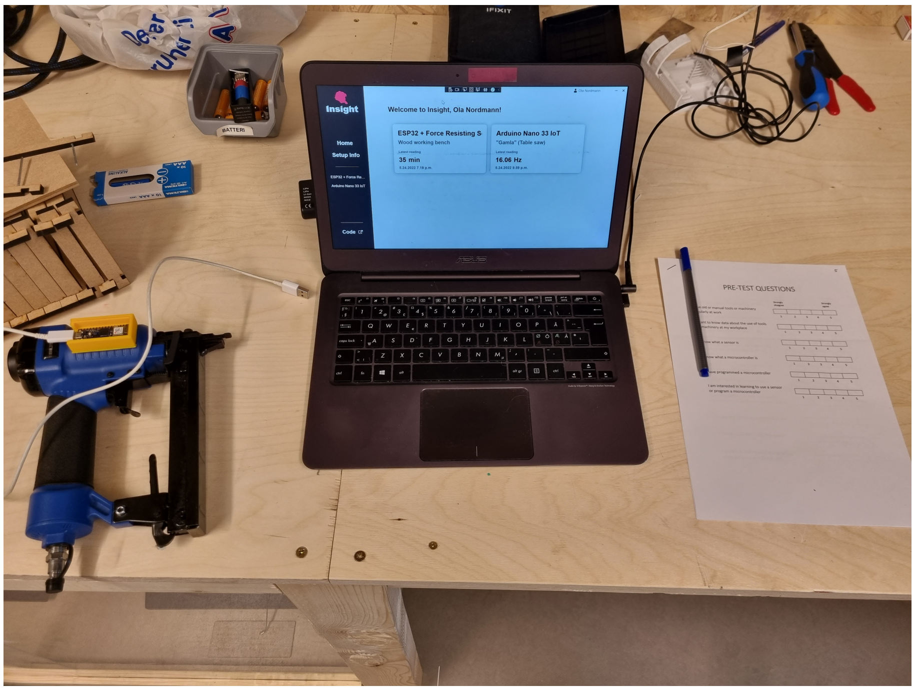


Figure 28: The setup when conducting usability tests.

A System Usability Scale (SUS) was used to collect data on the user satisfaction with the system. SUS is described as being technology agnostic, and therefore well suited for testing a system consisting of both hardware and software (Bangor et al., 2008). As the surveys were given to the subjects in English as opposed to their native Norwegian, the version proposed by (Bangor et al., 2008) was used. This version has some linguistic changes from the original scale by (Brooke et al., 1996), which should make it easier to understand for the subjects. 5-point Likert scales were used for quantitatively measuring user satisfaction with statements made in both the pre-test survey and the System Usabilty Scale (Appendix D)(Preece et al., 2015, p. 348).

The subjects were asked to solve the following nine tasks **a-i** as part of the usability test. Some of the tasks had follow-up questions which helped generate qualitative data.

a) Have a look around the home page of the software. Can you tell me what you observe?

b) Can you navigate to the sensor “Arduino Nano 33 IoT”? What do you observe?

c) You wish to add a new sensor to the program, where would you go to find info about it? Does the process look doable?

d) Can you go to the external website where code is located and open the “UsabilityTest” file?

e) Copy the contents of the file and paste it in the code editor which is currently open on the computer.

f) Change the info fields “sensorName” and “stationName” at the top of the file to your liking.

g) Using the setup page, can you figure out how to upload code to the microcontroller?

h) When the upload has finished, can you go to the software and check if your sensor is registered? How difficult did you find the setup process?

i) Can you go to your sensor’s settings, set the upper value limit to 20 seconds and trigger a notification?

There were two hardware components the subjects had to interact with as part of the testing, the staple gun with an Arduino and a loose cable, and a computer. As the software was still being developed at the time of the testing, it was built in the Visual Studio IDE and already running on the computer when the subject got the computer. While not part of the usability testing, this meant that possible errors during run-time could be reviewed after a finished test. Figure 29 shows the stapler that the subjects were tasked with moving as the final task of the usability test.

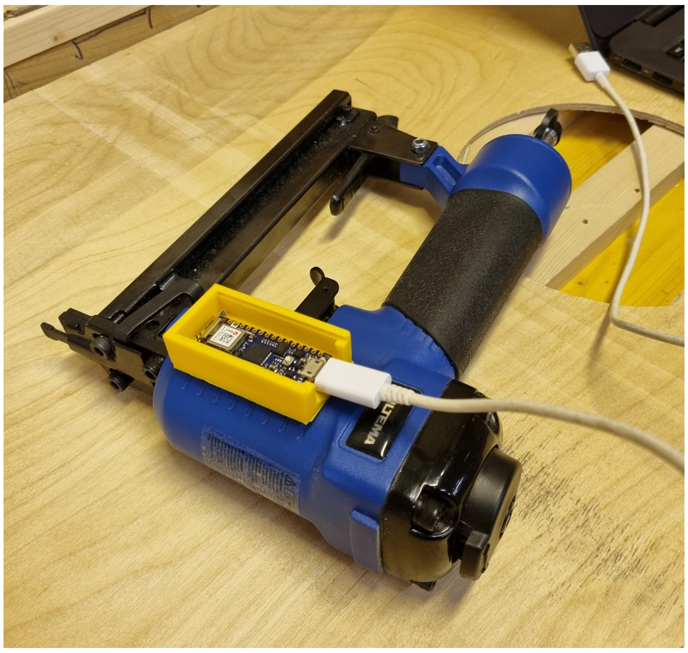


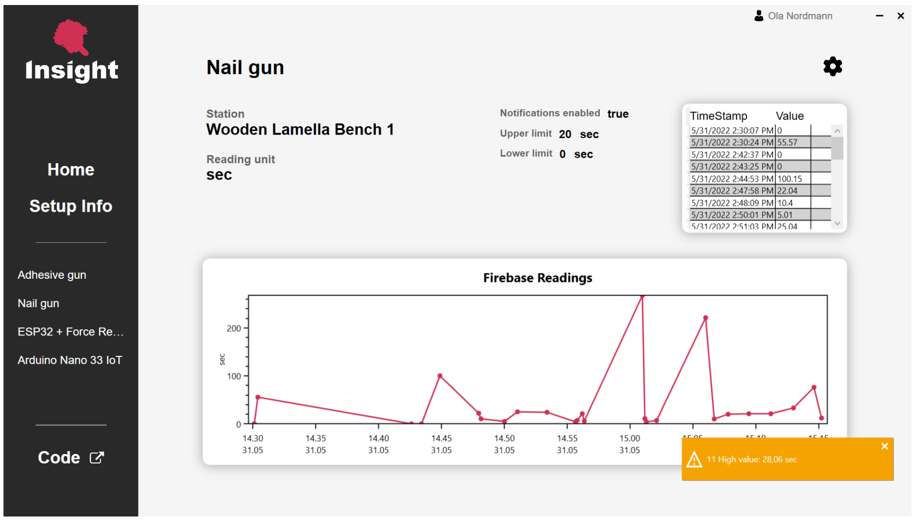
Figure 29: An Arduino on a small staple gun for the usability test.

4.3.3 Results

The observations and oral feedback made during testing resulted in a small yet insightful set of qualitative data. For one, the setup guide contained some formulations unfamiliar to the subjects, such as "GitHub" and "code editor". Surprisingly, the subjects who expressed that they were unfamiliar with the terms were also the ones who had the largest expressed confidence in believing that they could perform the tasks.

When asked about the information on the sensor pages, only one subject showed full understanding of what was presented. Other subjects were confused about both sensor names, the title "Firebase Readings" on the plot and the units presented. However all but one subject immediately knew that the updating plot and table showed sensor data. Interestingly, all subjects had some form of difficulty understanding the relation between the thresholds they were told to change in the sensor's settings, and the duration of physical movement they had to do to trigger a notification.

Figure 30 shows how a sensor page looked during the usability testing. Some small changes were made directly after, which is covered in the next chapter.



Nail gun

Station
Wooden Lamella Bench 1

Reading unit
sec

Notifications enabled true
Upper limit 20 sec
Lower limit 0 sec

TimeStamp	Value
5/31/2022 2:30:07 PM	0
5/31/2022 2:30:24 PM	55.57
5/31/2022 2:42:37 PM	0
5/31/2022 2:43:25 PM	0
5/31/2022 2:44:53 PM	100.15
5/31/2022 2:47:58 PM	22.04
5/31/2022 2:48:09 PM	10.4
5/31/2022 2:50:01 PM	5.01
5/31/2022 2:51:03 PM	25.04

11 High value: 28,06 sec

Figure 30: The state of "Insight" during usability testing.

Table 2 shows the results of the questionnaires in Appendix D, where participants answered a set of statements based on satisfaction-based Likert scale. The System Usability Scale got mean score of 60,8 with a standard deviation of 17,5. In a 2008 study, (Bangor et al., 2008) reported a mean SUS score of 70,14 after mapping the results of over two thousand surveys. They stated in the same report that viable or passable products often reached scores over 70, while superior products reached 90.

	A	B	C	D*	E	F**
Pre-test Survey						
I use old or manual tools or machinery regularly at work	2	1	1	5	5	4
I want to know data about the use of tools or machinery at my workplace	5	5	3	3	5	4
I know what a sensor is	5	3	3	3	5	2
I know what a microcontroller is	5	1	1	1	5	1
I have programmed a microcontroller	4	1	1	1	5	1
I am interested in learning to use a sensor or program a microcontroller	5	5	4	3	5	3
System Usability Scale (SUS)						
I think that I would like to use this system frequently	4	4	5	1	3	3
I found the system unnecessarily complex	2	3	1	3	3	2
I thought the system was easy to use	4	3	5	3	3	4
I think that I would need the support of a technical person to be able to use this system	2	3	2	5	4	4
I found that the various functions in this system were well integrated	3	4	4	5	2	4
I thought that there was too much inconsistency in this system	2	3	1	4	2	2
I would imagine that that most people would learn to use this system very quickly	2	4	5	3	4	4
I found the system very awkward to use	2	2	1	2	4	2
I felt very confident using the system	4	2	4	3	2	3
I needed to learn a lot of things before I could get going with this system	2	3	1	5	4	2
SUS Score	67,5	57,5	92,5	40,0	42,5	65

SUS Mean 60,8
SUS Standard deviation 17,5

** User had to get surveys translated to Norwegian*
*** Arduino failed to go online during usability test*

Table 2: Results from the pre-test survey and the System Usability Scale.

5 Discussion

This chapter contains a discussion of the test results from the two conducted tests at Otretek AS. The discussion revolves around the impact of the results, the ecological validity of the tests and the possible errors in the methods used. This chapter also addresses the three research questions asked at the beginnig of this thesis. These questions are repeated below.

1) Can data from low-cost Industry 4.0 solutions accurately represent the activity conducted at a wood product assembly station?

2) Is a production manager with minimal experience with IoT able to set up an Arduino and interpret its data when provided instructions and pre-written code?

3) Does a low-cost Industry 4.0 system generate valuable data for a wood product manufacturer where most processes are done manually?

5.1 Discussion of the Test Results

The following discussion relates to the results from testing sensors at the Assembly Station at Otretek AS, as well as usability testing with employees from Otretek AS and Nasjonalparken Næringshage in Oppdal.

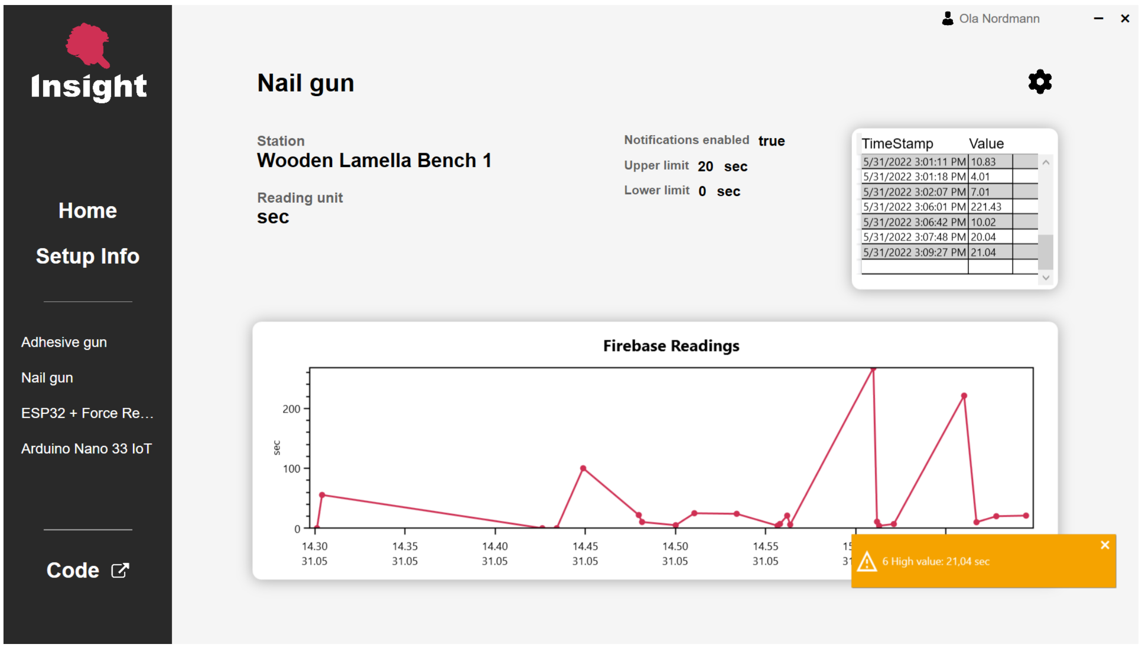
5.1.1 Sensor Testing

The sensor tests were conducted at the Assembly Station at Otretek AS. The station was selected as the segment of the production line to focus on, as it had several factors which could possibly be monitored using sensors. While this was also the case for all of the other segments, the Assembly Station a variety of different product assemblies and large variations between the work routines of the employees. There were also a wooden jig and three different types of tools that could potentially have sensors on them. Another factor was that there was a problem with cracking wood, which was rarely uncovered before the assembly began at the Assembly Station, as this happened after varnishing or staining. These types of losses belong between the defined segments or stations of the production line.

The test results from the pneumatic adhesive gun and pneumatic staple gun clearly show when the tools are in use. The conceptual prototypes are therefore deemed viable and can be further developed in order to refine the solution. The largest drawback of the prototypes as they appear at the moment, is that the container and power bank can be reduced in size or better tailored to the tools. The Arduinos fell off both the adhesive gun and staple gun at the beginning of the testing, and were then secured with tape.

5.1.2 Software Testing

The usability testing of the software gave a clear indication that the use of the software is not relevant for all employees in the small- and medium-sized enterprise. While some subjects showed great interest when answering the Pre-test survey, others showed little interest and questioned the solutions relevance for their work. The System Usability Scale gave a mean score of 60,8, which is below what (Bangor et al., 2008) describes is the value for passable products. Some changes were therefore quickly implemented after the qualitative feedback from the testing. The plot title which was questioned and decrlared as confusing was removed, as the choice database provider is not relevant for all end-users. Additionally, statistics on the latest readings was included. The changes can be seen from figure 31 to 32.



Nail gun

Station
Wooden Lamella Bench 1

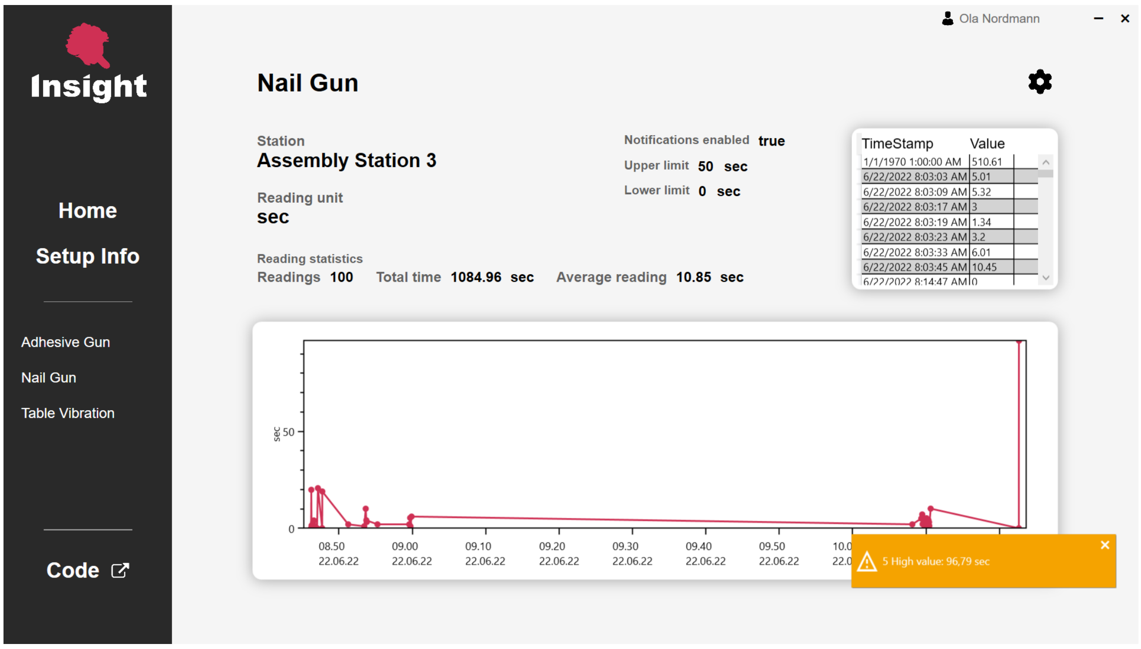
Reading unit
sec

Notifications enabled true
Upper limit 20 sec
Lower limit 0 sec

TimeStamp	Value
5/31/2022 3:01:11 PM	10.83
5/31/2022 3:01:18 PM	4.01
5/31/2022 3:02:07 PM	7.01
5/31/2022 3:06:01 PM	221.43
5/31/2022 3:06:42 PM	10.02
5/31/2022 3:07:48 PM	20.04
5/31/2022 3:09:27 PM	21.04

6 High value: 21,04 sec

Figure 31: The software "Insight" during usability tests, before some features were added.



Nail Gun

Station
Assembly Station 3

Reading unit
sec

Reading statistics
Readings 100 Total time 1084.96 sec Average reading 10.85 sec

Notifications enabled true
Upper limit 50 sec
Lower limit 0 sec

TimeStamp	Value
1/1/1970 1:00:00 AM	510.61
6/22/2022 8:03:03 AM	5.01
6/22/2022 8:03:09 AM	5.32
6/22/2022 8:03:17 AM	3
6/22/2022 8:03:19 AM	1.34
6/22/2022 8:03:23 AM	3.2
6/22/2022 8:03:33 AM	6.01
6/22/2022 8:03:45 AM	10.45
6/22/2022 8:14:47 AM	0

5 High value: 96,79 sec

Figure 32: The software "Insight" during sensor testing, after feature requests from usability testing had been worked on.

5.1.3 On Ecological Validity

The argument can be made that as the sensor testing was conducted during normal working hours and with an exprienced employee producing real orders, the ecological validity for this test is high. The primary thing which could have affected the ecological validity to some extent was the presence of an observer during the test. There were also other employees who approached the test environment during the test due to curiosity. Previous visits have shown that the standard work environment at the Assembly Station is somewhat social, meaning that the test environment was hopefully not too divergent. The observation point-of-view is shown again in 33.

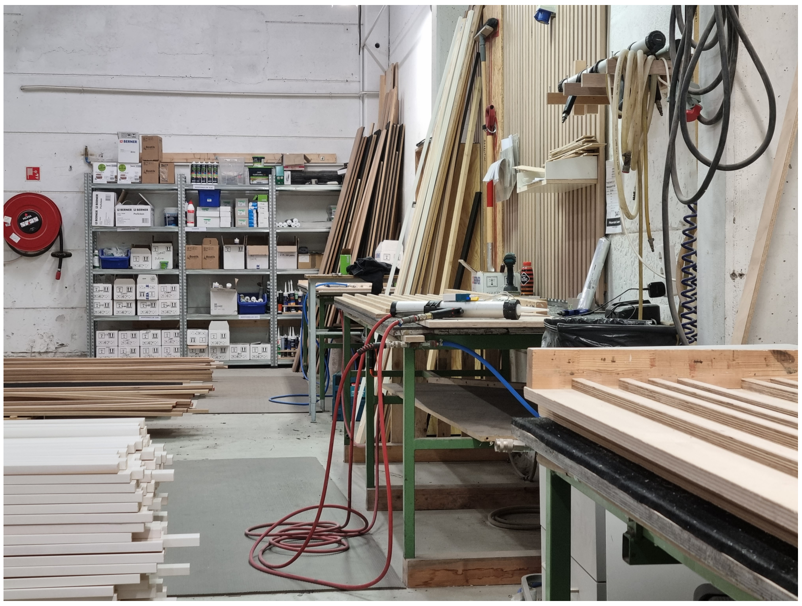


Figure 33: The point-of-view for test observation.

5.2 Addressing the Research Questions

The first research question asks whether data from low-cost Industry 4.0 solutions can give an accurate representation of the activity conducted at a wood product assembly station. To this question, the answer is a clear *yes*. The plots from section 3.4.2.

The second research question asked whether a production manager is able to set up and interpret the telemetry received from an Arduino when provided instructions and pre-written code. To this the answer is *maybe*. While all but one subject in the usability testing was able to figure out how to set up the provided Arduino and get readings in "Insight", several questions regarding the setup process were posed by the subjects. These ranged from not understanding the some of the terminology used in the guide, to not understanding the readings coming from the device. A solution to this can for example be to pair the initial uses of this IoT system with a course at the local makerspace.

The last research questions asks whether a wood product manufacturer will find value in low-cost Industry 4.0 solutions like the one presented in this thesis, and to that the answer is also *maybe*. For this to be the case, there must exist a wish or need to collect data about the production. At several points in the process, management expressed that they had confidence in the work the employees were doing and they did not provide any specific pain points which could be solved by adding Industry 4.0 solutions. This would therefore mean that such solutions add a new dimension to the production line, and the potential in this dimension must be understood by management.

5.3 Further Work

Some weaknesses in the code is apparent from the presented plots. The fact that there are several short readings indicating one ”use” of a tool requires some tinkering and testing of the code, and should be fixed relatively fast. The type of plot used to show time-based readings should be changed. One option is to remove lines between the reading points, another is to change the type of plot to for example a box plot. This should increase the user experience.

To incorporate new sensors will come at the cost of ease-of-use of the system. As the sensor prototype currently relies on the integrated inertial measurement unit of the Arduino Nano 33 IoT, any new sensors would need either wiring or a new development board with the specified sensor integrated. The cost of this is an evaluation that the end-user has to make, as it lays outside of the scope of the setup guide in the software. It might be relevant to expand the guide in the future, so that users can use ”Insight” to gain more in-depth knowledge in IoT and programming of microcontrollers.

An interesting continuation of the project would be to incorporate machine learning in the software. This will only be feasible once the code is at a point where the datapoints have few faulty readings and the patterns clearly show the different stages of assembly. As artificial intelligence and machine learning clearly fits into the realm of Industry 4.0, providing users with an AI-based tool that is not hidden behind a subscription model can be of relevance. This might mean that the software will require more in-depth training for proper usage, as it would become much more complex. This type of integration might be outside the scope of open-source projects of client desktop applications, and might push the development towards a scale that requires full-time employees and usage fees. It would also possibly push the solution outside of what the employees are able to spend time on learning and implementing.

