DESIGNING AN INTERFACE FOR A DIGITAL TWIN TO OPTIMIZE DATA TRANSFER AND VISUAL CUES

Supervisor: Job Zwiers
Critical Observer: Maaike Slot

Sil Tijssen

17-03-2021

Abstract

Digital Twinning is the act of representing a physical asset in a virtual way. Due to the rise of Industry 4.0, the emergence of Smart Factories is getting started. These new factories focus on flexibility, adaptability and making products to order. The project researched in this thesis focusses on designing and creating a user and technical interface in order to solve the possibility of information overload that comes with a large number of virtual robots. The link created between a TurtleBot and an interface created in Unity facilitated the transmittance of data from a Ros-based robot to the user interface. The link as such consisted of a TurtleBot connected to a pc with Ubuntu via Roscore. On this pc a connection was established using an MQTT broker to connect to an instance in Unity. Although the interface did not fulfil all the design requirements, it still provided insight in terms of future work and it laid down a foundation that can be used in future research. The creation of the user interface and the technical interface, or the connection, provide an opportunity to establish interfaces in this field.

Acknowledgements

I would like to thank my supervisor, Job Zwiers, and my critical observer, Maaike Slot, who helped me a lot with this project, providing me guidance and support, as well as feedback.

I would like to thank my parents, for supporting and pushing me through this project. I would also like to thank my girlfriend, who stayed with me many times and provided advise and support.

Finally I would like to thank Robert Breugelmans, a student working on a very similar topic, who provided me with advise on how to create the connection between the TurtleBot and Unity.

Table of Contents

Chapter 1: Introduction

Nowadays, manufacturing facilities use foremen and operators to steer the machines located within these facilities [1]. This means that the facility is monitored and adjusted by them, tailored to their needs at a specific moment. This allows them to run the facility in a manner that it can make products to order (MTO) [2]. It is not unwise to think that with technology ever developing, a new age could potentially be upon humanity quite soon: the age of the new industry.

In a facility, a Cyber-Physical System can be used. Cyber-physical systems or CPS are systems where there is a deep connection between digital programs and the physical world and what is happening within it [3]. This connection both provides and extracts data via a common core, meaning a central location that reads, writes and instructs everything connected to it. In a production environment, this can be very useful in terms of getting intricate data on several aspects in a manufacturing facility. Over the last few years, a new sort of industry has gained attention and has started to develop into the next phase, Industry 4.0 [3]. In the past, before this particular type of industry, regulating such a facility would be done via operators and foremen [1]. These people would tend to the facility, diverting production and solving problems on the spot.

Recently however, the complexity of such environments has been increasing rapidly, making it difficult for any employee to get a good overview of all the activities and states of all the machines and devices located within the facility that they provide their labour [1]. As such, another technology needs to be introduced to this system to provide a greater overview and better data communication to allow for more intricate and complex manufacturing facilities.

Digital Twinning is the act of integrating a physical world with its virtual representation [4]. Digital Twinning allows one to see a virtual representation of a certain asset, for example a robot. There are many applications for such a technology, but one particularly interesting is the industry sector. Digital Twins can provide a system with information about performance optimization and analysis. Someone who understands a machine's internal workings gives them great control of this machine, potentially from a distance.

Even though digital twinning has been started to be defined around 15 years ago, only the past few years has it gained ground and interest in the eyes of academics and the industry sector. According to a certain paper called 'Shaping the digital twin for design and production engineering', different digital twin perceptions can be seen in this sector, specifically with PLM (Product Lifecycle Management) [5] software [6].

What digital twinning means for the new phase of industry, Industry 4.0, is that it could provide a new way of relaying control and information to give an overview of a flock of machines. In this case, more like an overview of digital twins. Each machine could be assigned a digital twin, which then remotely accesses and distributes data about others and itself.

This introduces a potential problem. With a major supply of information and tons being accessed and distributed at the same time, it can be an immense issue to provide the potential foremen and overseers with the right information at the right time [1]. More information does not always mean more insight, and certain use of visual and tactile cues in accordance with task performance can yield better results [7]. Without proper usage and use of this information and these cues, it can be hard to get a grip on what is needed at a certain time. One way of conveying and displaying information is via an interface of some sort.

Hence, the following research question is proposed to solve this intricate overload:

- How can both a user interface and a communication infrastructure be designed and built for a digital twin to display information required for optimally performing and interacting with several different digital twins for an operator?

As well as the following sub questions:

- How and what kind of information should be displayed for a foreman operating machines in a manufacturing facility with autonomous machines?
- How can information be displayed in a way that optimizes interaction, performance, emergencies, and criticality at hand?
- What interactions does an operator require with an industrial based robot and how detailed should these interactions be?
- What design needs to be created in order to create a proper technical interface to connect to a user interface?