6 Conclusion

This thesis has been an investigation of whether it is possible to enable the use of low-cost Industry 4.0 prototypes in small- and medium-sized enterprises. The three research questions posed at the beginning of this thesis have all been discussed, and show promising results especially when it comes to the sensors prototyped for the Assembly Station. The software will be further developed, as both the setup process and the features are somewhat lacking as it stands today.

The work in this thesis show that it is possible to implement low-cost Industry 4.0 solutions in a small- and medium-sized enterprise and that the production managers are capable of setting up such a system on their own, with a little bit of guidance.

Bibliography

Arduino. (2022). *Nano 33 IoT Documentation.* https://docs.arduino.cc/hardware/nano-33-iot

Bangor, A., Kortum, P. T. & Miller, J. T. (2008). An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction*, *24*(6), 574–594.

Bartodziej, C. J. (2017). The concept industry 4.0. *The concept industry 4.0* (pp. 27–50). Springer.

Beck, K., Beedle, M., van Bennekum, A. et al. (2001). *The Agile Manifesto.* https://agilemanifesto.org/principles.html

Berisha, G. & Pula, J. S. (2015). Defining small and medium enterprises: A critical review. *Academic Journal of Business, Administration, Law and Social Sciences*, *1*(1), 17–28.

Brooke, J. et al. (1996). Sus-a quick and dirty usability scale. *Usability evaluation in industry*, *189*(194), 4–7.

Castelo-Branco, I., Cruz-Jesus, F. & Oliveira, T. (2019). Assessing industry 4.0 readiness in manufacturing: Evidence for the european union. *Computers in Industry*, *107*, 22–32.

Condes, E. (2022). *ArduinoFFT Library.* https://www.arduino.cc/reference/en/libraries/arduinofft/

Doorley, S., Holcomb, S., Klebahn, P., Segovia, K. & Utley, J. *Design thinking bootleg.* Distributed to students at d.school. 2018, July.

Doorley, S., Holcomb, S., Klebahn, P., Segovia, K. & Utley, J. (2018b). Design thinking bootleg.

Figma. (2022). *About Figma.* https://www.figma.com/about/

Gamma, E., Helm, R., Johnson, R., Johnson, R. E., Vlissides, J. et al. (1995). *Design patterns: Elements of reusable object-oriented software.* Pearson Deutschland GmbH.

Gerstenberg, A., Sjöman, H., Reime, T., Abrahamsson, P. & Steinert, M. (2015). A simultaneous, multidisciplinary development and design journey–reflections on prototyping. *International Conference on Entertainment Computing*, 409–416.

Guérin, F., Guinand, F., Brethé, J.-F., Pelvillain, H. et al. (2016). Towards an autonomous warehouse inventory scheme. *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1–8.

Guerreiro, B. V., Lins, R. G., Sun, J. & Schmitt, R. (2018). Definition of smart retrofitting: First steps for a company to deploy aspects of industry 4.0. *Advances in manufacturing* (pp. 161–170). Springer.

Houde, S. & Hill, C. (1997). What do prototypes prototype? *Handbook of human-computer interaction* (pp. 367–381). Elsevier.

IEEE. (2000). Ieee recommended practice for architectural description for software-intensive systems. *IEEE Std 1471-2000*, 1–30. https://doi.org/10.1109/IEEESTD.2000.91944

Johnson, G., Scholes, K. & Whittington, R. (2008). *Exploring corporate strategy: Text and cases.* "Pearson Education".

Koen, P. A., Ajamian, G. M., Boyce, S., Clamen, A., Fisher, E., Fountoulakis, S., Johnson, A., Puri, P. & Seibert, R. (2002). Fuzzy front end: Effective methods, tools, and techniques. *The PDMA toolbook*, *1*, 5–35.

Kruchten, P. B. (1995). The 4+1 view model of architecture. *IEEE software*, *12*(6), 42–50.

Lai, Z.-H., Tao, W., Leu, M. C. & Yin, Z. (2020). Smart augmented reality instructional system for mechanical assembly towards worker-centered intelligent manufacturing. *Journal of Manufacturing Systems*, *55*, 69–81.

Leifer, L. J. & Steinert, M. (2011). Dancing with ambiguity: Causality behavior, design thinking, and triple-loop-learning. *Information Knowledge Systems Management*, *10*(1-4), 151–173.

Lim, Y.-K., Stolterman, E. & Tenenberg, J. (2008). The anatomy of prototypes: Prototypes as filters, prototypes as manifestations of design ideas. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *15*(2), 1–27.

Lindberg, T., Meinel, C. & Wagner, R. (2011). Design thinking: A fruitful concept for it development? *Design thinking* (pp. 3–18). Springer.

Müller, J. M. (2019). Business model innovation in small-and medium-sized enterprises: Strategies for industry 4.0 providers and users. *Journal of Manufacturing Technology Management.*

Munro, D. (2013). *A guide to sme financing.* Springer.

NHO. (2018). *Fakta om små og mellomstore bedrifter.* https://www.nho.no/tema/sma-og-mellomstore-bedrifter/artikler/sma-og-mellomstore-bedrifter-smb/

Oliveira, T., Martins, R., Sarker, S., Thomas, M. & Popovič, A. (2019). Understanding saas adoption: The moderating impact of the environment context. *International Journal of Information Management*, *49*, 1–12.

OpenRemote.io. (2022). *OpenRemote Product.* https://openremote.io/product/

Preece, J., Sharp, H. & Rogers, Y. (2015). *Interaction design: Beyond human-computer interaction, 4th edition.* Wiley.

ProtonTechnologies. (2022). *Art. 4 GDPR: Definitions.* https://gdpr.eu/article-4-definitions/

Safizadeh, M. & Latifi, S. (2014). Using multi-sensor data fusion for vibration fault diagnosis of rolling element bearings by accelerometer and load cell. *Information fusion*, *18*, 1–8.

Schrage, M. (1996). Cultures of prototyping. *Bringing design to software*, *4*(1), 1–11.

Schröder, C. (2016a). The challenges of industry 4.0 for small and medium-sized enterprises. *Friedrich-Ebert-Stiftung: Bonn, Germany.*

Schröder, C. (2016b). The challenges of industry 4.0 for small and medium-sized enterprises. *Friedrich-Ebert-Stiftung: Bonn, Germany.*

ScrumGuides.org. (2020). *The Official 2020 Scrum Guide.* https://scrumguides.org/scrum-guide.html

Smith, J. (2009). *Patterns - WPF Apps With The Model-View-ViewModel Design Pattern.* https://docs.microsoft.com/en-us/archive/msdn-magazine/2009/february/patterns-wpf-apps-with-the-model-view-viewmodel-design-pattern

Sommerville, I. (2016). *Software engineering.* Pearson. https://www.pearson.com/us/higher-education/program/PGM35255.html

Steinert, M. & Leifer, L. J. (2012). 'finding one's way': Re-discovering a hunter-gatherer model based on wayfaring. *International Journal of Engineering Education*, *28*(2), 251.

ThingsBoard.io. (2022). *ThingsBoard.* https://thingsboard.io/

Tulip.co. (2022). *About Tulip.co.* https://tulip.co/about-us/

Ulrich, K. T. & Eppinger, S. D. (2016). *Product design and development.* McGraw-Hill Education.

Virzi, R. A. (1992). Refining the test phase of usability evaluation: How many subjects is enough? *Human factors*, *34*(4), 457–468.

Williams, S. (2011). *Free as in freedom [paperback]: Richard stallman's crusade for free software.* " O'Reilly Media, Inc."

Yang, H. & Yang, S.-H. (2009). Connectionless indoor inventory tracking in zigbee rfid sensor network. *2009 35th Annual Conference of IEEE Industrial Electronics*, 2618–2623.

Appendices

Appendix A: Project Thesis

Department of
Mechanical and Industrial Engineering

TMM4560 - Engineering Design and Materials

Preliminary Work for Introducing Industry 4.0 in Acoustic Wooden Lamella Manufacturing

Author
Hanna Aksetøy Aalmen

Main suprevisor
Federico Lozano
Co-supervisor
Martin Steinert

January, 2022

NTNU
Kunnskap for en bedre verden

TrollLABS

Abstract

This thesis presents the preliminary work for implementing Industry 4.0 concepts in small- and medium-sized enterprises. The project surrounds Otretek AS and their production of Acoustic Wooden Lamellas, which at its current state is a very manual process. The work has been done through needsfinding, prototyping and testing, and has resulted in three prototypes within the scope of Industry 4.0 that will be worked on in the spring of 2022.

Table of Contents

List of Figures

List of Tables

1 Introduction

This project thesis presents an exploration of how to implement elements from Industry 4.0 in small- and medium-sized enterprises, and to which extent it is reasonable to implement it. Otretek AS is a medium-sized wood product manufacturer from Oppdal, Norway. They were chosen for the thesis due to them communicating a will to improve their most manual assembly lines and a willingness to take part in prototype testing.

Throughout the autumn of 2021 there have been four visits to Otretek AS. As a way to standardize the output of the needsfinding and potential prototyping, all of the visits have been documented through photographs and written logs. Monitoring production and testing prototypes for improvement of production of the Acoustic Wooden Lamella was, in addition to conversations, the main objectives of the visits.

The prototyping for the project thesis was partly worked on in Oppdal, which as of 2021 hosts its own makerspace in the KRUX Innovation Center. The makerspace is part of Nasjonalparken Næringshage, a "Business Garden" facilitating innovation and local development, and is intended as a resource for the local population. The other part has been executed at the TrollLABS makerspace at the Norwegian University of Science and Technology in Trondheim.

The scope of this project is to perform and document needsfinding and prototyping in order to integrate Industry 4.0 concepts in the production of the Acoustic Wooden Lamella. This thesis is considered preliminary work on the subject, as the prototyping will continue in the spring of 2022.

Section 2 of this document presents general theory for the thesis. Section 3 and 4 covers the background, prototyping and testing for the two main objectives of the thesis. The former centers around finding a way to measure Otretek's manufacturing process, and the latter centers around improving this process. Section 5 covers the discussion in the thesis, and Section 6 presents the conclusion. Code used for the prototypes is found in the Appendix.

2 Theory

This section goes in-depth on the theory which forms the basis for new product development and prototyping. It also describes the background of the company being focused on in the work for the project thesis. Theory which relates directly to one of the two prototyping objectives is covered in sections 3.1 and 4.1.

2.1 Engineering Design

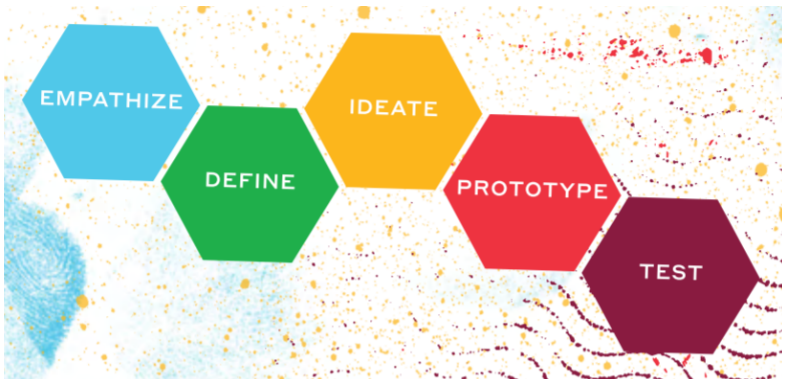
Several methodologies fall under the engineering design term, ranging from agile product development to the Waterfall method. Ulrich and Eppinger (2012) defines product development as ”the set of activities beginning with the perception of a market opportunity and ending in the production, sale, and delivery of a product” (Ulrich, 2016, p. 2). Engineering design covers technical product design and development in the product development process (Ulrich, 2016, p. 20), from initial needsfinding to prototype testing and evaluation.

2.1.1 Design Thinking

Design Thinking is a methodology for concept creation and product development consisting of the components “empathize”, “define”, “ideate”, “prototype” and “test”. The methodology emphasizes “playing with ambiguity” in novel concept creation and allows for the product developers to explore several ideas and subsequently get feedback over a short period of time (Jensen et al., 2016).

Especially the first component points to an increased focus on user interaction and the importance of getting user input throughout the product development process. Empathizing with the user can happen through for example casual conversation or observation, with the goal of immersing in the user’s thoughts and actions. This belongs in the needsfinding phase of product development. One approach is to start asking questions at a beginner level on the user’s actions or field being researched. Assuming a beginner’s mindset can aid in facilitating an open mindset for the product developer, a higher level of curiosity and better immersion (Doorley

Figure 1: The Design Thinking process visualized. (Doorley et al., 2018).



et al., 2018).

A generative mindset is of importance at several points in the design thinking process (Doorley et al., 2018) and might be particularly fruitful in question asking. Asking questions as part of the design thinking process is fundamental and keeping them generative in nature can be of special importance at multiple points in the design thinking process. Eris et al., 2003 identifies ideation as its own category of generative questions, coinciding with the third component of the process. Design thinking often ends up in a diverging and converging cycle, thus showing parallels to methodologies and processes such as agile product development and Wayfaring.

2.1.2 Prototypes and Prototyping

Ulrich and Eppinger (2012) defines a prototype as ”an approximation of the product along one or more dimensions of interest” (Ulrich, 2016, p. 293), whilst the act of prototyping is the development process of said prototype. There are several ways to define prototypes and their purpose.One example is the difference between high-fidelity prototypes, which are more comparable to a finished product, and low fidelity prototypes which have the required functionality, but are far from the market-ready version. Another example is the definition by Ulrich, 2016 of analytical prototypes as intangible, often in the form of mathemathical simulations or 3D models, while physical prototypes are tangible models which approximate part of or the full idea being built.

In *What do Prototypes Prototype?*, Houde and Hill, 1997 proposes a model for answering some fundamental questions that prototypes pose. While the article centers around prototyping in software development, its definitions and discussions are suitable for physical prototypes in engineering design as well. The proposed model asks about the role of the prototype in a user's life, its look and feel and how its function is implemented. Houde sees this as a way of escaping ambiguous definitions such as high-fidelity and low-fidelity prototypes, which can have different meanings depending on the circumstances. Such circumstances can for example be tied to the prototyping culture in an organisation, as described by Schrage, 1996.

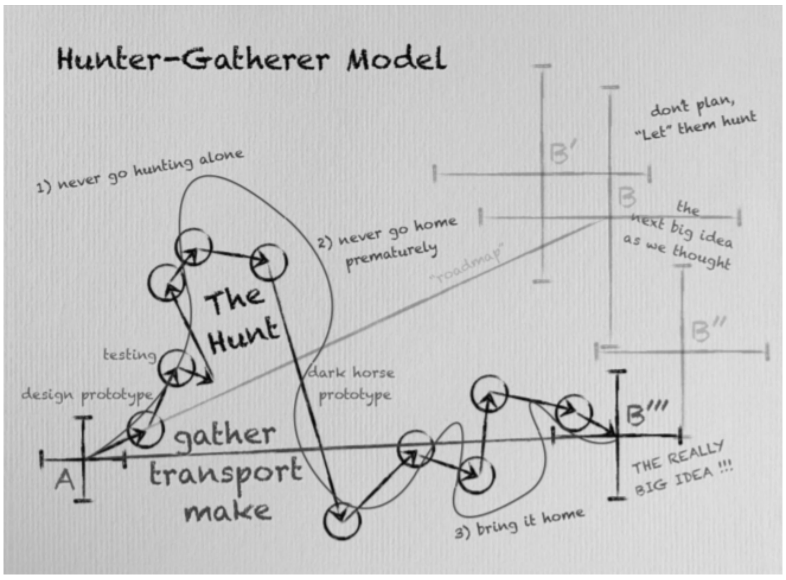
2.1.3 The Early-stages of Innovation Processes and Wayfaring

Koen et al., 2002 divides the innovation process in three parts, the fuzzy front-end, new product development and commercialization. The fuzzy front-end describes a stage with high uncertainty and ambiguity. Drawing from design thinking, this is the stage where the product developers define the user, the problem or task at hand and start ideating towards creating prototypes. The fuzzy front-end is the phase of exploration, and as stated by Elverum et al., 2014, this is the phase in new product development with the largest potential for cost reduction and product improvement.

Prototypes are developed in the new product development phase, which is some cases can have an overlap with the fuzzy front-end (Elverum et al., 2014). Jensen et al., 2017 defines the concept *prototrial* as early-stage prototypes specifically meant to elicit unknown unknowns in the early phases of a design process. Unknown unknowns being a lack of knowledge on a subject first aquired during the development process. Known unknowns are uncertainties and missing knowledge that you know about beforehand. *Prototrials* can be helpful in uncovering unknown unknowns as they cover the functionality of the idea, without being full-fledged prototypes.

The Wayfaring journey as described by Steinert and Leifer, 2012, is a hunt for the next big idea. It is a process characterized by high ambiguity, pulling this aspect from the fuzzy front-end into the new product development process. The Wayfaring model bids the product developers to explore the entire solution space through a series of diverging and converging activities. In the Hunter-Gatherer Model proposed by Steinert and Leifer, 2012, the diverging activities are described as *hunting* and the converging activities as *gathering.* At a point during the process a *dark horse prototype* should be developed, that is a prototype expanding the solution space in a new direction. They further define three rules for the Wayfaring. "Never go hunting alone", "Never go home prematurely" and "Bring it home" (Steinert and Leifer, 2012, p. 1-2). Each rule is referring to a stage in the Wayfaring journey as seen in the figure below, where the final goal is to arrive at *the really big idea.*

Figure 2: Wayfaring through the Hunter-Gatherer model (Steinert and Leifer, 2012).



2.2 Industry 4.0 in Small and Medium-sized Enterprises

Industry 4.0, often referred to as the fourth industrial revolution, points to the current era in technological progress in industry. Lasi et al., 2014 characterizes Industry 4.0 as having an application-pull and a technology-push as its driving forces. The application-pull originates from several factors, the most prominent being a focus on iterative innovation processes, a focus on flexibility in production and a change from a seller's to a buyer's market. A buyer's market indicates that there is a strong preference for bespoke solutions, which in turn is made possible due to shorter development times and more flexibile solutions. The technology-push stems from three main factors, which is an increased focus on automation, increased digitalisation and networking and miniaturization. Miniaturization refers to computing more in less space than before.

The term *smart factory* consists of fours dimensions as defined by Frank et al., 2019: Smart Manufacturing, Smart Products, Smart Supply Chain and Smart Working. Furthermore, four base tehnologies are defined, all falling under the Smart Manufacturing dimension: internet of things, cloud services, big data and analytics. I their study, Frank et al., 2019 found that especially big data and analytics were of little prevalence across the studied manufacturing companies. In a 2018 study from Brazil, Dalenogare et al., 2018 found similar results regarding big data and analytics and tied it to a lack of knowledge on cyber security and lack of resources for data storage.

In their article, Pech and Vrchota, 2020 classifies small- and medium-sized companies according to their level of Industry 4.0 integration on an index scale. The study concludes that even though there exists some degree of Industry 4.0 implementation in most of the small- and medium-sized companies sampled, this mainly concerns low level technology such as cloud storage, data collection and analysis. High level technology such as machine learning and virtual reality is mainly a feature in large enterprises. Sevinc et al., 2018 concluded in their study that the cost of technology investments and little knowledge on the competitive returns are the main reasons for low degree of Industry 4.0 implementation in their sampled companies.

Making manufacturing smarter can be seen as a competitive advantage among small- and medium-sized companies, especially due to the small prevalence of intermediate and high level technology (Pech and Vrchota, 2020).

2.3 Introducing Otretek AS

Otretek AS is a wood product manufacturer with its facilities situated in the rural town of Oppdal, Norway. With only one location, the entire process from product development to sales is executed in Oppdal. Otretek AS was ranked number 29 out of all companies in Sør-Trøndelag in the awarding of the title "Gasellebedrift" in 2013 by Dagens Næringsliv (Tøsse, 2013). The title is awarded companies who amongst other criteria can show a doubling of revenue over the last four years and has an overall positive operating result (Næringsliv, 2022). The company was also rewarded the local title “Årets bedrift”, or enterprise of the year in English, by the local bank Oppdalsbanken and the newspaper Opdalingen in 2017. “Årets bedrift” is meant to promote businesses in Oppdal and the neighboring commune Rennebu. The reasoning for giving Otretek the title in 2017 was due to its heavy foundation in the local community, good financial results, and generally good work ethics. The title holder is considered a role model for local business development (Silseth Naas, 2017).

2.3.1 The Scale of the Enterprise

In a town of about seven thousand inhabitants, an enterprise with 48 employees will leave a considerable footprint (Proff.no, 2022). The Confederation of Norwegian Enterprise (NHO) defines small and medium-sized enterprises (SMEs) as having 100 employees or less. Small enterprises have 1-20 employees, while medium-sizes enterprises have 21-100 employees. By the Norwegian definition Otretek is a medium-sized enterprise with its 48 employees. In numbers from 2018, the Confederation indicates that SMEs make up over 99 percent of all companies in Norway and stands for almost half of the value creation among Norwegian businesses (NHO, 2018).

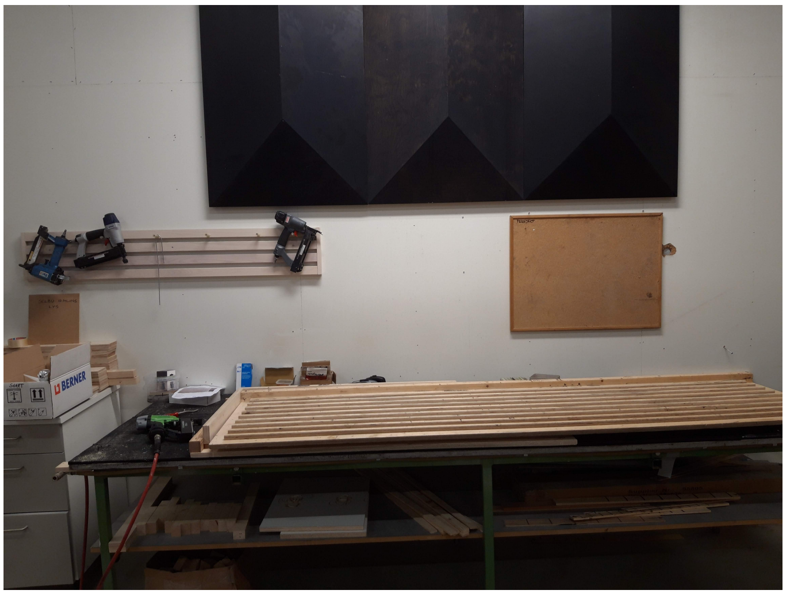
2.3.2 Manufacturing of the Acoustic Wooden Lamella

Acoustic Wooden Lamellas consist of two main elements – the wooden lamellas and a PET fiber backplate. Otretek AS offers several variations of lamellas with different backplates, depending on the area of mounting and the product attributes. The Acoustic Wooden Lamella is one of the company’s best-selling products, and they are the supplier of this product for one of Norway’s largest building material chains, Byggmakker. In 2020 they estimated a revenue of 1 million NOK from the supplier deal with Byggmakker, yet ended up with a result of approximately 8 million NOK (M. Thomassen, personal communication, November 11th 2021). The Acoustic Wooden Lamella has a production cost per panel of 267 NOK and a retail price of 1551 NOK when bought directly from Otretek AS (M. Thomassen, personal communication, December 15th 2021).

The production hall at Otretek AS houses several zones which serve different purposes. In addition to the wooden lamella manufacturing, they also assemble kitchen solutions which they receive from an off-site supplier and design and build more specialized wooden constructions. The latter is often related to extensive projects where they also supply wooden lamellas. The facility has a variety of machinery, some of them from recent years, others stemming from the wood working company Stompa, located in the same industrial area until it was closed in the early 2000s (J. Langseth, personal communication, August 31st 2021).