The research was conducted by means of the integration of a scalable ROS-based robot called a TurtleBot, with the concepts discussed here. One of these concepts that was integrated was the digital twin, and with it a specific interface, for example in the form of a digital interface, was developed and tested to the extent of the questions mentioned above. After performing the research and the technical creation of the prototype, the digital twin and its accompanying interface, several recommendations were made at the end to ensure that any potential follow-up research takes note from the results found in this thesis.

This thesis is part of a multiple of research projects dedicated to researching specific topics in the subjects of smart factories and the new age of industry. Using an area on the University of Twente called the Testbed, multiple researches take their clients and assignments from this area. Together, they will form researchers that will carry the understanding and development of smart factories to the next steps required to make it a widespread possibility.

The following chapters will describe some background research including cases already using automated guided vehicles, followed by a chapter of the methodology used in this research, after which a chapter will describe the ideation for the prototype. This will then be followed by a chapter detailing the technical specification after which the realisation will be described. Finally, the evaluation and the conclusion will finish the research.

Chapter 2: State-Of-The-Art Research

In order to gain some extra knowledge on these topics, some background research will have to be done. These topics in specific were researched because the researcher found it handy to get some more information about history but also to gather some information about technologies that are in use and some examples of where it is found.

2.1: The Industrial Revolution

According to H. Lasi and others, the four industrial revolutions can be classified as what they describe as ‘paradigm shifts’. The first industrial revolution was the ‘field of mechanization’ [8], the second revolution the intensive use of energy, and the third revolution the widespread digitalization. Now that we are approaching the fourth revolution, it is good to gain an understanding of the past, so that it be reflected upon for the future. Let us start at the beginning of factories and facilities and describe the first industrial revolution. As mentioned above, the first revolution was based on mechanization. What this means in a nutshell is that the process of manufacturing a product was now based on machines instead of manpower. This allowed factories to develop and use a new type of production that was only partly dependant on workers, as opposed to the earlier stages of manufacturing.

2.2: TurtleBot and AGVs

Automated Guided Vehicles (AGVs) are vehicles without drivers that used for transport of materials [9]. Plenty of AGVs are used in the industrial sector. AGVs are probably best used for surroundings with consistent transport lines, according to an author stated in the survey. The TurtleBot specifically is a low-cost open source robot that allows one to program it in a specific way (https://www.turtlebot.com/). It is a device with the possibility of adding in different sensors and motors in order to change its functionality. Its value comes not only from its user-friendly setup, but its ability to run ROS. It has plenty of uses in a home-service environment (https://www.turtlebot.com/about/), and is able to use SLAM algorithms to map and drive around a room. The Turtlebot3 uses a Raspberry Pi 3, allowing for easy manipulation and control via a Linux environment. The TurtleBot allows for research and prototyping ,as well as product development (https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/). The structure of the TurtleBot allows it to be expandible without negating its ability to function, all while at a lower cost. The TurtleBot can be changed and customized in an accessible way, allowing a user to tailor it for a specific task they may have in mind.

2.3: ROS

The TurtleBot runs on ROS on the Raspberry Pi. ROS (https://www.ros.org/about-ros/), also known as Robot Operating System, is a framework that allows one to write robot software, using a collection of several tools and libraries to create behaviour for robots in a simple fashion. Looking at the history of ROS (https://www.ros.org/history/), it is visible that ROS was developed for multiple robots and that it was developed at multiple facilities. ROS is an opensource project.

2.4: Digital Twinning

A digital twin, can be described, as one source states, as the following: “A tool that can potentially account for the whole system of a product or service. It keeps track of all the information about a system you need and from that, information assists in the decision-making process” (p. 541) [1]. This means that a digital twin does not necessarily have to be defined by being an exact copy of a physical object, but keeping track of all required information to form a proper representation. Another way to define digital twinning would be "a comprehensive digital representation of an individual product” (p. 64) [10]. Additionally, here is stated that that it includes models and data of the physical object, containing its properties, condition and behaviour. As such, it is clear that digital twins can have a certain range of definition, and that means that for the purposes in the project, a digital twin should be defined in its own terms as well, in order to avoid confusion.

The definition that will be used within this research will be the following: The art of digital twinning is taking a physical object, and representing it by means of a digital version that matches the object in its aspects and data. A digital twin, by definition is a twin of a physical object, represented digitally.

2.5: AGVs as an existing technology in companies

Looking at a different part of AGVs, how they are used in practice, can give an additional fraction of insight in to the inner workings of technology such as this used in facilities and companies. One such example is this one: https://www.afklcargo.com/NL/en/common/about_us/our_hubs.jsp.

This company, Air France KLM Martinair Cargo, is a airplane cargo business. They have 2 hubs, one at Paris Charles de Gaulle and Amsterdam Schiphol, with over 2 billion in combined revenue and 1.2 million tons of cargo transported via air. They have plenty of available aircraft to ship all sorts of goods. However, the point of interest is G1XL, a 1.4 million ton capacity airfreight hub. Equipped with high tech, they use 32 automated guided vehicles to transport goods across the 120,000 square-metered hub. The AGVs are connected to Pélican, a management system for the cargo (https://www.afklcargo.com/NL/en/common/about_us/cdg.jsp). This system and the AGVs together handle transporting almost all of the aircraft transport within the hub (https://cargoweb.airfrance.fr/FR/common/common/pdf/brochure_G1XLEn.pdf).