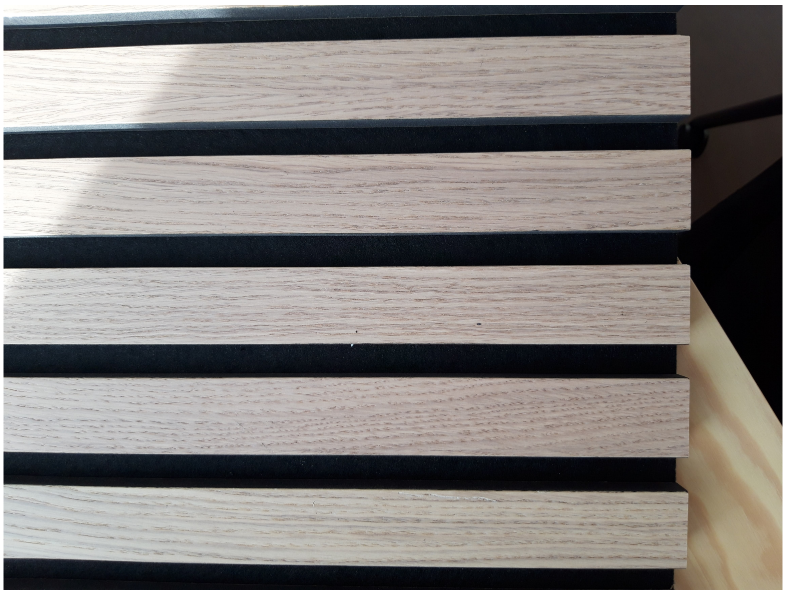
The production of Acoustic Wooden Lamellas takes place in a zone with four workbenches dedicated to the task. Each workbench has a section on the floor for unassembled material, and one where the assembled Acoustic Wooden Lamellas go. On the workbench there is a jig made specifically for the wooden lamella with backplate that they are manufacturing. The workbenches also have air pressure hoses which are coupled with the pressurized adhesive applicators and staple or nail guns being used in the assembly. A simple overview of the zone is shown in Figure 3.

Figure 3: A workbench in the manufacturing hall.



The Acoustic Wooden Lamellas are assembled in jigs measuring 60,5 x 240 centimeters, where the panels consist of eight lamellas each measuring 3,5 centimeter in width with a 1,4 centimeter wide gap, a PET-fiber backplate covering the full length and width of the jig, adhesive and staples. See Figure 4 for illustration. The assembly process consist of laying down lamellas in the jig, adding adhesive to all lamellas, putting the PET-fiber backplate on top and stapling it to the lamellas, and removing the assembled product from the jig.

Figure 4: The Acoustic Wooden Lamella.



3 Measuring Manufacturing Times

This section covers the background for why measuring manufacturing times is interesting, the prototypes developed for the objective and testing of certain prototypes. The section ends with a discussion of the test results.

3.1 Background

Tools for measuring manufacturing times have become more available as the interest in Internet of Things and sensors has increased over the years. Sensors are on the market for a couple of dollars, and there are a lot of free programming courses and projects online covering a range of programming languages and subjects. Low-cost sensors can be used to create valuabe systems, as suggested by Reinhart, 2004 and Oliviero et al., 2008. Incorporating them in manufacturing processes as a step towards low-cost Industry 4.0 might be the way to go for small- and medium-sized enterprises like Otretek AS.

3.1.1 Time Measurements for Benchmarking

Time measurements in manufacturing processes is one example of internal benchmarking in a company. Benchmarking is the act of evaluating ones performance and comparing it to those who one sees as similar to oneself. Baba et al., 2006 proposes a framework for benchmarking in small- and medium-sized enterprized, emphasizing how a standardization will cause an improvement. An automated system within the sphere of Industry 4.0 can be a step towards this, for example by incorporating sensors that log data to a cloud.

3.1.2 Retrofitting Machinery

According to Al-Maeeni et al., 2020, retrofitting ”is the process of optimizing the accuracy, speed, maintainability, and ease of use of an old machine”. This is done in order to optimize the general performance. Retrofitting in the traditional sense can for example mean equipping machinery og equipment with sensors to make monitoring possible. In the context of Industry 4.0, more high-tech approaches have also been proposed. Smart retrofitting is the concept suited for Industry 4.0, and focuses on expanding the sensors from traditional retrofitting with computers for for example machine learning integration (Al-Maeeni et al., 2020).

3.1.3 Findings from Company Visits

As the supplier of Acoustic Wooden Lamellas for the Byggmakker chain, a reasonable portion of the wooden lamella production will consist of Acoustic Wooden Lamellas. During the first visits in August and September of 2021, the Production Manager informed that at least one out of four workstations should always be dedicated to this product. This would also benefit the work with this thesis if any equipment had to be placed on the workstation.

The first two visits gave some insight into the equipment used in Otretek's manufacturing. Larger machinery had noticeable variations in age and complexity, with some of the machinery in use originating from the Stompa factory mentioned in section 2 Theory, making it approximately 20 years old. This was mostly the case for low-complexity machines like table saws and similar wood-cutting tools, and as communicated by the Production Manager, showed the reliability and reusability of this type of low-complexity machinery. One aspect with equipment and machinery that the Production Manager expressed some frustration with, was the quality of the smaller tools that they used for wooden lamella assembly. These tools, despite being in the price range of a couple hundred dollars, often had to be replaced annually (M. Thomassen, personal communication, October 13th 2021). Through observation, several of the employees had issues handling some of the tools as well. This was especially the case for the combined staple and nail guns, which became jammed two times during a 40-minute observation period. A coworker had to step in to get the tool working again, which distracted both workers for approximately five minutes each time. The Production Manager later informed that this specific brand of staple and nail gun had to be replaced several times a year (M. Thomassen, personal communication, October 13th 2021).

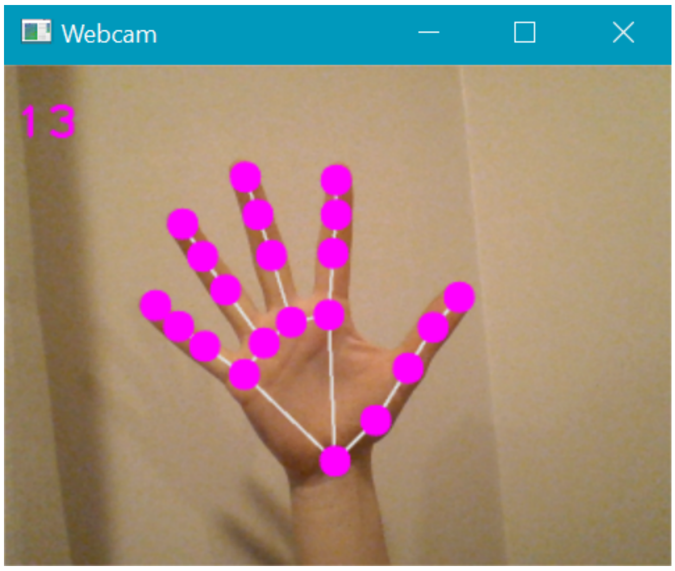
3.2 Wayfaring for Manufacturing Time Measurements

Wayfaring was used for the prototyping towards manufacturing time measurements. The development for monitoring the production resulted in three prototypes. The prototypes belonging to this section on monitoring production has an “M” in their name. This section provides info about and images of each prototype, as well as a final subsection reviewing the prototypes.

3.2.1 Prototype M1 - Tracking Using Computer Vision

There exist several solutions that enable relatively seamless integration of computer vision in projects, of course with the requirement that some programming skill is in place. Several of these solutions exist as Open-Source packages available for a variety of programming languages, one example being Google's machine learning solution for media files and streaming, MediaPipe. MediaPipe was used to test the viability of using hand-tracking to time the manufacturing of Acoustic Wooden Lamellas. The data collected would have been analyzed by upper management or the Production Manager, who is regarded as the user for this prototype. The figure below shows the result of implementing a pre-trained machine learning algorithm for hand tracking on streamed video footage from a web camera.

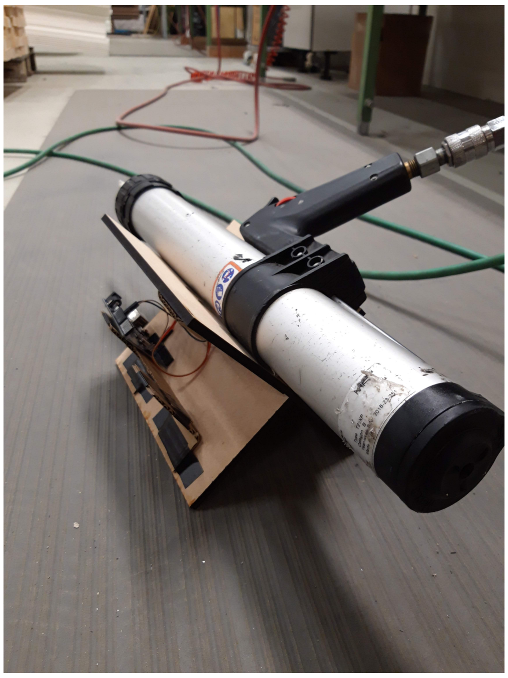
Figure 5: Hand tracking using Google's Mediapipe Python package.



3.2.2 Prototype M2 - Tool-in-use Using Ultrasonic Distance Sensor

Sensors for developing solutions within the subject of Internet-of-Things are often low-cost and relatively simple to wire, especially when using helper tools for wiring circuits. Arduino is a commonly used microcontroller for such projects, which for example can be combined with a laptop or smaller computer like the RaspberryPi for further data storage or processing. The HC-SR04 Ultrasonic Distance Sensor measures the distance between the sensor itself and objects ahead with a working range between 2 and 400 centimeters. It is on the market for about 3 USD. This prototype consists of four laser cut MDF parts, an ultrasonic distance sensor, an Arduino and the necessary wiring. The goal of the prototype is to log when a tool is docked in the tool holder, thereby measuring how long they are in use during the day.

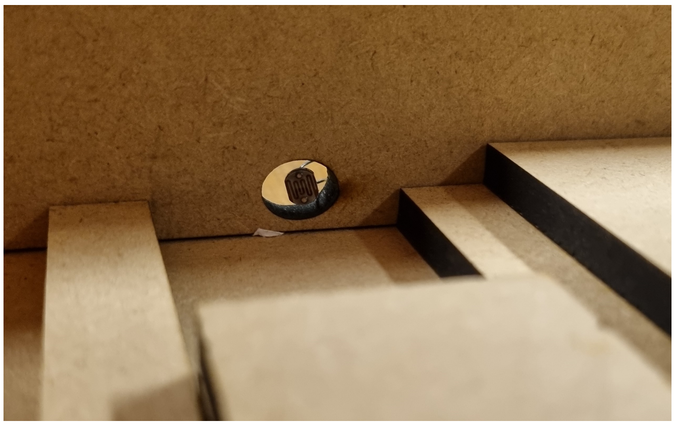
Figure 6: Laser cut docker for tool using Ultrasonic Distance Sensor.



3.2.3 Prototype M3 - Lamella Placement Using Photocells

Photocells are another type of sensor which are low-cost and can be wired to an Arduino or other microcontrollers to create a small monitoring system. This prototype was designed to replace one of the two border planks of Otretek's wooden lamella jigs, as presented in the figure below. Photocells register the amount of light received by the cell through a changing resistive value, and in this case serves the purpose of detecting when a lamella is placed in the jig. Pairing the output value with a timestamp does in this case work for gathering lamella placement times.

Figure 7: Photocell mounted to the mock jig for detecting lamella placement.



3.2.4 Prototyping Outcome

It became clear during the early development of prototype M1 that writing code for tracking hands in different situations could become quite time-consuming. Wearing dark protective gloves, something several employees did during the visits, can for example prevent proper shape detection which is critical for the algorithm to work properly. It would also have to work when the employees' hands were holding different tools or materials, which posed another problem to solve. A pivot was therefore made after some initial testing, as a more sensor-driven approach seemed more promising.

Both prototype M2 and M3 had a development time of approximately three hours, not taking testing and data collection into consideration. These prototypes were made with a higher focus on rapid prototyping yet ended up working during initial testing without further iterations. During the testing documented in the next section, both prototypes M2 and M3 were set up to gather data. Test setup where both of the prototypes are wired together is also shown.

3.3 Testing prototypes M2 and M3

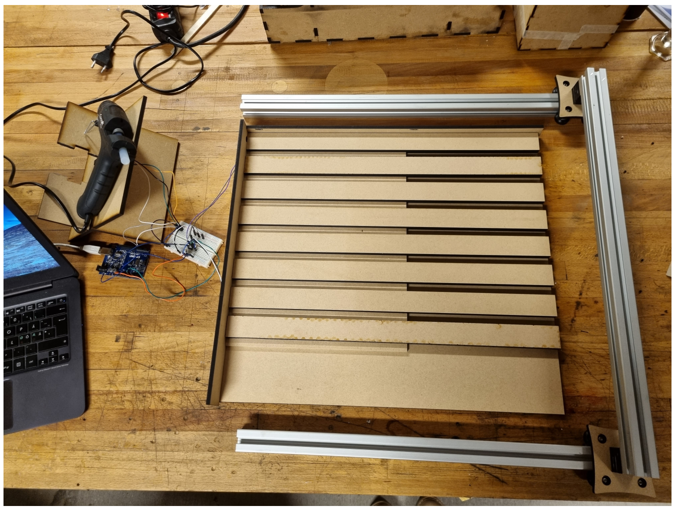
The main objective of this test is to confirm that the setup manages to capture the key events of the assembly, as described in Section 2.3.2. To specify, prototype M2 should be able to detect when a tool is in use and not, and prototype M3 should be able to detect when the first and last lamella is placed.

This section describes of the method used for testing the two prototypes which were evaluated as being viable during prototyping, as well as the result of the test. The code run on the Arduino is located in the Appendix.

3.3.1 Test Setup

The test simulates assembly using a mock jig and is not performed in the manufacturing hall at Otretek AS. It is set up so that one individual can replicate the process of picking up and putting down tools, as well as lay down lamellas in the jig. The process follows some of the key points of the assembly, listed in Table 1. At least three iterations of the process should be performed.

Figure 8: Test setup for measurement testing.



Threshold values defined in the Arduino code for the photocells are set based on the output values observed during the initial development. Table 2 shows the threshold values for the two types of sensors used.

Table 1: Testing process used to evaluate performance of prototypes M2 and M3.

Step	Action
1	Add eight lamellas to the slots in the jig
2	Pick up adhesive dispenser from M2 prototype.
3	Trace dispenser along the full length of all lamellas.
4	Put down adhesive dispenser in M2 prototpye.
5	Remove lamellas from the slots in the jig.

Table 2: Thresholds for M2 and M3 prototype sensors.

Sensor	Threshold	Description
HC-SR04	3.0 cm	All registered distances under 3.0 cm are evaluated as a docked tool. The working range starts at 2.0 cm, and distances below might evaluate to -1.0, which suits the threshold.
Photocell	300	The photocell registers 150 when it is covered and 800 when pointed directly towards the ceiling lights. 300 is considered a safe threshold.

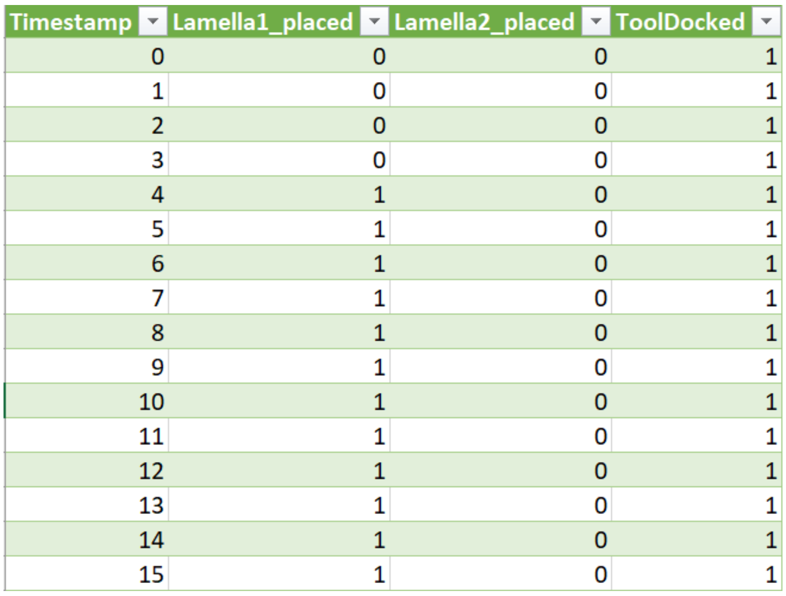
3.3.2 Equipment

A mock jig with lamellas was created early in the semester to enable testing of the prototypes. The mock jig has the same measurements as the Acoustic Wooden Lamella jig at Otretek AS, except for the length which is 50 centimeters as opposed to 240 centimeters for the real jig. As prototypes M2 and M3 are the ones to be properly tested with the mock jig, they have been wired together so that both the HC-SR04 Ultrasonic Distance Sensor and the Photocells are controlled by the same Arduino, passing data to a connected computer. A Python script is used to store the data points logged by the Arduino in CSV format so that it can be viewed in Microsoft Excel. The computer also serves as a 5V power source for the setup.

3.3.3 Results

The test was performed through three iterations of the assembly process shown. The logged data consists of timestamps in one-second intervals with true or false values for the three sensors based on the threshold values. The figure below shows the Excel document after importing the CSV file generated with the Python script in Appendix C. The reasoning for importing it to Microsoft Excel is to make the data easy to handle for people without much programming knowledge, which might be the case for employees at Otretek AS.

Figure 9: The COM port logger output, imported to Excel in CSV format.



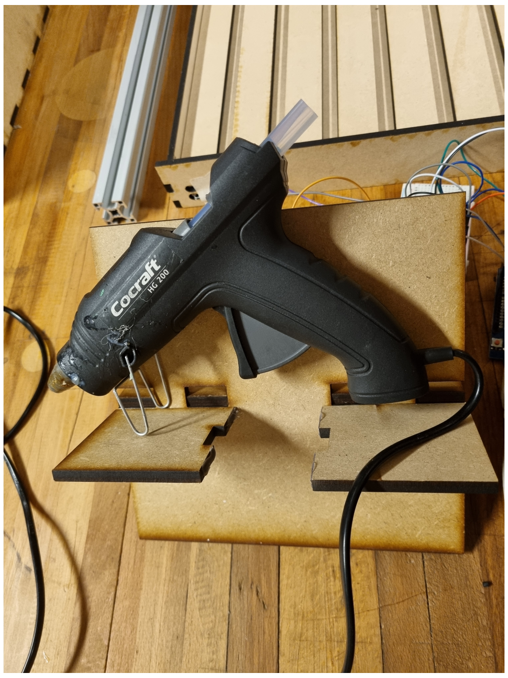
Timestamp	Lamella1_placed	Lamella2_placed	ToolDocked
0	0	0	1
1	0	0	1
2	0	0	1
3	0	0	1
4	1	0	1
5	1	0	1
6	1	0	1
7	1	0	1
8	1	0	1
9	1	0	1
10	1	0	1
11	1	0	1
12	1	0	1
13	1	0	1
14	1	0	1
15	1	0	1

3.4 Discussion of Test Results

One large error during this test was that the HC-SR04 Ultrasonic Distance Sensor failed during the first iteration, only registering true which is the docked state of the tool. The problem persisted through all three iterations, and the backup sensor also failed to work after wiring it to the system. The use of this specific sensor will be discussed further. The first iteration was recorded on video, and the missing values were added based on the recording. Either implementing a graphics package in Python or sending data in another format to Excel should be considered, as the tests created several hundred data points. As easy as Excel is to understand, it is not meant to be used as a database and large numbers of data points are not straightforward to read and can affect computer performance.

Testing should ideally be done by a user with more familiarity to the process in order to get proper feedback. Feedback from the actual user also builds upon the user-centric focus in design thinking. As a proof-of-concept test, this way of doing it was considered adequate under the circumstances. Having a mock jig in both working locations should be considered going forward, instead of moving it back and forth between Oppdal and Trondheim.

Figure 10: The failed M2 prototype.



4 Prototyping Solutions for Improving the Manufacturing Process

This section covers the background used in developing an improved manufacturing process, information on the three prototypes developed for the objective, testing and discussion of the testing.

4.1 Background

Improving the manufacturing process is a task with high ambiguity and several unknown unknowns. This process is therefore strongly based on needsfinding, ideation and the wayfaring method. Findings from the company visits will therefore be the main background for Section 4.

4.1.1 Findings from Company Visits

Already during the first visits the Production Manager informed that a lot of freedom was given in trying to improve the manufacturing process. One of the four workbenches were dedicated to the production of the Acoustic Wooden Lamella, in addition to being declared a testing area for potential prototypes.

One of the most important insights the production manager gave, was the lack of consistency in the wooden lamella assembly. Not only was this caused by the employees often having to change between orders (and therefore jigs and equipment), but there were also large differences in skill and efficiency between the workers. Inreasing consistency could therefore be a good starting point.

When asking the Production Manager which investments he wanted the company to make regarding the manufacturing of the Acoustic Wooden Lamella, he mentioned introducing robot manipulators to the process. He suggested a setup where one manipulator would feed material to the workbench and remove the product once assembled, while another would use the adhesive dispenser and staple gun to assemble the product (M. Thomassen, personal communication, November 11th 2021).

Another aspect of the Acoustic Wooden Lamella that was first mentioned during the late stages of the semester, was that the composition of the product could be changed. The Production Manager informed that some small experiments with other types of glue had been tried out, and that changing adhesive and staple with screws was attempted. The former had shown promising results, but they had chosen not to continue with it due to lack of time. They had never revisited the idea either, but he wanted to look more into the potential of hot glue (M. Thomassen, personal communication, November 11th 2021).

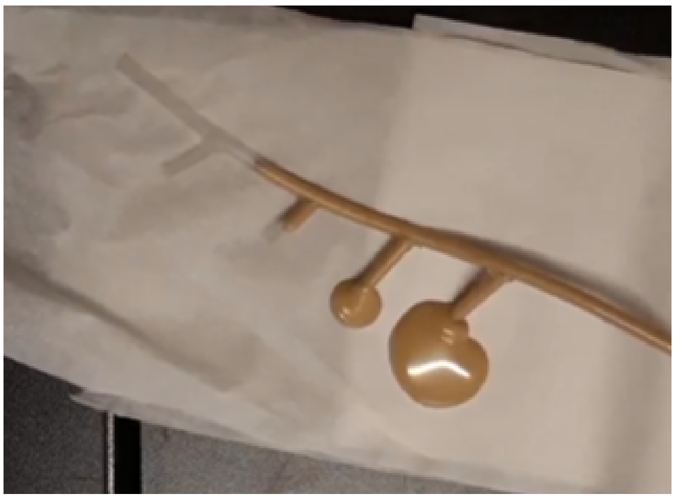
4.2 Wayfaring for Improved Manufacturing Processes

The prototyping with the objective of improving the manufacturing process also resulted in three prototypes. Wayfaring was the chosen method for the development of these prototypes. The prototypes belonging to this section on manufacturing improvement has an "R" in their name for *Renewal*. This section provides info about and images of each prototype, as well as a final subsection discussing the test results.

4.2.1 Prototype R1 - Multi-nozzle Adhesive Dispenser

The pneumatic adhesive dispenser used to attach lamellas to backplates has one nozzle. One strip of adhesive must be applied along the full length of each lamella, which in the case of the Acoustic Wooden Lamella equals eight strips measuring 240 centimeters each. Prototype R1 was an attempt at increasing the number of nozzles so that more MS-polymer adhesive got extruded from the dispenser at once. Increasing the number of lamellas getting adhesive to two at a time would mean a 50 % increase in manufacturing time for this specific process. R1 was a rapid prototype made using rubber tubing, cutting a small piece of tube to create an extension to the nozzle with two exit holes. Further iterations with more holes were also made and tested, but the prototype ended up having only two exit holes, or nozzles, as an increase in nozzles meant more force during extrusion.

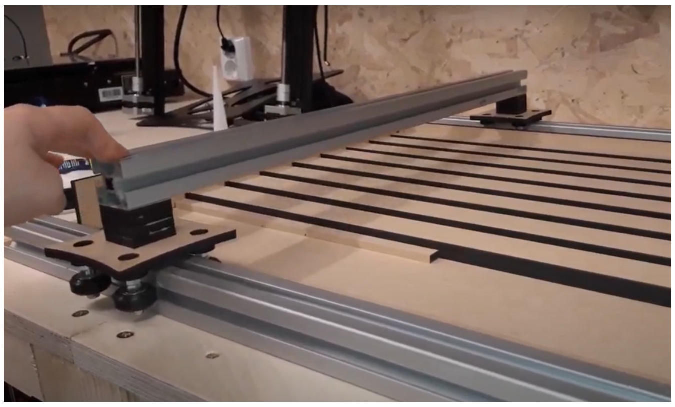
Figure 11: Testing of tube with several holes.



4.2.2 Prototype R2 - Tools on Aluminium Rails

Prototype R2 was a pivot from R1 concerning the problem of renewing the manufacturing process. The prototype centered around altering the jig instead of the equipment, which was the initial thought leading to prototype R1. This prototype consists of two aluminum rails placed parallel to the full length of the jig and one rail mounted on top parallel to the width of the jig. The rail parallel to the width connects to the other two with wheels fitting the aluminum profile and moves with low friction. As a proof-of-concept the prototype is moved manually, and existing tools are held on the top rail. The prototype was made with the motive of implementing motors with CNC software in the long-term, which is elaborated in section 5 Discussion.

Figure 12: Aluminum rails seen from the side of the mock jig,



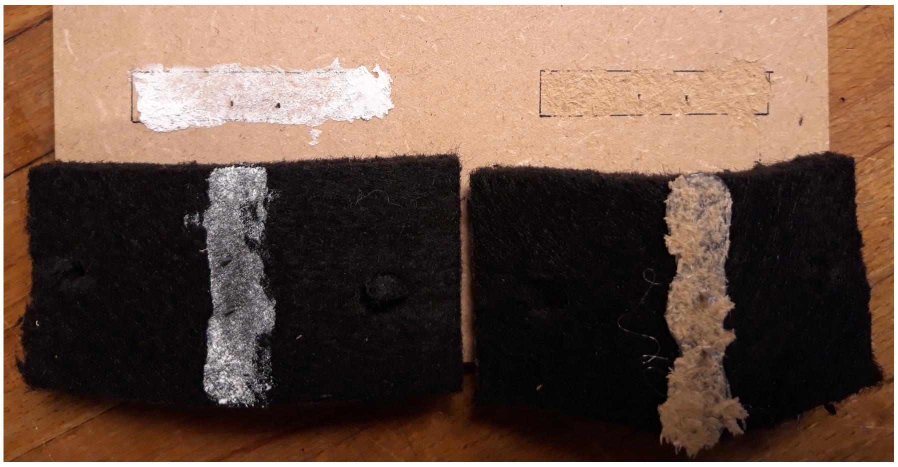
4.2.3 Prototype R3 - A New Adhesive?

Prototype R3 differs from the rest as it proposes a new composition of the Acoustic Wooden Lamella. The prototype is essentially the same product, but with a melting adhesive or "hot glue" replacing the one-component MS-polymer and staple. R3 was ultimately an attempt at reducing manufacturing time by substituting two processes with one, while still preserving the qualities of the finished product. The testing surrounding this prototype concerns the strength of the different types of adhesive, adhesive and staple and just the staple on its own.

4.2.3.1 Testing the Variations of Adhesive and Staple As part or the prototyping for prototype R3, a series of peel and tensile strength tests were performed. The tests were done without the use of testing machines, and should therefore not be regarded as of high scientific value. The results only serve a purpose for moving forward with the prototype or not, should the adhesive be comparable to the original adhesive and staple combination.

No specific equipment was used for the peel tests, other than the material for the prototype itself. The equipment used for the tensile strength tests was a kitchen weight, carabine hook, bag and a bench vise clamp.

Figure 13: MS-polymer adhesive and staple to the left. Heat glue and staple to the right.



A set of peel tests were first carried out to find the type of adhesive best suited to be a replacement for the MS-polymer adhesive and staple combination used on the Acoustic Wooden Lamella. The tested adhesives and/or staple types were Heat glue, Heat glue and staple, MS-polymer, MS-polymer and staple, Superglue and Quick Epoxy. This set of peel tests was done using MDF as the wood component, while the backplate was an original PET-fiber backplate. Adhesives were applied on a 1.0 cm x 5.0 cm area, and all adhesive types were allowed to set for the minimum time required by the manufacturer. The results from the peel test are listed in the table below.

Table 3: Peel tests with adhesives and/or staples and MDF.

Adhesive	Result
Superglue	Absorbs into both MDF and backplate, but does not bind them together. Backplate was loose after test, no peel needed.
Heat glue	Binds backplate well to MDF. Peel test loosens wood fibers from MDF (see Figure 13).
Heat glue plus staple	Obvious increase in strength due to staple. Hard to loosen staple from MDF.
MS-polymer	Barely binds backplate to MDF, has a clay-like texture and does not absorb well into backplate. Easy peel.
MS-polymer plus staple	Also an increase due to staple. MS-polymer adds close to no heft, only hard to loosen staple.
Quick-Epoxy	About the same strength as MS-polymer, does not absorb well into backplate. Relatively easy peel.

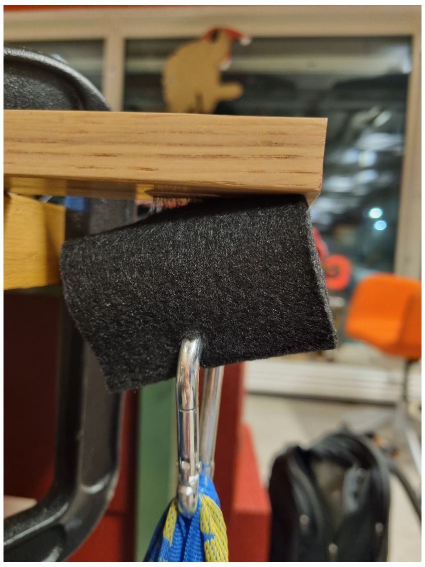
After peel tests, tensile strength tests were carried out for the original Acoustic Wooden Lamella combination of MS-polymer and staple on oak, and heat glue on oak. The PET-fiber backplate was attached to a hook with a bag for the test. Weight was then added to the bag to check when the backpate would come loose from the oak plank.

Table 4: Tensile strength tests with adhesives and/or staples and oak planks.

Adhesive	Result
MS-polymer plus staple	Came loose at 15,9 kg weight on hook.
Heat glue	Came loose at 10,3 kg weight on hook.

Both the MS-polymer plus staple combination and the heat glue could hold over 10.0 kg in weight over quite a small application area. The heat glue was therefore considered as a good enough replacement to use for prototype R3.

Figure 14: Setup for tensile strength test.



4.2.4 Prototyping Outcome

Prototype R1 showed potential as it made it possible to dispense adhesive from two nozzles (or tubes) at once. The main reason for not continuing with this specific prototype was the failure when it came to a further increase in tubes, as dispensing from one container through several tubes demanded much force and distributed unevenly between the tube exits. Prototype R2 allows for several adhesive guns or containers to be placed on the top rail, providing the same output as several nozzles could give. Assembly of prototype R3, the altered Acoustic Wooden Lamella, uses only melting adhesive. It is therefore possible to mount eight low-cost heat glue guns or self-made dispensers with heating elements along the rail.

4.3 Testing Prototype R2 through Assembly of Prototype R3

Testing the renewed process builds on the same test setup that was used for testing the measurement methods in Section 3. The goal of this test is to test the hypothesis that the renewed process prototypes shorten the assembly time, specifically for the adhesive-related sub-processes.

4.3.1 Test Setup

The test setup uses the same mock jig as for the monitoring test in Section 3.3, with the addition of prototype R3 and the use of a heat glue gun. One person will do three iterations of the assembly process where the old adhesive dispenser and staple gun is used, and three iterations where prototype R3 is used and prototype R2 is assembled. The old process is listed in Table 5 and the renewed is listed in Table 6.

Table 5: Testing process for the assembly of the orignal Acoustic Wooden Lamella.

Step	Action
1	Add eight lamellas to the slots in the jig
2	Pick up adhesive dispenser from M2 prototype.
3	Trace dispenser along the full length of all lamellas.
4	Put down adhesive dispenser in M2 prototype.
5	Place the PET-fiber backplate on the lamellas.
6	Pick up staple tool from M2 prototype.
7	Trace staples along all lamellas at two points.
8	Put down staple tool in M2 prototype.
9	Remove lamellas from the slots in the jig.

Table 6: Testing process for the new assembly process of the Acoustic Wooden Lamella with prototypes R2 and R3.

Step	Action
1	Add eight lamellas to the slots in the jig
2	Trace rail with glue gun along the full length of all lamellas.
3	Place PET-fiber backplate.
4	Wait 25 seconds for the adhesive to set.
5	Remove lamellas from the slots in the jig.

4.3.2 Equipment

The test to establish times for the original assembly uses the same mock jig with the M2 and M3 prototypes described in Section 3. For the renewed setup, the mock jig along with the R3 prototype's aluminum rail are added, and a heat glue gun mounted to the top aluminum rail replaces the pneumatic adhesive gun and staple gun. A computer powers the monitoring setup and gathers data from the sensors.

4.3.3 Results

The testing generated six datasets, three for each process. The number of data points in the dataset depends on the writing frequency from the Arduino to the serial port. The code for this test logs approximately four datapoints per second, and each iteration had code running for about 100 seconds. That gives approximately 1600 datapoints in the CSV file generated with the Python script.

The results from testing show that there is a small improvement in the manufacturing time for the Acoustic Wooden Lamella in the new assembly process. The original assembly is 86 seconds long averaging over three iterations, and the new assembly is 78 seconds long also averaging over three iterations.

Figures 16 and 17 are column charts showing the average time spent on each subprocess in seconds. Time in seconds is given on the x-axis and along the y-axis the name of the sub-processes are listed. The most notable difference between them is that on average, waiting for the set time of the melting adhesive in the new process saves seven seconds compared to using the staple gun.

Figure 15: Graphical overview of time spent on different parts of the original assembly.

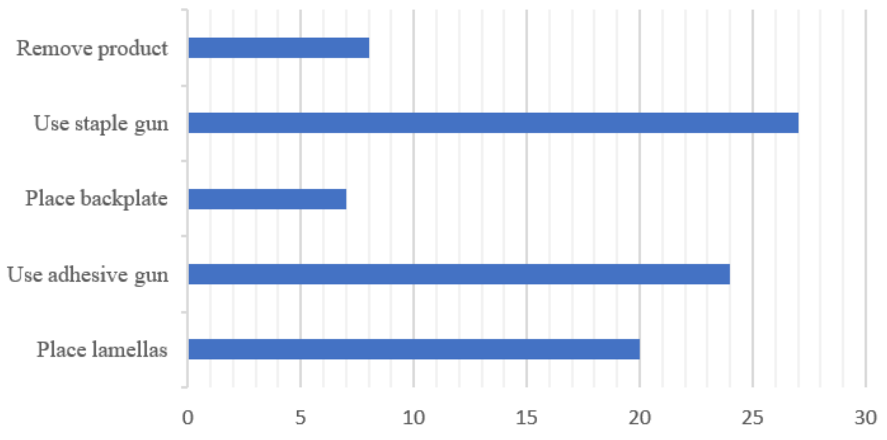
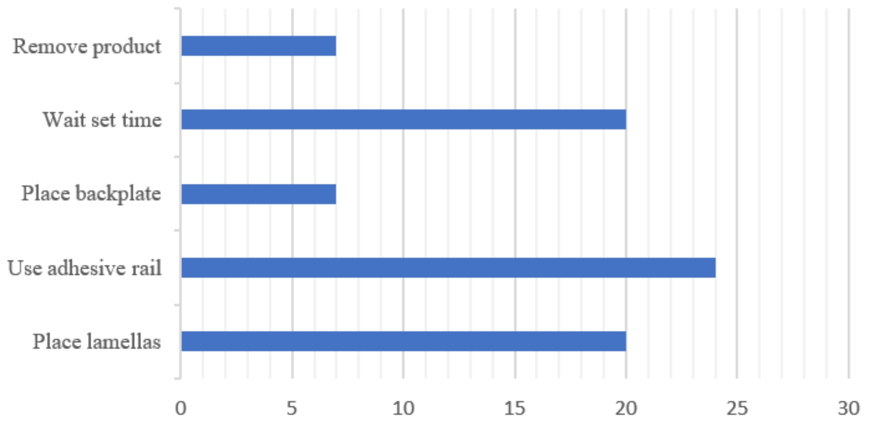


Figure 16: Graphical overview of time spent on different parts of the improved assembly.



4.4 Discussion of Test Results

The total time spent per Acoustic Wooden Lamella assembly differs by just a few seconds between the old and the new process. Using data from just three iterations per type of process gives a small basis for comparison, and ideally the data set would have been much larger. Setting up monitoring equipment at Otretek AS should be prioritized going forward. Analyzing the data points shows that the three iterations have quite similar assembly times for both the whole process and sub-processes. The average is therefore deemed usable for the purpose of this test.

The same error as in the monitoring testing occured with the M2 prototype for docked tool detection during this test. The wiring was checked to assure proper power supply to the system, and there was still something interfering with the signal from the HC-SR04 Ultrasonic Sensor. In a wood product manufacturing hall, it is common to have sawdust and other larger particles in the air and on surfaces. This might also be the case for the area where the prototypes were stored before testing. Faulty sensors or a dusty environment can therefore be the reasons why the M2 prototype using HC-SR04 sensors has failed. This type of sensor will therefore not be used for further work, and prototype M2 will go through a new set of iterations.

It is worth noting that had the assembly times been equal, it still would have been a step in the right direction concerning the plans for further work on the R3 prototype. The three iterations in the test had no interruptions or faulty assembly equipment interfering with it, which might not be the case in real-life manufacturing. As mentioned in sections 3.1.3 and 4.1.1 *Findings from Company Visits*, equipment had to be replaced regularly and the skill level of the employees had large variations. Further work with the R2 and R3 prototypes points towards making a system that to a larger extent standardizes the assembly process.

5 Discussion

This section presents a discussion of the product development project and a suggestion for further work.

5.1 On Product Development for Otretek AS

The fuzzy front-end in an innovation process is characterized by a high degree of ambiguity and many unknowns. Trying to define a problem to solve for Otretek AS was clearly affected by these factors. Drawing the empathy component from Design Thinking into the needsfinding process and focusing on asking generative questions helped in identifying aspects of their manufacturing that could be improved.

The prototyping activities have largely resulted in low-fidelty physical prototypes. Creating physical prototypes has been of importance as prototype testing and feedback has been in focus throughout the process. Tangible prototypes, even low-fidelty ones, are more suited for physical testing (Ulrich, 2016).

The testing ended up being performed using a mock jig at TrollLABS and not using the real jig at Otretek AS, which resulted in a lack of user input. In that sense new tests with users from Otretek AS should be performed going forward, maintaining the user-centric aspect of Design Thinking (Jensen et al., 2016). The tests themselves should also be reworked. Testing over a longer period of time and by using the prototypes in actual manufacturing will yield results better suited for analysis and decision-making. Including a user survey to get structured feedback might also be relevant.

The cost of implementing known Industry 4.0 solutions and a lack of knowledge on the subject are obstacles keeping small- and medium-sized companies from investing in technology. Prototyping Industry 4.0 solutions using low-cost sensors and the equipment available at the makerspace in Oppdal has therefore been in focus during the project. The makerspace is intended as a resource for the local population, where both individuals and companies can learn how to prototype and be innovative. Workshops where employees can test simple low-cost systems using tutorials and guidance can be a stepping stone for developing their own Industry 4.0 solutions.

5.2 Further Work

As this project is considered preliminary work for the Master's thesis on the same subject, the prototyping will continue in the spring of 2022.

The first course of action is to change the M2 prototype to use a sensor that is more reliable. The HS-SR04 Ultrasonic Distance Sensor has proved that it is very fragile to particle contamination and possibly bad wiring, and will not work in the prototype's current format. Exchanging it for a photocell or maybe an RFID reader might be the way to go if it is still relevant to monitor the original assembly process.

The largest task going forward will be motorizing the aluminum rails and integrating CNC software to control them. Mounting a solution for melting glue guns on the top rail is also on the agenda. Ideally this solution can be controlled and monitored remotely. The idea as it stands is to have an employee feed material in a jig which is then detected by sensors. This will trigger the top rail with melting glue to dispense glue on the lamellas. The solution space is large for such a system, and it will require testing at Otretek AS during the spring.

One area which has not received any focus but was shortly mentioned by the Production Manager during one visit, is the company's focus on HSE. Lifting lamellas from the floor up to the workbench, and then carrying the assembled panel off the workbench can take a toll on the worker. They had not yet implemented any actions to improve the HSE-aspect of the wooden lamella assembly as of the last visit in November. Looking into this could be interesting in terms of *dark horse prototyping* (Steinert and Leifer, 2012).

6 Conclusion

This project thesis has explored of the possibilities of implementing low-tech Industry 4.0 solutions in a small- and medium-sized enterprise.

Production of Acoustic Wooden Lamellas at Otretek AS, a wood product manufacturer from Oppdal, was explored. Needsfinding was performed for employees involved in the assembly of the wooden lamella, as well as through conversations with the Production Manager and Assistant Manager.

Prototyping and testing has resulted in three prototypes which will be worked on further. Prototype M3, Lamella placement using photocells, is the prototype which at the moment best encapsulates Industry 4.0. Prototypes R2, Tools on aluminum rails, and R3, A new adhesive, are preliminary prototypes for further development. The prototypes have been developed at makerspaces using resources which are also available for the employees at Otretek AS. With some guidance it should be possible for the employees to discover and integrate simple Industry 4.0 solutions themselves.

Bibliography

Al-Maeeni, S. S. H., Kuhnhen, C., Engel, B. & Schiller, M. (2020). Smart retrofitting of machine tools in the context of industry 4.0. *Procedia CIRP*, *88*, 369–374.

Baba, Deros, M., Mohd Yusof, S., Azhari & Salleh, M. (2006). A benchmarking implementation framework for automotive manufacturing smes. *Benchmarking: An International Journal*, *13*(4), 396–430.

Dalenogare, L. S., Benitez, G. B., Ayala, N. F. & Frank, A. G. (2018). The expected contribution of industry 4.0 technologies for industrial performance. *International Journal of Production Economics*, *204*, 383–394.

Doorley, S., Holcomb, S., Klebahn, P., Segovia, K. & Utley, J. *Design thinking bootleg.* Distributed to students at d.school. 2018, July.

Elverum, C. W., Welo, T. & Steinert, M. (2014). The fuzzy front end: Concept development in the automotive industry. *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, *46407*, V007T07A037.

Eris, Ö. et al. (2003). Asking generative design questions: A fundamental cognitive mechanism in design thinking. *DS 31: Proceedings of ICED 03, the 14th International Conference on Engineering Design, Stockholm*, 587–588.

Frank, A. G., Dalenogare, L. S. & Ayala, N. F. (2019). Industry 4.0 technologies: Implementation patterns in manufacturing companies. *International Journal of Production Economics*, *210*, 15–26.

Houde, S. & Hill, C. (1997). What do prototypes prototype? *Handbook of human-computer interaction* (pp. 367–381). Elsevier.

Jensen, M. B., Elverum, C. W. & Steinert, M. (2017). Eliciting unknown unknowns with prototypes: Introducing prototrials and prototrial-driven cultures. *Design Studies*, *49*, 1–31.

Jensen, M. B., Lozano, F. & Steinert, M. (2016). The origins of design thinking and the relevance in software innovations. *International Conference on Product-Focused Software Process Improvement*, 675–678.

Koen, P. A., Ajamian, G. M., Boyce, S., Clamen, A., Fisher, E., Fountoulakis, S., Johnson, A., Puri, P. & Seibert, R. (2002). Fuzzy front end: Effective methods, tools, and techniques. *The PDMA toolbook*, *1*, 5–35.

Lasi, H., Fettke, P., Kemper, H.-G., Feld, T. & Hoffmann, M. (2014). Industry 4.0. *Business & information systems engineering*, *6*(4), 239–242.

Næringsliv, D. (2022). *Gaselle*. https://www.dn.no/gaselle

NHO. (2018). *Fakta om små og mellomstore bedrifter*. https://www.nho.no/tema/sma-og-mellomstore-bedrifter/artikler/sma-og-mellomstore-bedrifter-smb/

Oliviero, C., Pastell, M., Heinonen, M., Heikkonen, J., Valros, A., Ahokas, J., Vainio, O. & Peltoniemi, O. A. (2008). Using movement sensors to detect the onset of farrowing. *Biosystems Engineering*, *100*(2), 281–285.

Pech, M. & Vrchota, J. (2020). Classification of small-and medium-sized enterprises based on the level of industry 4.0 implementation. *Applied Sciences*, *10*(15), 5150.

Proff.no. (2022). *Otretek AS*. https://www.proff.no/selskap/otretek-as/oppdal/trevarer/IGDZLG101RN/

Reinhart, C. F. (2004). Lightswitch-2002: A model for manual and automated control of electric lighting and blinds. *Solar energy*, *77*(1), 15–28.

Schrage, M. (1996). Cultures of prototyping. *Bringing design to software*, *4*(1), 1–11.

Sevinc, A., Gür, Ş. & Eren, T. (2018). Analysis of the difficulties of smes in industry 4.0 applications by analytical hierarchy process and analytical network process. *Processes*, *6*(12), 264.

Silseth Naas, K. (2017). Otretek er årets bedrift. *Opp*. Retrieved 6th January 2022, from https://www.opp.no/nyheter/i/23zVvx/otretek-er-arets-bedrift

Steinert, M. & Leifer, L. J. (2012). 'finding one's way': Re-discovering a hunter-gatherer model based on wayfaring. *International Journal of Engineering Education*, *28*(2), 251.

Tøsse, A. (2013). Otretek gasellebedrift. *Opp*. Retrieved 6th January 2022, from https://www.opp.no/nyheter/i/g0lJoq/otretek-gasellebedrift

Ulrich, S. D., Karl T. Eppinger. (2016). *Product design and development*. McGraw-Hill Education.

Appendix B: Arduino code measuring movement

```
/* ***** MINIMAL SETUP *****
This is the code that will run on your microcontroller!

Some fields will have to be modified for it to work, these fields
are the following:

[sensorID] The ID of your sensor, must be unique
[sensorName] The name of your sensor that you will see in Insight
[stationName] The name of the station where your sensor is placed
[readingUnit] Unit for the readings, for example "min", "Hz" or "hour"
[readingType] Type for readings, use "double" if unfamiliar with types
*/

/* FILL IN YOUR PREFERED FIELD NAMES BETWEEN THESE SYMBOLS "" */
String sensorID = "ExampleSensor";
String sensorName = "Example Sensor";
String stationName = "Example Station";
String readingUnit = "sec";
String readingType = "double";


/* ***** FURTHER SETUP AND DOCUMENTATION *****

The code requires a Secret.h header file with the following definitions:

#define SECRET_SSID "YOUR WIFI SSID"
#define SECRET_PASS "YOUR WIFI PASSWORD"
#define FIREBASE_HOST "FIREBASE REAL TIME DATABASE URL ENDING IN
.firebasedatabase.app"
#define FIREBASE_AUTH "FIREBASE SECRET"


**********************************************
Documentation for the LSM6DS33 6-DoF IMU accelerometer + Gyro can be found at
https://adafruit.github.io/Adafruit_LSM6DS/html/index.html.

Based on example code from the official Adafruit GitHub, found at
https://github.com/adafruit/Adafruit_LSM6DS/blob/master/examples/
adafruit_lsm6ds33_test/adafruit_lsm6ds33_test.ino

The frequency output is generated based on the Arduino FFT library. test
https://www.arduino.cc/reference/en/libraries/arduinofft/
```

```

*/

#include <math.h>
#include <stdlib.h>
#include <Arduino_LSM6DS3.h>
#include <Firebase_Arduino_WiFiNINA.h>
#include <WiFiNINA.h>
#include "Secret.h"

char ssid[] = SECRET_SSID;
char pass[] = SECRET_PASS;

int wifi_status = WL_IDLE_STATUS;

unsigned long motion_start_time;
unsigned long current_time;
unsigned long duration;

float accel_x, accel_y, accel_z;
float last_loop_accel_z;

bool motion_active;
bool last_loop_motion_active;

//Define Firebase data object
FirebaseData fbdo;

void setup() {
  Serial.begin(9600);

  // Don't continue if IMU is not ready
  if (!IMU.begin()) {
    Serial.println("Failed to initialize IMU!");
    while (1);
  }

  // Set accel_z to 0 in setup
  accel_z = 0;
  last_loop_accel_z = 0;

  // Set motion active to false in setup
  motion_active = false;
  last_loop_motion_active = false;

  // Don't continue if WiFi is not ready
  WiFi.begin(SECRET_SSID, SECRET_PASS);
  if (WiFi.status() == WL_NO_MODULE) {
    Serial.println("Communication with WiFi module failed!");
    // don't continue
    while (true);
  }

  // Connect to WiFi network
  while (wifi_status != WL_CONNECTED) {
    Serial.print("Attempting connection to SSID: ");
    Serial.println(SECRET_SSID);
    wifi_status = WiFi.begin(SECRET_SSID, SECRET_PASS);

    delay(1000);
  }

  // Establish Firebase connection
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH, SECRET_SSID, SECRET_PASS);
  Firebase.reconnectWiFi(true);

  // Push initial data to Firebase
  if (Firebase.setString(fbdo, "/" + sensorID + "/name", sensorName))
  {
    Serial.println("Path: " + fbdo.dataPath());
    Serial.println("Type: " + fbdo.dataType());
  }
  else
  {
    Serial.println("Error, " + fbdo.errorReason());
  }

  if (Firebase.setString(fbdo, "/" + sensorID + "/stationName", stationName))
  {
    Serial.println("Path: " + fbdo.dataPath());
    Serial.println("Type: " + fbdo.dataType());
  }
  else
  {
    Serial.println("Error, " + fbdo.errorReason());
  }

  if (Firebase.setString(fbdo, "/" + sensorID + "/readingunit", readingUnit))
  {
    Serial.println("Path: " + fbdo.dataPath());
    Serial.println("Type: " + fbdo.dataType());
  }
  else
  {
    Serial.println("Error, " + fbdo.errorReason());
  }

  if (Firebase.setString(fbdo, "/" + sensorID + "/readingtype", readingType))
  {
    Serial.println("Path: " + fbdo.dataPath());
    Serial.println("Type: " + fbdo.dataType());
  }
  else
  {
    Serial.println("Error, " + fbdo.errorReason());
  }


  // Push initial data to Firebase to avoid plot nullpointer
    if (Firebase.setDouble(fbdo, "/" + sensorID + "/readings/" +
    String(WiFi.getTime()), 0))
    {
      Serial.println("Successful write!");
      Serial.println("Path: " + fbdo.dataPath());
      Serial.println("Type: " + fbdo.dataType());
    }
    else
    {
      Serial.println("Write error, " + fbdo.errorReason());
    }
}

void loop() {

  // Read acceleration data from IMU
  current_time = millis();

  // Read acceleration data from IMU
  if (IMU.accelerationAvailable()) {
    last_loop_accel_z = accel_z;
    IMU.readAcceleration(accel_x, accel_y, accel_z);

    Serial.println(last_loop_accel_z);
    Serial.println(accel_z);
    Serial.println();
  }

  last_loop_motion_active = motion_active;
  if ((accel_z < 0.95 || accel_z > 1.05) && (accel_z <
  (last_loop_accel_z - 0.05) ||
  accel_z > (last_loop_accel_z + 0.05)))  {
    motion_active = true;
    current_time = millis();
    duration = current_time - motion_start_time;
    Serial.println("first");
    Serial.println();
  } else if ((duration > 1000) && (accel_z < (last_loop_accel_z + 0.02) ||
  accel_z > (last_loop_accel_z - 0.02)))  {
    motion_active = false;
    motion_start_time = millis();
    current_time = millis();
    Serial.println("second");
    Serial.println();
  }

  if (motion_active == false && last_loop_motion_active) {
    Serial.println(duration/1000.0);

    // Push data to Firebase
    if (Firebase.setDouble(fbdo, "/" + sensorID + "/readings/" +
    String(WiFi.getTime()), duration/1000.0))
    {
      Serial.println("Successful write!");
      Serial.println("Path: " + fbdo.dataPath());
      Serial.println("Type: " + fbdo.dataType());
    }
    else
    {
      Serial.println("Write error, " + fbdo.errorReason());
    }
  }

  delay(1000);
}
```

Appendix C: Arduino code measuring vibration

```
/* ***** MINIMAL SETUP *****
This is the code that will run on your microcontroller!

Some fields will have to be modified for it to work, these fields
are the following:

[sensorID] The ID of your sensor, must be unique
[sensorName] The name of your sensor that you will see in Insight
[stationName] The name of the station where your sensor is placed
[readingUnit] Unit for the readings, for example "min", "Hz" or "hour"
[readingType] Type for readings, use "double" if unfamiliar with types
*/

/* FILL IN YOUR PREFERED FIELD NAMES BETWEEN THESE SYMBOLS "" */
String sensorID = "ExampleSensor";
String sensorName = "Example Sensor";
String stationName = "Example Station";
String readingUnit = "sec";
String readingType = "double";


/* ***** FURTHER SETUP AND DOCUMENTATION *****

The code requires a Secret.h header file with the following definitions:

#define SECRET_SSID "YOUR WIFI SSID"
#define SECRET_PASS "YOUR WIFI PASSWORD"
#define FIREBASE_HOST "FIREBASE REAL TIME DATABASE URL ENDING IN
.firebasedatabase.app"
#define FIREBASE_AUTH "FIREBASE SECRET"

**********************************************
Documentation for the LSM6DS33 6-DoF IMU accelerometer + Gyro can be found at
https://adafruit.github.io/Adafruit_LSM6DS/html/index.html.

Based on example code from the official Adafruit GitHub, found at
https://github.com/adafruit/Adafruit_LSM6DS/blob/master/examples/
adafruit_lsm6ds33_test/adafruit_lsm6ds33_test.ino

The frequency output is generated based on the Arduino FFT library, found at
https://www.arduino.cc/reference/en/libraries/arduinofft/
*/
```

```

#include <math.h>
#include <stdlib.h>
#include <arduinoFFT.h>
#include <Arduino_LSM6DS3.h>
#include <Firebase_Arduino_WiFiNINA.h>
#include <WiFiNINA.h>
#include "Secret.h"

char ssid[] = SECRET_SSID;
char pass[] = SECRET_PASS;

int wifi_status = WL_IDLE_STATUS;

// This value MUST ALWAYS be a power of 2
const uint16_t samples = 64;

// Set to 104 as the accelerometer of the LSM6DS3 is capped at 104 Hz
const double sampling_rate = 104;

unsigned int sampling_period_us;
unsigned long microseconds;
unsigned long current_time;

/*
These are the input and output vectors
Input vectors receive computed results from FFT
*/
double vReal[samples];
double vImag[samples];

float accel_x, accel_y, accel_z;
float magnitude;

arduinoFFT FFT = arduinoFFT();

//Define Firebase data object
FirebaseData fbdo;

void setup() {
  Serial.begin(9600);

  // Set sampling period
  sampling_period_us = round(1000000*(1.0/sampling_rate));

  // Don't continue if IMU is not ready
  if (!IMU.begin()) {
    Serial.println("Failed to initialize IMU!");
    while (1);
  }

  // Don't continue if WiFi is not ready
  WiFi.begin(SECRET_SSID, SECRET_PASS);
  if (WiFi.status() == WL_NO_MODULE) {
    Serial.println("Communication with WiFi module failed!");
    // don't continue
    while (true);
  }

  // Connect to WiFi network
  while (wifi_status != WL_CONNECTED) {
    Serial.print("Attempting connection to SSID: ");
    Serial.println(SECRET_SSID);
    wifi_status = WiFi.begin(SECRET_SSID, SECRET_PASS);

    delay(1000);
  }

  // Establish Friebase connection
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH, SECRET_SSID, SECRET_PASS);
  Firebase.reconnectWiFi(true);

  // Push initial data to Firebase
  if (Firebase.setString(fbdo, "/" + sensorID + "/name", sensorName))
  {
    Serial.println("Path: " + fbdo.dataPath());
    Serial.println("Type: " + fbdo.dataType());
  }
  else
  {
    Serial.println("Error, " + fbdo.errorReason());
  }

  if (Firebase.setString(fbdo, "/" + sensorID + "/stationName", stationName))
  {
    Serial.println("Path: " + fbdo.dataPath());
    Serial.println("Type: " + fbdo.dataType());
  }
  else
  {
    Serial.println("Error, " + fbdo.errorReason());
  }

  if (Firebase.setString(fbdo, "/" + sensorID + "/readingunit", readingUnit))
  {
    Serial.println("Path: " + fbdo.dataPath());
    Serial.println("Type: " + fbdo.dataType());
  }
  else
  {
    Serial.println("Error, " + fbdo.errorReason());
  }

  if (Firebase.setString(fbdo, "/" + sensorID + "/readingtype", readingType))
  {
    Serial.println("Path: " + fbdo.dataPath());
    Serial.println("Type: " + fbdo.dataType());
  }
  else
  {
    Serial.println("Error, " + fbdo.errorReason());
  }

  // Push initial data to Firebase to avoid plot nullpointer
  if (Firebase.setDouble(fbdo, "/" + sensorID + "/readings/" +
  String(WiFi.getTime()), 0))
  {
    Serial.println("Successful write!");
    Serial.println("Path: " + fbdo.dataPath());
    Serial.println("Type: " + fbdo.dataType());
  }
  else
  {
    Serial.println("Write error, " + fbdo.errorReason());
  }
}

void loop() {

  // FFT
  /*SAMPLING*/
  microseconds = micros();
  for(int i=0; i < samples; i++)
  {
    if (IMU.accelerationAvailable()) {
      IMU.readAcceleration(accel_x, accel_y, accel_z);
      //Serial.println(IMU.accelerationSampleRate());
      magnitude = sqrt(pow(accel_x, 2) + pow(accel_y, 2) + pow(accel_z, 2));
    }

    current_time = micros();
    vReal[i] = magnitude;
    vImag[i] = 0;
    while(micros() < (current_time + sampling_period_us)) {
      // Cool down period
    }
    //Serial.println(vReal[i]);
  }

  // Fast Fourier Transform computations
  FFT.DCRemoval();
  FFT.Windowing(vReal, samples, FFT_WIN_TYP_HAMMING, FFT_FORWARD);
  FFT.Compute(vReal, vImag, samples, FFT_FORWARD);
  FFT.ComplexToMagnitude(vReal, vImag, samples);

  double x = FFT.MajorPeak(vReal, samples, sampling_rate);
  Serial.print("Peak frequency: ");
  Serial.print(x, 6); //Print out what frequency is the most dominant
  Serial.println(" Hz");

  // Push data to Firebase
  if (Firebase.setDouble(fbdo, "/" + sensorID + "/readings/" +
  String(WiFi.getTime()), x))
  {
    Serial.println("Successful write!");
    Serial.println("Path: " + fbdo.dataPath());
    Serial.println("Type: " + fbdo.dataType());
    }
    else
    {
      Serial.println("Write error, " + fbdo.errorReason());
  }

  delay(2000);
}
```

Appendix D: Pre-test survey and SUS

SYSTEM USABILITY SCALE

	Strongly disagree				Strongly agree
I think that I would like to use this system frequently	1	2	3	4	5
I found the system unnecessarily complex	1	2	3	4	5
I thought the system was easy to use	1	2	3	4	5
I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
I found that the various functions in this system were well integrated	1	2	3	4	5
I thought that there was too much inconsistency in this system	1	2	3	4	5
I would imagine that that most people would learn to use this system very quickly	1	2	3	4	5
I found the system very awkward to use	1	2	3	4	5
I felt very confident using the system	1	2	3	4	5
I needed to learn a lot of things before I could get going with this system	1	2	3	4	5

PRE-TEST QUESTIONS

	Strongly disagree				Strongly agree
I use old or manual tools or machinery regularly at work	1	2	3	4	5
I want to know data about the use of tools or machinery at my workplace	1	2	3	4	5
I know what a sensor is	1	2	3	4	5
I know what a microcontroller is	1	2	3	4	5
I have programmed a microcontroller	1	2	3	4	5
I am interested in learning to use a sensor or program a microcontroller	1	2	3	4	5

Appendix E: ”Insight” Source Code

Target Framework

.NET Core 3.1

Packages

CommonServiceLocator, Version 2.0.6

CommunityToolkit.Mvvm, Version 7.1.2

FirebaseAuthentication.net, Version 4.0.0-alpha.2

FirebaseAuthentication.WPF, Version 4.0.0-alpha.2

FirebaseDatabase.net, Version 4.0.7

MvvmLightLibs, Version 5.4.1.1

OxyPlot.Wpf, Version 2.1.0

ToastNotifications, Version 2.5.1

ToastNotifications.Messages, Version 2.5.1

Folders

Insight

– Assets

– Models

– Services

– – Commands

– – Converters

– ViewModels

– Views

Insight/Models/ErrorMessage.cs

```
namespace Insight.Models
{
    public class ErrorMessage
    {
        public string Message { get; private set; }

        public ErrorMessage(string message)
        {
            Message = message;
        }
    }
}
```

Insight/Models/Reading.cs

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Text;

namespace Insight.Models
{
    public class Reading
    {
        public DateTime TimeStamp { get; set; }

        public double Value { get; set; }
    }
}
```

Insight/Models/Sensor.cs

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Text;
using Insight.Services.Commands;
using Newtonsoft.Json;
using OxyPlot;

namespace Insight.Models
{
    public class Sensor
    {
        [JsonProperty("key")]
        public string ID { get; set; }

        [JsonProperty("name")]
        public string Name { get; set; }

        [JsonProperty("stationname")]
        public string StationName { get; set; }

        [JsonProperty("readingunit")]
        public string ReadingUnit { get; set; }

        [JsonProperty("readingtype")]
        public string ReadingType { get; set; }

        [JsonProperty("enablereadingvaluelimits")]
        public string EnableReadingValueLimits { get; set; }

        [JsonProperty("maxreadingvalue")]
        public int ReadingValueUpperLimit { get; set; }

        [JsonProperty("minreadingvalue")]
        public int ReadingValueLowerLimit { get; set; }

        public ObservableCollection<Reading> ReadingList { get; set; }

        public NavigateSensorCommand NavigateSensorCommand { get; set; }

        public NavigateSensorSettingsCommand
            NavigateSensorSettingsCommand { get; set; }

        public int SumReadings { get; set; }

        public double SumReadingValues { get; set; }

        public double AverageReadingValue { get; set; }
    }
}
```

Insight/Models/SensorUpdateMessage.cs

```
using CommunityToolkit.Mvvm.Messaging;
using System;
using System.Collections.Generic;
using System.Text;

namespace Insight.Models
{
    public class SensorUpdateMessage
    {
        public string Message { get; private set; }

        public SensorUpdateMessage(string message)
        {
            Message = message;
        }
    }
}
```

Insight/Services/Commands/CommandBase.cs

```
using System;
using System.Windows.Input;

namespace Insight.Services.Commands
{
    public abstract class CommandBase : ICommand
    {
        public event EventHandler CanExecuteChanged;

        public virtual bool CanExecute(object parameter) => true;

        public abstract void Execute(object parameter);

        protected void OnExecuteChanged()
        {
            CanExecuteChanged?.Invoke(this, new EventArgs());
        }
    }
}
```

Insight/Services/Commands/NavigateHomeCommand.cs

```csharp
using Insight.ViewModels;
using System;
using System.Collections.Generic;
using System.Text;

namespace Insight.Services.Commands
{
    public class NavigateHomeCommand : CommandBase
    {
        private readonly NavigationStore navigationStore;
        private readonly FirebaseAuthStore firebaseAuthStore;

        public NavigateHomeCommand(FirebaseAuthStore firebaseAuthStore,
        NavigationStore navigationStore)
        {
            this.firebaseAuthStore = firebaseAuthStore;
            this.navigationStore = navigationStore;
        }

        public override void Execute(object parameter)
        {
            navigationStore.CurrentViewModel = new HomeViewModel(
            firebaseAuthStore, navigationStore);
        }
    }
}
```

Insight/Services/Commands/NavigateInfoCommand.cs

```csharp
using Insight.ViewModels;

namespace Insight.Services.Commands
{
    public class NavigateInfoCommand : CommandBase
    {
        private readonly NavigationStore navigationStore;

        public NavigateInfoCommand(NavigationStore navigationStore)
        {
            this.navigationStore = navigationStore;
        }

        public override void Execute(object parameter)
        {
            navigationStore.CurrentViewModel = new InfoViewModel(navigationStore);
        }
    }
}
```

Insight/Services/Commands/NavigateLoginCommand.cs

```
using Insight.ViewModels;
using System;
using System.Collections.Generic;
using System.Text;

namespace Insight.Services.Commands
{
    public class NavigateLoginCommand : CommandBase
    {
        private readonly NavigationStore navigationStore;

        public NavigateLoginCommand(NavigationStore navigationStore)
        {
            this.navigationStore = navigationStore;
        }

        public override void Execute(object parameter)
        {
            navigationStore.CurrentViewModel = new LoginViewModel(navigationStore);
        }
    }
}
```

Insight/Services/Commands/NavigateSensorCommand.cs

```csharp
using Insight.Models;
using Insight.ViewModels;
using System;
using System.Collections.Generic;
using System.Text;

namespace Insight.Services.Commands
{
    public class NavigateSensorCommand : CommandBase
    {
        private readonly NavigationStore navigationStore;
        private readonly FirebaseAuthStore firebaseAuthStore;
        private Sensor sensor;

        public NavigateSensorCommand(NavigationStore navigationStore,
        FirebaseAuthStore firebaseAuthStore, Sensor sensor)
        {
            this.navigationStore = navigationStore;
            this.firebaseAuthStore = firebaseAuthStore;
            this.sensor = sensor;
        }

        public override void Execute(object parameter)
        {
            navigationStore.CurrentViewModel = new
            SensorViewModel(firebaseAuthStore, sensor);
        }
    }
}
```

Insight/Services/Commands/ NavigateSensorSettingsCommand.cs

```
using Insight.Models;
using Insight.ViewModels;

namespace Insight.Services.Commands
{
    public class NavigateSensorSettingsCommand : CommandBase
    {
        private readonly NavigationStore navigationStore;
        private readonly FirebaseAuthStore firebaseAuthStore;
        private Sensor sensor;

        public NavigateSensorSettingsCommand(NavigationStore navigationStore,
        FirebaseAuthStore firebaseAuthStore, Sensor sensor)
        {
            this.navigationStore = navigationStore;
            this.firebaseAuthStore = firebaseAuthStore;
            this.sensor = sensor;
        }

        public override void Execute(object parameter)
        {
            navigationStore.CurrentViewModel = new
            SensorSettingsViewModel(firebaseAuthStore, sensor);
        }
    }
}
```

Insight/Services/Commands/ OpenHttpLinkCommand.cs

```csharp
using GalaSoft.MvvmLight.Messaging;
using Insight.Models;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Text;

namespace Insight.Services.Commands
{
    public class OpenHttpLinkCommand : CommandBase
    {
        public override void Execute(object parameter)
        {
            try
            {
                var psi = new ProcessStartInfo
                {
                    FileName = "https://github.com/hannaaks/Insight_Sensors",
                    UseShellExecute = true
                };
                Process.Start(psi);
            }
            catch (Exception)
            {
                Messenger.Default.Send(new ErrorMessage("Could not open link!"));
            }
        }
    }
}
```

Insight/Services/Commands/SaveSettingsCommand.cs

```
using System.ComponentModel;

namespace Insight.Services.Commands
{
    public class SaveSettingsCommand : CommandBase, INotifyPropertyChanged
    {
        private readonly FirebaseDataService firebaseDataService;
        private readonly FirebaseAuthStore firebaseAuthStore;
        public event PropertyChangedEventHandler PropertyChanged;

        private string id_0;
        public string Id_0
        {
            get => id_0;
            set
            {
                id_0 = value;
                OnPropertyChanged(nameof(Id_0));
            }
        }

        private string item_0;
        public string Item_0
        {
            get => item_0;
            set
            {
                item_0 = value;
                OnPropertyChanged(nameof(Item_0));
            }
        }

        private string id_1;
        public string Id_1
        {
            get => id_1;
            set
            {
                id_1 = value;
                OnPropertyChanged(nameof(Id_1));
            }
        }

        private string item_1;
        public string Item_1
        {
            get => item_1;
            set
            {
                item_1 = value;
                OnPropertyChanged(nameof(Item_1));
            }
        }

        private string id_2;
        public string Id_2
        {
            get => id_2;
            set
            {
                id_2 = value;
                OnPropertyChanged(nameof(Id_2));
            }
        }

        private string item_2;
        public string Item_2
        {
            get => item_2;
            set
            {
                item_2 = value;
                OnPropertyChanged(nameof(Item_2));
            }
        }

        public SaveSettingsCommand(FirebaseDataService firebaseDataService,
        FirebaseAuthStore firebaseAuthStore)
        {
            this.firebaseDataService = firebaseDataService;
            this.firebaseAuthStore = firebaseAuthStore;
        }

        public override async void Execute(object parameter)
        {
            await firebaseDataService.UpdateItemAsync(firebaseAuthStore,
            id_0, item_0);
            await firebaseDataService.UpdateItemAsync(firebaseAuthStore,
            id_1, item_1);
            await firebaseDataService.UpdateItemAsync(firebaseAuthStore,
            id_2, item_2);
        }

        public void OnPropertyChanged(string propertyName = null)
        {
            PropertyChanged?.Invoke(this, new
            PropertyChangedEventArgs(propertyName));
        }
    }
}
```

Insight/Services/Commands/SignOutCommand.cs

```
using Firebase.Auth;

namespace Insight.Services.Commands
{
    public class SignOutCommand : CommandBase
    {
        private FirebaseAuthClient client;

        public SignOutCommand(FirebaseAuthClient client)
        {
            this.client = client;
        }

        public override void Execute(object parameter)
        {
            client.SignOutAsync();
        }
    }
}
```

Insight/Services/Converters/LastItemConverter.cs

```
// Based on https://stackoverflow.com/questions/
38076783/binding-to-last-array-element

using Insight.Models;
using System.Linq;
using System.Windows.Data;
using System.Collections.ObjectModel;

namespace Insight.Services.Converters
{
    class LastItemConverter : IValueConverter
    {
        public object Convert(object value, System.Type targetType,
        object parameter, System.Globalization.CultureInfo culture)
        {
            ObservableCollection<Reading> readings =
            value as ObservableCollection<Reading>;
            if (readings != null)
            {
                return readings.LastOrDefault().Value;
            }
            else return Binding.DoNothing;
        }

        public object ConvertBack(object value, System.Type targetType,
        object parameter, System.Globalization.CultureInfo culture)
        {
            throw new System.NotImplementedException();
        }
    }
}
```

Insight/Services/Converters/ LastItemTimeStampConverter.cs

```
// Based on https://stackoverflow.com/questions/38076783/
binding-to-last-array-element

using Insight.Models;
using System.Linq;
using System.Windows.Data;
using System.Collections.ObjectModel;

namespace Insight.Services.Converters
{
    class LastItemTimeStampConverter : IValueConverter
    {
        public object Convert(object value, System.Type targetType,
        object parameter, System.Globalization.CultureInfo culture)
        {
            ObservableCollection<Reading> readings = value as
            ObservableCollection<Reading>;
            if (readings != null)
            {
                return readings.LastOrDefault().TimeStamp;
            }
            else return Binding.DoNothing;
        }

        public object ConvertBack(object value, System.Type targetType,
        object parameter, System.Globalization.CultureInfo culture)
        {
            throw new System.NotImplementedException();
        }
    }
}
```

Insight/Services/Converters/ UnixToDateTimeConverter.cs

```csharp
using System;

namespace Insight.Services.Converters
{
    public class UnixToDateTimeConverter
    {
        public static DateTime Convert(double unixTimeStamp)
        {
            // Unix timestamp is defined as seconds after January 1st 1970
            // Used to set a global timestamp to microcontroller readings
            DateTime dateTime = new DateTime(1970, 1, 1, 0, 0,
            0, DateTimeKind.Utc);
            dateTime = dateTime.AddSeconds(unixTimeStamp).ToLocalTime();

            return dateTime;
        }
    }
}
```

Insight/Services/FirebaseAuthStore.cs

```
using Firebase.Auth;
using Firebase.Auth.Providers;
using Firebase.Auth.Repository;
using Firebase.Auth.UI;
using Firebase.Database;
using System;
using System.Configuration;
using System.Threading.Tasks;

namespace Insight.Services
{
    public class FirebaseAuthStore
    {
        public event Action UIConfigChanged;
        public event Action AuthConfigChanged;
        public event Action ClientChanged;

        private FirebaseUIConfig uiConfig;
        public FirebaseUIConfig UIConfig
        {
            get => uiConfig;
            set
            {
                uiConfig = value;
                OnUIConfigChanged();
            }
        }

        private FirebaseAuthConfig authConfig;
        public FirebaseAuthConfig AuthConfig
        {
            get => authConfig;
            set
            {
                authConfig = value;
                OnAuthConfigChanged();
            }
        }

        private FirebaseClient client;
        public FirebaseClient Client
        {
            get => client;
            set
            {
                client = value;
                OnClientChanged();
            }
        }

        private Task<string> clientToken;
        public Task<string> ClientToken
        {
            get => clientToken;
            set
            {
                clientToken = value;
            }
        }

        public FirebaseAuthStore()
        {
            string APIstring = ConfigurationManager.AppSettings.Get("");
            string firebaseURL = ConfigurationManager.AppSettings.Get("");
            string firebaseURI = ConfigurationManager.AppSettings.Get("");

            uiConfig = new FirebaseUIConfig
            {
                ApiKey = APIstring,
                AuthDomain = firebaseURL,
                Providers = new FirebaseAuthProvider[]
                {
                    new GoogleProvider(),
                    new EmailProvider()
                },
                PrivacyPolicyUrl = "",
                TermsOfServiceUrl = "t",
                IsAnonymousAllowed = false,
                AutoUpgradeAnonymousUsers = true,
                UserRepository = new FileUserRepository("Insight"),
                AnonymousUpgradeConflict = conflict =>
                conflict.SignInWithPendingCredentialAsync(true)
            };

            authConfig = new FirebaseAuthConfig
            {
                ApiKey = APIstring,
                AuthDomain = firebaseURL,
                Providers = new FirebaseAuthProvider[]
                {
                    new GoogleProvider(),
                    new EmailProvider()
                },
                UserRepository = new FileUserRepository("Insight"),
            };
            client = new FirebaseClient(firebaseURI);
        }

        private void OnUIConfigChanged()
        {
            UIConfigChanged?.Invoke();
        }

        private void OnAuthConfigChanged()
        {
            AuthConfigChanged?.Invoke();
        }

        private void OnClientChanged()
        {
            ClientChanged?.Invoke();
        }
    }
}
```

Insight/Services/FirebaseDataService.cs

```
using Firebase.Database;
using Firebase.Database.Query;
using GalaSoft.MvvmLight.Messaging;
using Insight.Models;
using Insight.Services.Commands;
using Insight.Services.Converters;
using OxyPlot;
using OxyPlot.Axes;
using OxyPlot.Series;
using System;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;

namespace Insight.Services
{
    public class FirebaseDataService : INotifyPropertyChanged
    {
        public static ObservableCollection<Sensor> Sensors =
        new ObservableCollection<Sensor>();
        public static ObservableCollection<Reading> SensorReadings =
        new ObservableCollection<Reading>();
        public event PropertyChangedEventHandler PropertyChanged;

        private string updateMessage;
        public string UpdateMessage
        {
            get => updateMessage;
            set
            {
                updateMessage = value;
                OnPropertyChanged(nameof(UpdateMessage));
            }
        }

        public static async Task<ObservableCollection<Sensor>>
        GetSensorsAsync(FirebaseAuthStore firebaseAuthStore,
        NavigationStore navigationStore)
        {
            try
            {
                Sensors.Clear();

                var tempList = (await firebaseAuthStore.Client
                    .Child("")
                    .OnceAsync<Sensor>())
                    .Select(s => new Sensor
                    {
                        ID = s.Key,
                        Name = s.Object.Name,
                        StationName = s.Object.StationName,
                        ReadingUnit = s.Object.ReadingUnit,
                        ReadingType = s.Object.ReadingType,
                        EnableReadingValueLimits = s.Object.EnableReadingValueLimits,
                        ReadingValueUpperLimit = s.Object.ReadingValueUpperLimit,
                        ReadingValueLowerLimit = s.Object.ReadingValueLowerLimit
                    }).ToList();
                Debug.WriteLine(Sensors.Count);
                Debug.WriteLine(tempList.Count);

                foreach (var temp in tempList)
                {
                    temp.NavigateSensorCommand = new NavigateSensorCommand(
                    navigationStore, firebaseAuthStore, temp);
                    temp.NavigateSensorSettingsCommand = new NavigateSensorSettingsCommand(
                    navigationStore, firebaseAuthStore, temp);
                    Sensors.Add(temp);
                }
            }
            catch (FirebaseException ex)
            {
                Debug.WriteLine(ex);
            }

            await Task.Delay(TimeSpan.FromSeconds(5)).ConfigureAwait(false);

            return Sensors;
        }

        public static async Task<ObservableCollection<Reading>>
        GetReadingsAsync(FirebaseAuthStore firebaseAuthStore,
        string sensorKey)
        {
            try
            {
                SensorReadings.Clear();

                var tempList = (await firebaseAuthStore.Client
                    .Child(sensorKey + "/readings")
                    .OrderByKey()
                    .LimitToLast(100)
                    .OnceAsync<double>())
                    .Select(r => new Reading
                    {
                        TimeStamp = UnixToDateTimeConverter.
                        Convert(
                        Convert.ToDouble(r.Key)),
                        Value = Math.Round(r.Object, 2)
                    }).ToList();

                foreach (var temp in tempList)
                {
                    SensorReadings.Add(temp);
                }
            }
            catch (FirebaseException ex)
            {
                Debug.WriteLine(ex);
            }

            await Task.Delay(TimeSpan.FromSeconds(5)).ConfigureAwait(false);

            return SensorReadings;
        }

        public static async Task<PlotModel> GetPointPlotModelAsync(
        FirebaseAuthStore firebaseAuthStore, Sensor sensor)
        {
            var plotModel = new PlotModel();
            plotModel.InvalidatePlot(true);
            plotModel.Axes.Clear();
            plotModel.Series.Clear();
            //plotModel.Title = "Firebase Readings";

            var plotSeries = generateLineSeries();

            var tempList = (await firebaseAuthStore.Client
                    .Child(sensor.ID + "/readings")
                    .OrderByKey()
                    .LimitToLast(40)
                    .OnceAsync<double>())
                    .Select(r => new Reading
                    {
                        TimeStamp = UnixToDateTimeConverter.Convert(
                        Convert.ToDouble(r.Key)),
                        Value = Math.Round(r.Object, 2)
                    }).ToList();
            plotModel.Axes.Add(new DateTimeAxis
            {
                Position = AxisPosition.Bottom,
                StringFormat = "HH:mm\ndd/MM/yy",
                MinorIntervalType = DateTimeIntervalType.Minutes,
                IntervalType = DateTimeIntervalType.Minutes
            });

            foreach (var r in tempList)
            {
                plotSeries.Points.Add(new DataPoint(DateTimeAxis.
                ToDouble(r.TimeStamp), r.Value));
            }

            plotModel.Axes.Add(new LinearAxis {
                Position = AxisPosition.Left,
                Title = sensor.ReadingUnit,
                Maximum = plotSeries.Points.Select(v => v.Y).Max(),
                Minimum = plotSeries.Points.Select(v => v.Y).Min()
            });

            plotModel.Series.Add(plotSeries);

            return plotModel;
        }

        // TODO: Implement bar chart in next version
        private static LinearBarSeries generateLinearBarSeries()
        {

            return new LinearBarSeries
            {
                BarWidth = 20,
                FillColor = OxyColor.FromRgb(207, 48, 84)
            };
        }

        private static LineSeries generateLineSeries()
        {
            return new LineSeries
            {
                Color = OxyColor.FromRgb(207, 48, 84),
                MarkerType = MarkerType.Circle,
                MarkerSize = 3,
                MarkerStroke = OxyColor.FromRgb(207, 48, 84)
            };
        }

        public static async Task<ObservableCollection<Sensor>>
        GetSensorsWithReadingsAsync(FirebaseAuthStore
        firebaseAuthStore, NavigationStore navigationStore)
        {
            try
            {
                Sensors.Clear();

                var tempSensorsList = (await firebaseAuthStore.Client
                    .Child("")
                    .OnceAsync<Sensor>())
                    .Select(s => new Sensor
                    {
                        ID = s.Key,
                        Name = s.Object.Name,
                        StationName = s.Object.StationName,
                        ReadingUnit = s.Object.ReadingUnit,
                        ReadingType = s.Object.ReadingType,
                        EnableReadingValueLimits = s.Object.EnableReadingValueLimits,
                        ReadingValueUpperLimit = s.Object.ReadingValueUpperLimit,
                        ReadingValueLowerLimit = s.Object.ReadingValueLowerLimit
                    }).ToList();
                Debug.WriteLine(Sensors.Count);
                Debug.WriteLine(tempSensorsList.Count);

                foreach (var temp in tempSensorsList)
                {
                    var tempReadingList = (await firebaseAuthStore.Client
                    .Child(temp.ID + "/readings")
                    .OrderByKey()
                    .LimitToLast(100)
                    .OnceAsync<double>())
                    .Select(r => new Reading
                    {
                        TimeStamp = UnixToDateTimeConverter.Convert(Convert.ToDouble(r.Key)),
                        Value = Math.Round(r.Object, 2)
                    }).ToList();


                    temp.ReadingList = new ObservableCollection<Reading>(
                    tempReadingList);
                    temp.NavigateSensorCommand = new NavigateSensorCommand(
                    navigationStore, firebaseAuthStore, temp);
                    temp.NavigateSensorSettingsCommand = new
                    NavigateSensorSettingsCommand(navigationStore, firebaseAuthStore, temp);
                    Sensors.Add(temp);
                }
            }
            catch (FirebaseException ex)
            {
                Debug.WriteLine(ex);
            }

            await Task.Delay(TimeSpan.FromSeconds(5)).ConfigureAwait(false);

            return Sensors;
        }

        public async Task<bool> UpdateItemAsync(FirebaseAuthStore
        firebaseAuthStore, string id, string item)
        {
            try
            {
                await firebaseAuthStore.Client
                    .Child(id)
                    .PutAsync(item);

                UpdateMessage = "Stored to database successfully!";
                Messenger.Default.Send(new SensorUpdateMessage(UpdateMessage));
            }
            catch (Exception ex)
            {
                UpdateMessage = ex.Message;
                Messenger.Default.Send(new ErrorMessage(UpdateMessage));
                return false;
            }

            return true;
        }

        public void OnPropertyChanged(string propertyName = null)
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}
```

Insight/Services/FirebaseDataStore.cs

```
using Insight.Models;
using System;
using System.Windows.Threading;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Threading.Tasks;

namespace Insight.Services
{
    public class FirebaseDataStore : INotifyPropertyChanged
    {
        private readonly NavigationStore navigationStore;
        private readonly FirebaseAuthStore firebaseAuthStore;
        private DispatcherTimer dispatcherTimer = new DispatcherTimer();

        public event Action SensorsChanged;

        private ObservableCollection<Sensor> sensors;
        public ObservableCollection<Sensor> Sensors
        {
            get => sensors;
            set
            {
                sensors = value;
                OnPropertyChanged(nameof(Sensors));
                OnSensorsChanged();
            }
        }

        public FirebaseDataStore(FirebaseAuthStore firebaseAuthStore,
        NavigationStore navigationStore)
        {
            this.navigationStore = navigationStore;
            this.firebaseAuthStore = firebaseAuthStore;

            Sensors = initiateSensors().Result;
            dispatcherTimer.Tick += new EventHandler(dispatcherTimerTick);
            dispatcherTimer.Interval = new TimeSpan(0, 0, 20);
            dispatcherTimer.Start();
        }

        public event PropertyChangedEventHandler PropertyChanged;
        public void OnPropertyChanged(string propertyName = null)
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
        }

        public void OnSensorsChanged()
        {
            SensorsChanged?.Invoke();
        }

        private async void dispatcherTimerTick(object sender, EventArgs e)
        {
            Sensors = await getSensorsAsync();
        }

        private NotifyTaskCompletion<ObservableCollection<Sensor>> initiateSensors()
        {
            var notifyTaskCompletion = new NotifyTaskCompletion<
            ObservableCollection<Sensor>>(

                FirebaseDataService.GetSensorsWithReadingsAsync(fi
                rebaseAuthStore, navigationStore));

            return notifyTaskCompletion;
        }

        private async Task<ObservableCollection<Sensor>> getSensorsAsync()
        {
            var sensors = await FirebaseDataService.GetSensorsWithReadingsAsync(
                    firebaseAuthStore, navigationStore);
            return new ObservableCollection<Sensor>(sensors);
        }
    }
}
```

Insight/Services/NavigationStore.cs

```
using Insight.ViewModels;
using System;
using System.Collections.Generic;
using System.Text;

namespace Insight.Services
{
    public class NavigationStore
    {
        public event Action CurrentViewModelChanged;

        private ViewModelBase currentViewModel;
        public ViewModelBase CurrentViewModel
        {
            get => currentViewModel;
            set
            {
                currentViewModel = value;
                OnCurrentViewModelChanged();
            }
        }

        private ViewModelBase sideBarViewModel;
        public ViewModelBase SideBarViewModel
        {
            get => sideBarViewModel;
            set => sideBarViewModel = value;
        }

        private void OnCurrentViewModelChanged()
        {
            CurrentViewModelChanged?.Invoke();
        }
    }
}
```

Insight/Services/NotifyTaskCompletion.cs

```
/// Helper class from the Official Microsoft documentation
/// for MVVM applications.
///
/// Available at https://docs.microsoft.com/en-us/archive/msdn-magazine/2014/
march/async-programming-patterns-for-asynchronous-mvvm-
applications-data-binding
/// Example by Stephen Cleary (fetched April 30th 2022)

using System;
using System.ComponentModel;
using System.Threading.Tasks;
public sealed class NotifyTaskCompletion<TResult> : INotifyPropertyChanged
{
    public NotifyTaskCompletion(Task<TResult> task)
    {
        Task = task;
        if (!task.IsCompleted)
        {
            var _ = WatchTaskAsync(task);
        }
    }
    private async Task WatchTaskAsync(Task task)
    {
        try
        {
            await task;
        }
        catch
        {
        }
        var propertyChanged = PropertyChanged;
        if (propertyChanged == null)
            return;
        propertyChanged(this, new PropertyChangedEventArgs("Status"));
        propertyChanged(this, new PropertyChangedEventArgs("IsCompleted"));
        propertyChanged(this, new PropertyChangedEventArgs("IsNotCompleted"));
        if (task.IsCanceled)
        {
            propertyChanged(this, new PropertyChangedEventArgs("IsCanceled"));
        }
        else if (task.IsFaulted)
        {
            propertyChanged(this, new PropertyChangedEventArgs("IsFaulted"));
            propertyChanged(this, new PropertyChangedEventArgs("Exception"));
            propertyChanged(this,
              new PropertyChangedEventArgs("InnerException"));
            propertyChanged(this, new PropertyChangedEventArgs("ErrorMessage"));
        }
        else
        {
            propertyChanged(this,
              new PropertyChangedEventArgs("IsSuccessfullyCompleted"));
            propertyChanged(this, new PropertyChangedEventArgs("Result"));
        }
    }
    public Task<TResult> Task { get; private set; }
    public TResult Result
    {
        get
        {
            return (Task.Status == TaskStatus.RanToCompletion) ?
                Task.Result : default(TResult);
        }
    }
    public TaskStatus Status { get { return Task.Status; } }
    public bool IsCompleted { get { return Task.IsCompleted; } }
    public bool IsNotCompleted { get { return !Task.IsCompleted; } }
    public bool IsSuccessfullyCompleted
    {
        get
        {
            return Task.Status ==
                TaskStatus.RanToCompletion;
        }
    }
    public bool IsCanceled { get { return Task.IsCanceled; } }
    public bool IsFaulted { get { return Task.IsFaulted; } }
    public AggregateException Exception { get { return Task.Exception; } }
    public Exception InnerException
    {
        get
        {
            return (Exception == null) ?
                null : Exception.InnerException;
        }
    }
    public string ErrorMessage
    {
        get
        {
            return (InnerException == null) ?
```

```
                null : InnerException.Message;
        }
    }
    public event PropertyChangedEventHandler PropertyChanged;
}
```

Insight/ViewModels/ViewModelBase.cs

```
using System;
using System.ComponentModel;
using System.Windows.Input;

namespace Insight.ViewModels
{
    public class ViewModelBase : INotifyPropertyChanged, IDisposable
    {
        public string ViewTitle { get; set; }

        public event PropertyChangedEventHandler PropertyChanged;

        public void OnPropertyChanged(string propertyName = null)
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
        }

        public virtual void Dispose() { }
    }
}
```

Insight/ViewModels/HomeViewModel.cs

```
using Firebase.Auth.UI;
using GalaSoft.MvvmLight.Ioc;
using Insight.Services;
using OxyPlot;
using System;
using System.Diagnostics;
using System.Windows.Threading;

namespace Insight.ViewModels
{
    class HomeViewModel : ViewModelBase
    {
        public FirebaseDataStore FirebaseDataStore { get; private set; }

        public HomeViewModel(FirebaseAuthStore firebaseAuthStore,
        NavigationStore navigationStore)
        {
            var userName = FirebaseUI.Instance.Client.User.Info.DisplayName;
            ViewTitle = "Welcome to Insight, " + userName + "!";

            FirebaseDataStore = SimpleIoc.Default.GetInstance<FirebaseDataStore>();
        }
    }
}
```

Insight/ViewModels/InfoViewModel.cs

```
using Insight.Services;
using System;
using System.ComponentModel;

namespace Insight.ViewModels
{
    class InfoViewModel : ViewModelBase
    {

        public InfoViewModel(NavigationStore navigationStore)
        {
            ViewTitle = "How to use Insight";

        }
    }
}
```

Insight/ViewModels/LoginViewModel.cs

```
using Firebase.Auth;
using Insight.Services;
using System.ComponentModel;
using System.Diagnostics;
using System.Windows;

namespace Insight.ViewModels
{
    class LoginViewModel : ViewModelBase
    {
        public LoginViewModel(NavigationStore navigationStore)
        {
        }
    }
}
```

Insight/ViewModels/MainWindowViewModel.cs

```
using Firebase.Auth;
using Firebase.Auth.UI;
using GalaSoft.MvvmLight.Ioc;
using GalaSoft.MvvmLight.Messaging;
using Insight.Models;
using Insight.Services;
using Insight.Services.Commands;
using System;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Diagnostics;
using System.Windows;
using System.Windows.Input;
using ToastNotifications.Core;

namespace Insight.ViewModels
{
    class MainWindowViewModel : ViewModelBase, INotifyPropertyChanged
    {
        private readonly NavigationStore navigationStore;
        private readonly FirebaseAuthStore firebaseAuthStore;
        private readonly ToastNotificationViewModel toastNotificationViewModel;
        private int toastNotificationCounter = 0;
        private string lastMessage;
        public FirebaseDataStore FirebaseDataStore { get; private set; }
        public ObservableCollection<ViewModelBase> Pages { get; }
        public ViewModelBase CurrentViewModel => navigationStore.CurrentViewModel;
        public ViewModelBase MenuViewModel => navigationStore.SideBarViewModel;
        public ObservableCollection<Sensor> Sensors { get; }
        public ICommand NavigateLoginCommand { get; }
        public ICommand NavigateHomeCommand { get; }
        public ICommand SignOutCommand { get; }
        public User User { get; set; }

        public MainWindowViewModel(NavigationStore navigationStore,
        FirebaseAuthStore firebaseAuthStore)
        {
            this.navigationStore = navigationStore;
            this.navigationStore.CurrentViewModelChanged +=
            OnCurrentViewModelChanged;

            NavigateLoginCommand = new NavigateLoginCommand(navigationStore);
            NavigateHomeCommand = new
            NavigateHomeCommand(firebaseAuthStore, navigationStore);

            // Setup for Toast style notifications
            toastNotificationViewModel = new ToastNotificationViewModel();
            Messenger.Default.Register<SensorUpdateMessage>(
                this,
                message =>
                {
                    ShowMessage(toastNotificationViewModel.
                    ShowSuccess, message.Message);
                });
            Messenger.Default.Register<ErrorMessage>(
                this,
                message =>
                {
                    ShowMessage(toastNotificationViewModel.ShowError,
                    message.Message);
                });

            // Setup for Firebase authentication
            this.firebaseAuthStore = firebaseAuthStore;
            FirebaseUI.Initialize(firebaseAuthStore.UIConfig);
            FirebaseUI.Instance.Client.AuthStateChanged += this.AuthStateChanged;

            // Setup Firebase storage singleton
            FirebaseDataStore = SimpleIoc.Default.GetInstance<FirebaseDataStore>();
            this.FirebaseDataStore.SensorsChanged += OnSensorsChanged;

            SignOutCommand = new SignOutCommand(FirebaseUI.Instance.Client);
        }

        private void OnSensorsChanged()
        {
            foreach (var sensor in FirebaseDataStore.Sensors)
            {
                if (sensor.EnableReadingValueLimits == "true")
                {
                    foreach (var reading in sensor.ReadingList)
                    {
                        if ((DateTime.Now - reading.TimeStamp).TotalSeconds < 35)
                        {
                            // Test
                            if (reading.Value < sensor.ReadingValueLowerLimit)
                            {
                                Debug.WriteLine("Lower value limit exceeded!");
                                ShowMessage(toastNotificationViewModel.
                                ShowWarning, "Low value: " +
                                reading.Value.ToString() + " " +
                                sensor.ReadingUnit);
                            }
                            else if (reading.Value > sensor.ReadingValueUpperLimit)
                            {
                                Debug.WriteLine("Upper value limit exceeded!");
                                ShowMessage(toastNotificationViewModel.
                                ShowWarning, "High value: " +
                                reading.Value.ToString() + " " +
                                sensor.ReadingUnit);
                            }
                        }
                    }
                }
            }
        }

        private void OnCurrentViewModelChanged()
        {
            OnPropertyChanged(nameof(CurrentViewModel));
        }

        public void ShowMessage(Action<string, MessageOptions> action,
        string messageType)
        {
            MessageOptions opts = new MessageOptions
            {
                CloseClickAction = CloseNotificationAction,
                Tag = $"[This is Tag Value ({++toastNotificationCounter})]",
                FreezeOnMouseEnter = true,
                UnfreezeOnMouseLeave = true,
                ShowCloseButton = true
            };
            lastMessage = $"{toastNotificationCounter} {messageType}";
            action(lastMessage, opts);
        }

        private void CloseNotificationAction(NotificationBase obj)
        {
            var opts = obj.DisplayPart.Notification.Options;
        }

        private void AuthStateChanged(object sender, UserEventArgs e)
        {
            User = e.User;

            Application.Current.Dispatcher.Invoke((Action)(() =>
            {
                if (e.User == null)
```

```
                {
                    NavigateLoginCommand.Execute((object)null);
                }
                else
                {
                    firebaseAuthStore.ClientToken = FirebaseUI.Instance.
                    Client.User.GetIdTokenAsync();
                    NavigateHomeCommand.Execute((object)null);
                }
            }));
        }
    }
}
```

Insight/ViewModels/SensorSettingsViewModel.cs

```
using Insight.Models;
using Insight.Services;
using Insight.Services.Commands;
using System.Diagnostics;
using System.Windows.Input;

namespace Insight.ViewModels
{
    public class SensorSettingsViewModel : ViewModelBase
    {
        public ICommand NavigateSensorCommand { get; }
        private Sensor sensor;
        public Sensor Sensor
        {
            get => sensor;
            set { sensor = value; }
        }

        public FirebaseDataService FirebaseDataService { get; private set; }

        private SaveSettingsCommand saveSettingsCommand;
        public SaveSettingsCommand SaveSettingsCommand
        {
            get => saveSettingsCommand;
            set
            {
                saveSettingsCommand = value;
                OnPropertyChanged(nameof(SaveSettingsCommand));
            }
        }

        private bool maxMinReadingsEnabled;
        public bool MaxMinReadingsEnabled
        {
            get => maxMinReadingsEnabled;
            set
            {
                maxMinReadingsEnabled = value;
                OnPropertyChanged(nameof(MaxMinReadingsEnabled));
            }
        }

        private int maxReadingValue;
        public int MaxReadingValue
        {
            get => maxReadingValue;
            set
            {
                maxReadingValue = value;
                OnPropertyChanged(nameof(MaxReadingValue));
                updateCommandData();
                Debug.WriteLine("maxchange");
            }
        }

        private int minReadingValue;
        public int MinReadingValue
        {
            get => minReadingValue;
            set
            {
                minReadingValue = value;
                OnPropertyChanged(nameof(MinReadingValue));
                updateCommandData();
                Debug.WriteLine("minchange");
            }
        }

        public SensorSettingsViewModel(FirebaseAuthStore firebaseAuthStore, Sensor sensor)
        {
            ViewTitle = "Sensor Settings";
            this.sensor = sensor;
            FirebaseDataService = new FirebaseDataService();
            saveSettingsCommand = new
            SaveSettingsCommand(FirebaseDataService, firebaseAuthStore);

            maxMinReadingsEnabled = (sensor.EnableReadingValueLimits == "true")
            ? true : false;
            maxReadingValue = sensor.ReadingValueUpperLimit;
            minReadingValue = sensor.ReadingValueLowerLimit;
        }

        private void updateCommandData()
        {
            saveSettingsCommand.Id_0 = sensor.ID + "/enablereadingvaluelimits";
            saveSettingsCommand.Item_0 = maxMinReadingsEnabled ? "true" : "false";
            saveSettingsCommand.Id_1 = sensor.ID + "/maxreadingvalue";
            saveSettingsCommand.Item_1 = maxReadingValue.ToString();
            saveSettingsCommand.Id_2 = sensor.ID + "/minreadingvalue";
            saveSettingsCommand.Item_2 = minReadingValue.ToString();
        }
```

```
    }
}
```

Insight/ViewModels/SensorViewModel.cs

```
using GalaSoft.MvvmLight.Ioc;
using Insight.Models;
using Insight.Services;
using OxyPlot;
using System;
using System.Linq;
using System.Windows.Input;
using System.Windows.Threading;
using System.Collections.ObjectModel;
using System.Diagnostics;

namespace Insight.ViewModels
{
    class SensorViewModel : ViewModelBase
    {
        public ICommand NavigateSensorSettingsCommand { get; private set; }
        public FirebaseDataStore FirebaseDataStore { get; private set; }

        // Update using DispatcherTimer as FirebaseDatabase's
        // subscribe method is unreliable
        private DispatcherTimer dispatcherTimer = new DispatcherTimer();
        private FirebaseAuthStore firebaseAuthStore;

        private NotifyTaskCompletion<ObservableCollection<Reading>> sensorReadings;
        public NotifyTaskCompletion<ObservableCollection<Reading>> SensorReadings
        {
            get => sensorReadings;
            set
            {
                sensorReadings = value;
                OnPropertyChanged(nameof(SensorReadings));
            }
        }

        private NotifyTaskCompletion<PlotModel> plotModel;
        public NotifyTaskCompletion<PlotModel> PlotModel
        {
            get => plotModel;
            set
            {
                plotModel = value;
                OnPropertyChanged(nameof(PlotModel));
            }
        }

        private Sensor sensor;
        public Sensor Sensor
        {
            get => sensor;
            set
            {
                sensor = value;
                OnPropertyChanged(nameof(Sensor));
            }
        }

        private Sensor updatedInfoSensor;
        public Sensor UpdatedInfoSensor
        {
            get => updatedInfoSensor;
            set
            {
                updatedInfoSensor = value;
                OnPropertyChanged(nameof(UpdatedInfoSensor));
            }
        }

        private int sumReadings;
        public int SumReadings
        {
            get => sumReadings;
            set
            {
                sumReadings = value;
                OnPropertyChanged(nameof(SumReadings));
            }
        }

        private double sumReadingValues;
        public double SumReadingValues
        {
            get => sumReadingValues;
            set
            {
                sumReadingValues = value;
                OnPropertyChanged(nameof(SumReadingValues));
            }
        }

        private double averageReadingValue;
        public double AverageReadingValue
        {
            get => averageReadingValue;
            set
            {
                averageReadingValue = value;
                OnPropertyChanged(nameof(AverageReadingValue));
            }
        }

        public SensorViewModel(FirebaseAuthStore firebaseAuthStore, Sensor sensor)
        {
            this.firebaseAuthStore = firebaseAuthStore;
            this.sensor = sensor;

            ViewTitle = sensor.Name;

            NavigateSensorSettingsCommand = sensor.NavigateSensorSettingsCommand;
            FirebaseDataStore = SimpleIoc.Default.GetInstance<FirebaseDataStore>();
            this.FirebaseDataStore.SensorsChanged += OnSensorsChanged;

            updatePlots();
            dispatcherTimer.Tick += new EventHandler(dispatcherTimerTick);
            dispatcherTimer.Interval = new TimeSpan(0, 0, 20);
            dispatcherTimer.Start();
        }

        private void dispatcherTimerTick(object sender, EventArgs e)
        {
            updatePlots();
        }

        private void OnSensorsChanged()
        {
            UpdatedInfoSensor = FirebaseDataStore.Sensors.FirstOrDefault(
            s => s.ID == sensor.ID);
            SumReadings = UpdatedInfoSensor.ReadingList.Count;
            SumReadingValues = Math.Round(UpdatedInfoSensor.ReadingList.
            Sum(r => r.Value), 2);
            AverageReadingValue = Math.Round(SumReadingValues / SumReadings, 2);
        }

        private void updatePlots()
        {
            sensorReadings = new NotifyTaskCompletion<ObservableCollection<Reading>>(
                FirebaseDataService.GetReadingsAsync(firebaseAuthStore, sensor.ID));

            plotModel = new NotifyTaskCompletion<PlotModel>(
                FirebaseDataService.GetPointPlotModelAsync(firebaseAuthStore, sensor));

            // Fire PropertyChanged event for entire view on each Dispatcher tick
            OnPropertyChanged(nameof(PlotModel));
        }
    }
}
```

Insight/ViewModels/SideBarViewModel.cs

```
using GalaSoft.MvvmLight.Ioc;
using Insight.Models;
using Insight.Services;
using Insight.Services.Commands;
using System.Windows.Input;
using System.Collections.ObjectModel;

namespace Insight.ViewModels
{
    public class SideBarViewModel : ViewModelBase
    {
        public FirebaseDataStore FirebaseDataStore { get; private set; }
        public ICommand NavigateHomeCommand { get; }
        public ICommand NavigateInfoCommand { get; }
        public ICommand OpenHttpLinkCommand { get; }

        private NotifyTaskCompletion<ObservableCollection<Sensor>> sensors;
        public NotifyTaskCompletion<ObservableCollection<Sensor>> Sensors
        {
            get => sensors;
            set
            {
                sensors = value;
                OnPropertyChanged();
            }
        }

        public SideBarViewModel(NavigationStore navigationStore,
        FirebaseAuthStore firebaseAuthStore)
        {
            ViewTitle = "Sidebar";
            NavigateHomeCommand = new NavigateHomeCommand(firebaseAuthStore,
            navigationStore);
            NavigateInfoCommand = new NavigateInfoCommand(navigationStore);
            OpenHttpLinkCommand = new OpenHttpLinkCommand();

            FirebaseDataStore = SimpleIoc.Default.GetInstance<FirebaseDataStore>();
        }
    }
}
```

Insight/ViewModels/ToastNotificationViewModel.cs

```
using System;
using System.ComponentModel;
using ToastNotifications;
using ToastNotifications.Core;
using ToastNotifications.Messages;
using ToastNotifications.Position;
using ToastNotifications.Lifetime;
using ToastNotifications.Lifetime.Clear;

namespace Insight.ViewModels
{
    public class ToastNotificationViewModel : INotifyPropertyChanged
    {
        private readonly Notifier notifier;
        public ToastNotificationViewModel()
        {
            notifier = new Notifier(cfg =>
            {
                cfg.PositionProvider = new WindowPositionProvider(
                    parentWindow: App.Current.MainWindow,
                    corner: Corner.BottomRight,
                    offsetX: 25,
                    offsetY: 50);

                cfg.LifetimeSupervisor = new TimeAndCountBasedLifetimeSupervisor(
                    notificationLifetime: TimeSpan.FromSeconds(8),
                    maximumNotificationCount: MaximumNotificationCount.FromCount(4));

                cfg.Dispatcher = App.Current.Dispatcher;

                cfg.DisplayOptions.TopMost = false;
                cfg.DisplayOptions.Width = 300;
            });

            notifier.ClearMessages(new ClearAll());
        }

        public event PropertyChangedEventHandler PropertyChanged;

        protected virtual void OnPropertyChanged(string propertyName = null)
        {
            var handler = PropertyChanged;
            handler?.Invoke(this, new PropertyChangedEventArgs(propertyName));
        }

        // Unloaded event primarily used in code-behind in non-MVVM apps
        public void OnUnloaded()
        {
            notifier.Dispose();
        }

        public void ShowInformation(string message, MessageOptions messageOptions)
        {
            notifier.ShowInformation(message, messageOptions);
        }

        public void ShowSuccess(string message, MessageOptions messageOptions)
        {
            notifier.ShowSuccess(message, messageOptions);
        }

        public void ShowWarning(string message, MessageOptions messageOptions)
        {
            notifier.ShowWarning(message, messageOptions);
        }

        public void ShowError(string message, MessageOptions messageOptions)
        {
            notifier.ShowError(message, messageOptions);
        }

        public void ClearMessages(string message)
        {
            notifier.ClearMessages(new ClearByMessage(message));
        }

        public void ClearAll()
        {
            notifier.ClearMessages(new ClearAll());
        }
    }
}
```

Insight/Views/HomeView.xaml

```
<UserControl x:Class="Insight.Views.HomeView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2
             ↪ 006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:Insight.Views"
             xmlns:converters="clr-namespace:Insight.Services.Converters"
             xmlns:oxy="http://oxyplot.org/wpf"
             mc:Ignorable="d"
             Background="WhiteSmoke"
             d:DesignHeight="450" d:DesignWidth="800">
    <UserControl.Resources>
        <converters:LastItemConverter x:Key="LastItemConverter" />
        <converters:LastItemTimeStampConverter
        ↪ x:Key="LastItemTimeStampConverter" />
    </UserControl.Resources>
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="*" />
            <RowDefinition Height="2*" />
            <RowDefinition Height="6*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="6*" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="5*" />
            <ColumnDefinition Width="5*" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>
        <Label Grid.Row="1"
               Grid.Column="1"
               Grid.ColumnSpan="2"
               Content="{Binding ViewTitle}"
               Style="{StaticResource TitleLabelDark}"/>
        <ItemsControl Grid.Row="2"
                      Grid.Column="1"
                      Grid.RowSpan="3"
                      Grid.ColumnSpan="2"
                      ItemsSource="{Binding Path=FirebaseDataStore.Sensors}">
            <ItemsControl.ItemTemplate>
```

```
                <DataTemplate>
                    <Border x:Name="SensorCard"
                            Width="400"
                            Height="210"
                            CornerRadius="10"
                            Background="White"
                            Margin="20, 0, 0, 20">
                        <Border.BitmapEffect>
                            <DropShadowBitmapEffect Color="LightGray"
                            Direction="270" ShadowDepth="1"
                            Opacity="1" Softness="2" />
                        </Border.BitmapEffect>
                        <Button Command="{Binding Path=NavigateSensorCommand}">
                            <StackPanel Orientation="Vertical"
                                        HorizontalAlignment="Left"
                                        Margin="10, 10, 10, 10">
                                <Label Content="{Binding Path=Name}"
                                       Style="{StaticResource TitleLabelDark}"
                                       ↪  />
                                <Label Content="{Binding Path=StationName}"
                                       Style="{StaticResource
                                       ↪  SubTitleLabelOpaqueDark}" />
                                <Label Margin="0, 10, 0, 0"
                                       Content="Latest reading"
                                       Style="{StaticResource
                                       ↪  SubSubSubTitleLabelOpaqueDark}" />
                                <StackPanel Orientation="Horizontal">
                                    <Label Content="{Binding Path=ReadingList,
                                    Converter={StaticResource
↪  LastItemConverter}}"
                                           Style="{StaticResource
                                           ↪  TitleLabelDark}" />
                                    <Label Content="{Binding Path=ReadingUnit}"
                                           Style="{StaticResource
                                           ↪  TitleLabelDark}" />
                                </StackPanel>
                                <Label Content="{Binding Path=ReadingList,
                                ↪  Converter={StaticResource
                                ↪  LastItemTimeStampConverter}}"
                                       Style="{StaticResource
                                       ↪  SubSubTitleLabelOpaqueDark}" />
                            </StackPanel>
                            <Button.Style>
                                <Style TargetType="{x:Type Button}">
                                    <Setter Property="Background"
                                    ↪  Value="Transparent"/>
                                    <Setter Property="Template">
```

```
                                        <Setter.Value>
                                            <ControlTemplate
                                            ↪   TargetType="Button">
                                                <Border Background="{TemplateBi┘
                                                ↪   nding Background}"
                                                ↪   BorderThickness="1"
                                                ↪   Padding="5">
                                                    <ContentPresenter Horizonta┘
                                                    ↪   lAlignment="Left"
                                                    ↪   VerticalAlignment="Top"
                                                    ↪   />
                                                </Border>
                                            </ControlTemplate>
                                        </Setter.Value>
                                    </Setter>
                                    <Style.Triggers>
                                        <Trigger Property="IsMouseOver"
                                        ↪   Value="True">
                                            <Setter Property="Background"
                                            ↪   Value="{StaticResource
                                            ↪   InsightAquaGreenBrush}" />
                                        </Trigger>
                                    </Style.Triggers>
                                </Style>
                            </Button.Style>
                        </Button>
                    </Border>
                </DataTemplate>
            </ItemsControl.ItemTemplate>
            <ItemsControl.ItemsPanel>
                <ItemsPanelTemplate>
                    <WrapPanel />
                </ItemsPanelTemplate>
            </ItemsControl.ItemsPanel>
        </ItemsControl>
    </Grid>
</UserControl>
```

Insight/Views/InfoView.xaml

```xml
<UserControl x:Class="Insight.Views.InfoView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2
             ↪  006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:Insight.Views"
             mc:Ignorable="d"
             Background="WhiteSmoke"
             d:DesignHeight="450" d:DesignWidth="800">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="*" />
            <RowDefinition Height="2*" />
            <RowDefinition Height="6*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="6*" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="5*" />
            <ColumnDefinition Width="5*" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>
        <Label Grid.Row="1"
               Grid.Column="1"
               Content="{Binding ViewTitle}"
               Style="{StaticResource TitleLabelDark}" />
        <StackPanel Orientation="Vertical"
                    Grid.Row="2"
                    Grid.Column="1"
                    Grid.RowSpan="3"
                    Grid.ColumnSpan="2"
                    CanVerticallyScroll="True">
            <Label Content="Adding Sensors"
                   Style="{StaticResource SubTitleLabelOpaqueDark}" />
            <Label Content="1. Go to the GitHub code library by clicking on the
            ↪  Code button in the sidebar."
                   Style="{StaticResource SubSubTitleLabelDark}" />
            <Label Content="2. Copy the file content corresponding with the
            ↪  microcontroller and sensor setup that you have."
                   Style="{StaticResource SubSubTitleLabelDark}" />
            <Label Content="Available pre-written code is listed in the Readme section in the GitHub library."
                   Margin="17, -5, 0, 0"
                   Style="{StaticResource SubSubTitleLabelDark}" />
            <Image Source="/Assets/copy_raw_github.png"
                   Width="200"
                   Margin="0, 10, 0, 10"
                   HorizontalAlignment="Center" />
            <Label Content="3. Paste the code in the Arduino IDE and make changes to the relevant fields."
                   Style="{StaticResource SubSubTitleLabelDark}" />
            <Label Content="4. Connect the microcontroller to your computer. Pick the correct board in"
                   Style="{StaticResource SubSubTitleLabelDark}" />
            <Label Content="the Arduino IDE Boards Manager."
                   Margin="17, -5, 0, 0"
                   Style="{StaticResource SubSubTitleLabelDark}" />
            <Image Source="/Assets/arduino_ide.png"
                   Width="370"
                   Margin="0, 10, 0, 10"
                   HorizontalAlignment="Center" />
            <Label Content="5. Press the 'Upload' button (arrow to the right) on the top menu to verify"
                   Style="{StaticResource SubSubTitleLabelDark}" />
            <Label Content="and upload code to your microcontroller."
                   Margin="17, -5, 0, 0"
                   Style="{StaticResource SubSubTitleLabelDark}" />
        </StackPanel>
    </Grid>
</UserControl>
```

Insight/Views/LoginView.xaml

```
<UserControl x:Class="Insight.Views.LoginView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2
             ↪  006"
             xmlns:UI="clr-namespace:Firebase.Auth.UI;assembly=Firebase.Auth.UI
             ↪  .WPF"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:Insight.Views"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800">
    <Grid>
        <UI:FirebaseUIControl>
            <UI:FirebaseUIControl.Header>
                <StackPanel>
                    <Image
                        Height="80"
                        Source="/Assets/firebase-logo-vertical.png"
                        />
                    <Label Content="Test"
                           HorizontalAlignment="Center"
                           Padding="10" />
                </StackPanel>
            </UI:FirebaseUIControl.Header>
        </UI:FirebaseUIControl>
    </Grid>
</UserControl>
```

Insight/Views/MainWindow.xaml

```
<Window x:Class="Insight.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Insight"
        xmlns:viewmodels="clr-namespace:Insight.ViewModels"
        xmlns:views="clr-namespace:Insight.Views"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800"
        WindowState="Maximized" WindowStyle="None"
        Background="WhiteSmoke">

    <Window.CommandBindings>
        <CommandBinding Command="ApplicationCommands.Close"
                        Executed="CloseCommandHandler" />
        <CommandBinding Command="ApplicationCommands.New"
                        Executed="MinimizeCommandHandler" />
    </Window.CommandBindings>

    <DockPanel>
        <ContentControl DockPanel.Dock="Left"
                        Content="{Binding MenuViewModel}">
            <ContentControl.Resources>
                <ResourceDictionary>
                    <DataTemplate DataType="{x:Type
                    ↪   viewmodels:SideBarViewModel}">
                        <views:SideBarView />
                    </DataTemplate>
                </ResourceDictionary>
            </ContentControl.Resources>
        </ContentControl>
        <StackPanel DockPanel.Dock="Top"
                    Name="MainStackPanel"
                    Orientation="Horizontal"
                    HorizontalAlignment="Right"
                    Height="30">
            <Button x:Name="UserButton"
                    Command="{Binding SignOutCommand}"
                    Margin="0, 0, 40, 0"
                    IsEnabled="False">
                <StackPanel Orientation="Horizontal">
                    <Image Source="/Assets/man-user.png"
```

```
                           Height="16" />
                    <Label Content="{Binding User.Info.DisplayName}"
                           Margin="0, -4, 0, -4"
                           FontFamily="Arial"
                           FontSize="14"
                           FontWeight="Regular" />
                </StackPanel>
                <Button.Style>
                    <Style TargetType="{x:Type Button}">
                        <Setter Property="Background" Value="Transparent"/>
                        <Setter Property="Template">
                            <Setter.Value>
                                <ControlTemplate TargetType="Button">
                                    <Border Background="{TemplateBinding
                                    ↪   Background}" BorderThickness="1"
                                    ↪   Padding="5">
                                        <ContentPresenter
                                        ↪   HorizontalAlignment="Center"
                                        ↪   VerticalAlignment="Center" />
                                    </Border>
                                </ControlTemplate>
                            </Setter.Value>
                        </Setter>
                        <Style.Triggers>
                            <Trigger Property="IsMouseOver" Value="True">
                                <Setter Property="Background"
                                ↪   Value="{StaticResource
                                ↪   InsightAquaGreenBrush}" />
                            </Trigger>
                        </Style.Triggers>
                    </Style>
                </Button.Style>
            </Button>
            <Button Command="ApplicationCommands.New"
                    FontFamily="Segoe UI Symbol"
                    Content="&#xE108;"
                    Width="30">
                <Button.Style>
                    <Style TargetType="{x:Type Button}">
                        <Setter Property="Background" Value="Transparent"/>
                        <Setter Property="Template">
                            <Setter.Value>
                                <ControlTemplate TargetType="Button">
                                    <Border Background="{TemplateBinding
                                    ↪   Background}" BorderThickness="1"
                                    ↪   Padding="5">
                                        <ContentPresenter
                                        ↪   HorizontalAlignment="Center"
                                        ↪   VerticalAlignment="Center" />
                                    </Border>
                                </ControlTemplate>
                            </Setter.Value>
                        </Setter>
                        <Style.Triggers>
                            <Trigger Property="IsMouseOver" Value="True">
                                <Setter Property="Background"
                                ↪   Value="{StaticResource
                                ↪   InsightAquaGreenBrush}" />
                            </Trigger>
                        </Style.Triggers>
                    </Style>
                </Button.Style>
            </Button>
            <Button Command="ApplicationCommands.Close"
                    FontFamily="Segoe UI Symbol"
                    Content="&#xE10A;"
                    Width="30">
                <Button.Style>
                    <Style TargetType="{x:Type Button}">
                        <Setter Property="Background" Value="Transparent"/>
                        <Setter Property="Template">
                            <Setter.Value>
                                <ControlTemplate TargetType="Button">
                                    <Border Background="{TemplateBinding
                                    ↪   Background}" BorderThickness="1"
                                    ↪   Padding="5">
                                        <ContentPresenter
                                        ↪   HorizontalAlignment="Center"
                                        ↪   VerticalAlignment="Center" />
                                    </Border>
                                </ControlTemplate>
                            </Setter.Value>
                        </Setter>
                        <Style.Triggers>
                            <Trigger Property="IsMouseOver" Value="True">
                                <Setter Property="Background"
                                ↪   Value="{StaticResource
                                ↪   InsightAquaGreenBrush}" />
                            </Trigger>
                        </Style.Triggers>
                    </Style>
                </Button.Style>
            </Button>
        </StackPanel>
        <ContentControl DockPanel.Dock="Bottom"
                        Content="{Binding CurrentViewModel}">
            <ContentControl.Resources>
                <ResourceDictionary>
                    <DataTemplate DataType="{x:Type viewmodels:LoginViewModel}">
                        <views:LoginView />
                    </DataTemplate>
                    <DataTemplate DataType="{x:Type viewmodels:HomeViewModel}">
                        <views:HomeView />
                    </DataTemplate>
                    <DataTemplate DataType="{x:Type viewmodels:InfoViewModel}">
                        <views:InfoView />
                    </DataTemplate>
                    <DataTemplate DataType="{x:Type
                    ↪   viewmodels:SensorViewModel}">
                        <views:SensorView />
                    </DataTemplate>
                    <DataTemplate DataType="{x:Type
                    ↪   viewmodels:SensorSettingsViewModel}">
                        <views:SensorSettingsView />
                    </DataTemplate>
                </ResourceDictionary>
            </ContentControl.Resources>
        </ContentControl>
    </DockPanel>
</Window>
```

Insight/Views/SensorSettingsView.xaml

```xml
<UserControl x:Class="Insight.Views.SensorSettingsView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:Insight.Views"
             xmlns:System="clr-namespace:System;assembly=mscorlib"
             xmlns:Model="clr-namespace:Insight.Models"
             xmlns:sysControls="http://schemas.microsoft.com/netfx/2009/xaml/presentation"
             mc:Ignorable="d"
             Background="WhiteSmoke"
             d:DesignHeight="450" d:DesignWidth="800">
    <UserControl.Resources>
        <sysControls:BooleanToVisibilityConverter x:Key="BooleanToVisibilityConverter" />
    </UserControl.Resources>
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="*" />
            <RowDefinition Height="2*" />
            <RowDefinition Height="6*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="6*" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="5*" />
            <ColumnDefinition Width="5*" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>
        <Label Grid.Row="1"
               Grid.Column="1"
               Style="{StaticResource TitleLabelDark}"
               Content="{Binding ViewTitle}" />
        <Button Grid.Row="1"
                Grid.Column="2"
                Width="40"
                Height="40"
                VerticalAlignment="Top"
                HorizontalAlignment="Right"
                Command="{Binding Sensor.NavigateSensorCommand}">
            <Image Source="/Assets/return.png" />
            <Button.Style>
                <Style TargetType="{x:Type Button}">
                    <Setter Property="Background" Value="Transparent"/>
                    <Setter Property="Template">
                        <Setter.Value>
                            <ControlTemplate TargetType="Button">
                                <Border Background="{TemplateBinding
                                ↪   Background}" BorderThickness="1"
                                ↪   Padding="5">
                                    <ContentPresenter
                                    ↪   HorizontalAlignment="Center"
                                    ↪   VerticalAlignment="Center" />
                                </Border>
                            </ControlTemplate>
                        </Setter.Value>
                    </Setter>
                    <Style.Triggers>
                        <Trigger Property="IsMouseOver" Value="True">
                            <Setter Property="Background"
                            ↪   Value="{StaticResource InsightAquaGreenBrush}"
                            ↪   />
                        </Trigger>
                    </Style.Triggers>
                </Style>
            </Button.Style>
        </Button>

        <StackPanel Grid.Row="2"
                    Grid.Column="1"
                    Grid.RowSpan="3"
                    Grid.ColumnSpan="2"
                    Orientation="Vertical">
            <RibbonGroup FontFamily="Arial"
                         FontSize="20">
                <RibbonCheckBox x:Name="MaxMinValuesCB"
                                Label="Enable max and min value warnings"
                                IsChecked="{Binding MaxMinReadingsEnabled}" />
                <RibbonTextBox x:Name="MaxValueTB"
                               Margin="0, 20, 0, 0"
                               Width="300"
                               Height="30"
                               Label="Upper value limit  "
                               Text="{Binding MaxReadingValue}"
                               IsEnabled="{Binding IsChecked,
                               ↪   ElementName=MaxMinValuesCB}"/>
                <RibbonTextBox x:Name="MinValueTB"
                               Margin="0, 5, 0, 0"
                               Width="300"
                               Height="30"
                               Label="Lower value limit  "
                               Text="{Binding MinReadingValue}"
                               IsEnabled="{Binding IsChecked,
                               ↪   ElementName=MaxMinValuesCB}"/>
            </RibbonGroup>
            <Button Margin="0, 100, 0, 0"
                    Width="200"
                    FontFamily="Arial"
                    FontSize="20"
                    Content="Save"
                    Command="{Binding SaveSettingsCommand}">
                <Button.Style>
                    <Style TargetType="{x:Type Button}">
                        <Setter Property="Background" Value="{StaticResource
                        ↪   InsightAquaGreenBrush}"/>
                        <Setter Property="Template">
                            <Setter.Value>
                                <ControlTemplate TargetType="Button">
                                    <Border Background="{TemplateBinding
                                    ↪   Background}" BorderThickness="1"
                                    ↪   Padding="5">
                                        <ContentPresenter
                                        ↪   HorizontalAlignment="Center"
                                        ↪   VerticalAlignment="Center" />
                                    </Border>
                                </ControlTemplate>
                            </Setter.Value>
                        </Setter>
                        <Style.Triggers>
                            <Trigger Property="IsMouseOver" Value="True">
                                <Setter Property="Background"
                                ↪   Value="{StaticResource
                                ↪   InsightAquaGreenHardBrush}" />
                            </Trigger>
                        </Style.Triggers>
                    </Style>
                </Button.Style>
            </Button>
        </StackPanel>

    </Grid>
</UserControl>
```

Insight/Views/SensorView.xaml

```
<UserControl x:Class="Insight.Views.SensorView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2
             ↪  006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:Insight.Views"
             xmlns:oxy="http://oxyplot.org/wpf"
             mc:Ignorable="d"
             Background="WhiteSmoke"
             d:DesignHeight="450" d:DesignWidth="800">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="*" />
            <RowDefinition Height="2*" />
            <RowDefinition Height="5*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="8*" />
            <RowDefinition Height="2*" />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="7*" />
            <ColumnDefinition Width="3*" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>
        <Label Grid.Row="1"
               Grid.Column="1"
               Style="{StaticResource TitleLabelDark}"
               Content="{Binding ViewTitle}" />
        <Button Grid.Row="1"
                Grid.Column="2"
                Width="40"
                Height="40"
                HorizontalAlignment="Right"
                VerticalAlignment="Top"
                Command="{Binding NavigateSensorSettingsCommand}">
            <Image Source="/Assets/settings-symbol.png" />
            <Button.Style>
                <Style TargetType="{x:Type Button}">
                    <Setter Property="Background" Value="Transparent"/>
                    <Setter Property="Template">
                        <Setter.Value>
```

```
                            <ControlTemplate TargetType="Button">
                                <Border Background="{TemplateBinding
                                ↪   Background}" BorderThickness="1"
                                ↪   Padding="5">
                                    <ContentPresenter
                                    ↪   HorizontalAlignment="Center"
                                    ↪   VerticalAlignment="Center" />
                                </Border>
                            </ControlTemplate>
                        </Setter.Value>
                    </Setter>
                    <Style.Triggers>
                        <Trigger Property="IsMouseOver" Value="True">
                            <Setter Property="Background"
                            ↪   Value="{StaticResource InsightAquaGreenBrush}"
                            ↪   />
                        </Trigger>
                    </Style.Triggers>
                </Style>
            </Button.Style>
        </Button>

        <Border Grid.Row="2"
                Grid.Column="2"
                CornerRadius="10"
                Background="White"
                Margin="40, 0, 0, 0">
            <Border.BitmapEffect>
                <DropShadowBitmapEffect Color="LightGray" Direction="270"
                ↪   ShadowDepth="1" Opacity="1" Softness="2" />
            </Border.BitmapEffect>
            <DataGrid x:Name="SensorGrid"
                      Margin="5"
                      ItemsSource="{Binding SensorReadings.Result}"
                      AutoGenerateColumns="True"
                      BorderThickness="0"
                      BorderBrush="Transparent"
                      Background="White"
                      AlternatingRowBackground="LightGray">
                <DataGrid.ColumnHeaderStyle>
                    <Style TargetType="{x:Type DataGridColumnHeader}">
                        <Setter Property="Background" Value="Transparent" />
                        <Setter Property="FontFamily" Value="Arial" />
                        <Setter Property="FontSize" Value="16" />
                        <Setter Property="Height" Value="24" />
                        <Setter Property="Width" Value="Auto" />
                        <Setter Property="Margin" Value="0, 0, 10, 0" />
```

```xml
                    </Style>
                </DataGrid.ColumnHeaderStyle>
                <DataGrid.CellStyle>
                    <Style TargetType="{x:Type DataGridCell}">
                        <Setter Property="BorderThickness" Value="0" />
                        <Setter Property="BorderBrush" Value="Transparent" />
                    </Style>
                </DataGrid.CellStyle>
            </DataGrid>
        </Border>
        <Border Grid.Row="4"
                Grid.Column="1"
                Grid.ColumnSpan="2"
                CornerRadius="10"
                Background="White">
            <Border.BitmapEffect>
                <DropShadowBitmapEffect Color="LightGray" Direction="270"
                ↪  ShadowDepth="1" Opacity="1" Softness="2" />
            </Border.BitmapEffect>
            <oxy:PlotView x:Name="SensorPlot"
                          Margin="5"
                          Model="{Binding PlotModel.Result}" />
        </Border>

        <StackPanel Orientation="Vertical"
                    Margin="0, 0, 20, 0"
                    Grid.Row="2"
                    Grid.Column="1">
            <Label Content="Station"
                    Style="{StaticResource SubSubTitleLabelOpaqueDark}" />
            <Label Content="{Binding Sensor.StationName}"
                    Style="{StaticResource SubTitleLabelDark}"
                    Margin="0, -10, 0, 0" />
            <Label Content="Reading unit  "
                   Style="{StaticResource SubSubTitleLabelOpaqueDark}"
                   Margin="0, 10, 0, 0"/>
            <Label Content="{Binding Sensor.ReadingUnit}"
                   Style="{StaticResource SubTitleLabelDark}"
                   Margin="0, -10, 0, 0"/>
            </StackPanel>
        <StackPanel Orientation="Vertical"
                    Grid.Row="2"
                    Grid.Column="1"
                    Margin="0, 0, 30, 0"
                    HorizontalAlignment="Right">
            <StackPanel Orientation="Horizontal">
                <Label Content="Notifications enabled"
```

```
                       Style="{StaticResource SubSubSubTitleLabelOpaqueDark}" />
                <Label Content="{Binding
                ↪  UpdatedInfoSensor.EnableReadingValueLimits}"
                       Style="{StaticResource SubSubTitleLabelDark}" />
            </StackPanel>
            <StackPanel Orientation="Horizontal">
                <Label Content="Upper limit"
                       Style="{StaticResource SubSubSubTitleLabelOpaqueDark}" />
                <Label Content="{Binding
                ↪  UpdatedInfoSensor.ReadingValueUpperLimit}"
                       Style="{StaticResource SubSubTitleLabelDark}" />
                <Label Content="{Binding UpdatedInfoSensor.ReadingUnit}"
                       Margin="2, 0, 0, 0"
                       Style="{StaticResource SubSubTitleLabelDark}" />
            </StackPanel>
            <StackPanel Orientation="Horizontal">
                <Label Content="Lower limit"
                       Style="{StaticResource SubSubSubTitleLabelOpaqueDark}" />
                <Label Content="{Binding
                ↪  UpdatedInfoSensor.ReadingValueLowerLimit}"
                       Style="{StaticResource SubSubTitleLabelDark}" />
                <Label Content="{Binding UpdatedInfoSensor.ReadingUnit}"
                       Margin="2, 0, 0, 0"
                       Style="{StaticResource SubSubTitleLabelDark}" />
            </StackPanel>
        </StackPanel>
        <StackPanel Orientation="Vertical"
                    Grid.Row="2"
                    Grid.Column="1"
                    HorizontalAlignment="Left"
                    VerticalAlignment="Bottom"
                    Margin="0, 20, 0, 0"
                    Visibility="Visible">
            <Label Content="Reading statistics"
                   Margin="0, 0, 0, -7"
                   Style="{StaticResource SubSubSubTitleLabelOpaqueDark}" />
            <StackPanel Orientation="Horizontal">
                <Label Content="Readings"
                       Style="{StaticResource SubSubTitleLabelOpaqueDark}" />
                <Label Content="{Binding SumReadings}"
                       Margin="0, 0, 15, 0"
                       Style="{StaticResource SubSubTitleLabelDark}" />
                <Label Content="Total time"
                       Style="{StaticResource SubSubTitleLabelOpaqueDark}" />
                <Label Content="{Binding SumReadingValues}"
                       Style="{StaticResource SubSubTitleLabelDark}" />
                <Label Content="{Binding UpdatedInfoSensor.ReadingUnit}"
```

```
                                Margin="0, 0, 15, 0"
                                Style="{StaticResource SubSubTitleLabelDark}" />
                        <Label Content="Average reading"
                                Style="{StaticResource SubSubTitleLabelOpaqueDark}" />
                        <Label Content="{Binding AverageReadingValue}"
                                Style="{StaticResource SubSubTitleLabelDark}" />
                        <Label Content="{Binding UpdatedInfoSensor.ReadingUnit}"
                                Margin="0, 0, 15, 0"
                                Style="{StaticResource SubSubTitleLabelDark}" />
                    </StackPanel>
                </StackPanel>
            </Grid>
</UserControl>
```

Insight/Views/SideBarView.xaml

```
<UserControl x:Class="Insight.Views.SideBarView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2
             ↪  006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:Insight.Views"
             mc:Ignorable="d"
             d:DesignHeight="450" >
    <Grid Background="{StaticResource InsightDarkGrayBrush}">
        <Grid.RowDefinitions>
            <RowDefinition Height="4*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="4*" />
            <RowDefinition Height="*" />
            <RowDefinition Height="2*" />
        </Grid.RowDefinitions>
        <StackPanel Grid.Row="0"
                    Orientation="Vertical">
            <Image Width="50"
                   Margin="0, 20, 0, 0"
                   Source="<SIDEBAR IMAGE>" />
            <Image Width="150"
                   Margin="20, 0, 20, 0"
                   Source="/Assets/Insight_Logo2.png" />
        </StackPanel>
        <Button x:Name="HomeButton"
                Grid.Row="1"
                Content="Home"
                Command="{Binding NavigateHomeCommand}"
                Style="{StaticResource SubtitleLabel}" />
        <Button x:Name="SetupInfoButton"
                Grid.Row="2"
                Content="Setup Info"
                Style="{StaticResource SubtitleLabel}"
                Command="{Binding NavigateInfoCommand}" />
        <Separator Grid.Row="3"
                   VerticalAlignment="Center"
                   Width="100"/>
        <ItemsControl Grid.Row="4"
                      ItemsSource="{Binding Path=FirebaseDataStore.Sensors}">
```

```xml
            <ItemsControl.ItemTemplate>
                <DataTemplate>
                    <Button x:Name="SensorButton"
                            Style="{StaticResource SubsubtitleLabel}"
                            Height="40"
                            Command="{Binding Path=NavigateSensorCommand}">
                        <TextBlock Text="{Binding Path=Name}"
                                   Width="150"
                                   TextTrimming="CharacterEllipsis" />
                    </Button>
                </DataTemplate>
            </ItemsControl.ItemTemplate>
            <ItemsControl.ItemsPanel>
                <ItemsPanelTemplate>
                    <StackPanel Orientation="Vertical" />
                </ItemsPanelTemplate>
            </ItemsControl.ItemsPanel>
        </ItemsControl>
        <Separator Grid.Row="5"
                   VerticalAlignment="Center"
                   Width="100" />
        <Button x:Name="LibraryButton"
                Grid.Row="6"
                VerticalAlignment="Top"
                Style="{StaticResource SubtitleLabel}"
                Command="{Binding OpenHttpLinkCommand}">
            <StackPanel Orientation="Horizontal">
                <TextBlock Text="Code  " />
                <Image Height="20"
                       Width="20"
                       Source="/Assets/external-link-symbol.png" />
            </StackPanel>
        </Button>

    </Grid>
</UserControl>
```

Insight/App.xaml

```xml
<Application x:Class="Insight.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:local="clr-namespace:Insight"
             StartupUri="Views/MainWindow.xaml">
    <Application.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary Source="pack://application:,,,/ToastNotific⌋
                ↪   ations.Messages;component/Themes/Default.xaml"
                ↪   />
            </ResourceDictionary.MergedDictionaries>
            <Style x:Key="FuiTextBlockTitleStyle" TargetType="TextBlock">
                <Setter Property="Foreground" Value="Red" />
            </Style>
            <Style x:Key="TitleLabel" TargetType="Label">
                <Setter Property="FontFamily" Value="Arial" />
                <Setter Property="Foreground" Value="White" />
                <Setter Property="FontSize" Value="28" />
            </Style>
            <Style x:Key="TitleLabelDark" TargetType="Label">
                <Setter Property="FontFamily" Value="Arial" />
                <Setter Property="Foreground" Value="Black" />
                <Setter Property="FontSize" Value="28" />
                <Setter Property="FontWeight" Value="Bold" />
            </Style>
            <Style x:Key="SubTitleLabelDark" TargetType="Label">
                <Setter Property="FontFamily" Value="Arial" />
                <Setter Property="Foreground" Value="Black" />
                <Setter Property="FontSize" Value="22" />
                <Setter Property="FontWeight" Value="Bold" />
            </Style>
            <Style x:Key="SubTitleLabelOpaqueDark" TargetType="Label">
                <Setter Property="FontFamily" Value="Arial" />
                <Setter Property="Foreground" Value="Black" />
                <Setter Property="FontSize" Value="22" />
                <Setter Property="FontWeight" Value="Bold" />
                <Setter Property="Opacity" Value="0.6" />
            </Style>
            <Style x:Key="SubSubTitleLabelDark" TargetType="Label">
                <Setter Property="FontFamily" Value="Arial" />
                <Setter Property="Foreground" Value="Black" />
                <Setter Property="FontSize" Value="16" />
                <Setter Property="FontWeight" Value="Bold" />
            </Style>
            <Style x:Key="SubSubTitleLabelOpaqueDark" TargetType="Label">
                <Setter Property="FontFamily" Value="Arial" />
                <Setter Property="Foreground" Value="Black" />
                <Setter Property="FontSize" Value="16" />
                <Setter Property="FontWeight" Value="Bold" />
                <Setter Property="Opacity" Value="0.6" />
            </Style>
            <Style x:Key="SubSubSubTitleLabelOpaqueDark" TargetType="Label">
                <Setter Property="FontFamily" Value="Arial" />
                <Setter Property="Foreground" Value="Black" />
                <Setter Property="FontSize" Value="14" />
                <Setter Property="FontWeight" Value="Bold" />
                <Setter Property="Opacity" Value="0.6" />
            </Style>
            <Style x:Key="ContentLabelDark" TargetType="Label">
                <Setter Property="FontFamily" Value="Arial" />
                <Setter Property="Foreground" Value="Black" />
                <Setter Property="FontSize" Value="12" />
                <Setter Property="FontWeight" Value="Regular" />
            </Style>
            <Style x:Key="SubtitleLabel" TargetType="Button">
                <Setter Property="FontFamily" Value="Arial" />
                <Setter Property="Foreground" Value="White" />
                <Setter Property="FontSize" Value="24" />
                <Setter Property="FontWeight" Value="Bold" />
                <Setter Property="BorderThickness" Value="0" />
                <Setter Property="Background" Value="Transparent" />
                <Setter Property="Template">
                    <Setter.Value>
                        <ControlTemplate TargetType="Button">
                            <Border Background="{TemplateBinding Background}"
                            ↪  BorderThickness="1" Padding="5">
                                <ContentPresenter HorizontalAlignment="Center"
                                ↪  VerticalAlignment="Center" />
                            </Border>
                        </ControlTemplate>
                    </Setter.Value>
                </Setter>
                <Style.Triggers>
                    <Trigger Property="IsMouseOver" Value="True">
                        <Setter Property="Background" Value="Black" />
                    </Trigger>
                </Style.Triggers>
            </Style>
            <Style x:Key="SubsubtitleLabel" TargetType="Button">
```

```
                <Setter Property="FontFamily" Value="Arial" />
                <Setter Property="Foreground" Value="White" />
                <Setter Property="FontSize" Value="16" />
                <Setter Property="FontWeight" Value="Regular" />
                <Setter Property="BorderThickness" Value="0" />
                <Setter Property="Background" Value="Transparent" />
                <Setter Property="Template">
                    <Setter.Value>
                        <ControlTemplate TargetType="Button">
                            <Border Background="{TemplateBinding Background}"
                            ↪   BorderThickness="1" Padding="5">
                                <ContentPresenter HorizontalAlignment="Center"
                                ↪   VerticalAlignment="Center" />
                            </Border>
                        </ControlTemplate>
                    </Setter.Value>
                </Setter>
                <Style.Triggers>
                    <Trigger Property="IsMouseOver" Value="True">
                        <Setter Property="Background" Value="Black" />
                    </Trigger>
                </Style.Triggers>
            </Style>
            <Color x:Key="InsightBrightPink">#CF3054</Color>
            <Color x:Key="InsightAquaGreen">#30CFAB</Color>
            <Color x:Key="InsightDarkGray">#292929</Color>
            <Color x:Key="InsightBrightYellow">#E1BF1E</Color>
            <SolidColorBrush x:Key="InsightDarkGrayBrush"
            ↪   Color="{StaticResource InsightDarkGray}" />
            <SolidColorBrush x:Key="InsightBrightPinkBrush"
            ↪   Color="{StaticResource InsightBrightPink}" Opacity="0.3" />
            <SolidColorBrush x:Key="InsightAquaGreenHardBrush"
            ↪   Color="{StaticResource InsightAquaGreen}" />
            <SolidColorBrush x:Key="InsightAquaGreenBrush"
            ↪   Color="{StaticResource InsightAquaGreen}" Opacity="0.3" />
            <SolidColorBrush x:Key="InsightBrightYellowBrush"
            ↪   Color="{StaticResource InsightBrightYellow}" Opacity="0.5" />
        </ResourceDictionary>
    </Application.Resources>
</Application>
```