Another example is Vanderlande (https://www.vanderlande.com/about-vanderlande/innovation/). Vanderlande is also a company specialized in moving goods, but more based in the sense of automating logistic processes. This means they focus more on optimizing logistics for companies such as Air France KLM Martinair Cargo. They developed FLEET, a solution that uses automated vehicles to replace traditional transport tools, such as conveyer belts. On top of that, they are developing a new generation scalable solutions like AIRPICK, that integrates all sorts of systems. Going back to FLEET, this solution uses intelligent software to divide cargo into bags or batches of bags. They use AGVs, in this case called ‘runners’ used for transporting cargo. FLEET enables a user of the program to easily add runners or alter routes. On top of that, their solution is sustainable, allowing users to re-use runners for different purposes, as well as being able to recycle them if necessary.

Last, but not least, we have Körber (https://www.koerber-supplychain.com/), a company that offers clients the opportunity to provide them with supply chain technology and provides them with the expertise to set it up for them. They also provide the client with Autonomous Mobile Robots, or AMRs, to improve the transport of all sorts of locations and tasks with it. Here they describe an important difference between AGVs and AMRs, namely the following definition: “AMRs operate from a digital map of their environment, and require little or no infrastructure modification.” They claim AMR to be more affordable and easy to scale compared with traditional automation, and they mention key benefits to be flexibility, easy scalability, and smooth integration.

These 3 cases and usages of technology show us a couple things. First, AGVs are already widely used and often have some sort of framework set up that lets them automate (part of) their work. This frameworks then allows operators controlling and observing the setup to more easily get the AGVs or AMRs to the job they want, presumably. Secondly, AGVs and AMRs often come into greater numbers. There do not seem to be a small numbers of AGVs among these companies, and as such, the robots driving around often have to take into account quite a number of other instances of these robots when manipulating their position.

2.6: Information and visual cues

Information is an interesting topic. In one way, one can define seeking information as an interaction [11]. In this way, several methods of seeking information can be defined as strategies, and their interactions can be defined to be between the user and anything else in terms of a certain system. Applying this to the purpose of the research here, will result in one way of designing some guidelines in terms of interface creation. Looking at how people search and find information might give some perspective on how to design an interface designed with providing information. On another note, one article created a visualisation of a digital twin within Augmented Reality (AR) on a HoloLens [12]. While an initial thought for a design of an interface may have been a screen, a device such as a HoloLens proves in this article to be a good contender for such an interface. The HoloLens provided the operator with the ability to monitor and operate the machine tool. However, it also allowed the user to interact with the digital twin. This device allowed the interaction to be intuitive and consistent, which brough efficiency during operating time.

Whatever the means, how the information is brough is crucial. A survey conducted by the U.S. Army Research Laboratory, using tactile and visual cues and a starting condition [7]. They made 3 types of comparisons within their review. In short, they hade tactile cues to a starting condition, a comparison between tactile and visual cues that show the same information, and the comparison of visual and tactile cues in some combination that also show the same information. Important information shown within the article explains that tactile cues enhance a starting task or visual cues and enhance task performance. Tactile cues replacing visual cues, effects depend on the cue complexity. They can still be effective, but tactile direction cues do not improve the performance when they replace visual direction cues.

In short, information, when presented properly via a good combination of certain cues, can absolutely mean a difference in performance of certain tasks. Certain methods and devices might bring it a different and a potentially more intuitive way, but the main priority should the presentation of the information itself.

2.7 Conclusion

In this chapter, a lot has been discussed. Maybe most notable are the technology, information presentation, and the interview. From all this data, the following can be concluded. To program a digital twin, information bringing has to be kept in mind at all times in order to optimize task performance that an operator may have operating such a digital twin. It is important that the digital twin gets designed in such a way that it optimizes flow, and does not disturb the surrounding environment in case of an unfavourable position. It is important to look at other companies and technologies in order to not just gain inspiration, but to see what matters in those companies in regards to design with AGVs and why. It is crucial the TurtleBot gets built and designed in such a way that it indeed simulates some form of AGV environment usually found in facilities such as an airport. Then, designing a digital twin for this TurtleBot complying with the parameters above should be held in high regard. All in all, as long as the data from this background research is kept into high regard in the design process, such as the information display, the digital twin as a result from this research should be able to provide a good addition to research in the field of AGvs and smart factories.

Chapter 3: Methods and Techniques

In order to take a look at the initial design methods, the Creative Technology Design Process shall be analysed. As described in this process[13], there are 4 main sections that are present, namely the ideation, the specification, the realisation and the evaluation. The design process tries to merge design processes from two other fields, the one of typical engineering and the one of design typically revolving around users and their needs. In Figure 1, this process in the form of an image from the article can be seen in a bit more detail.

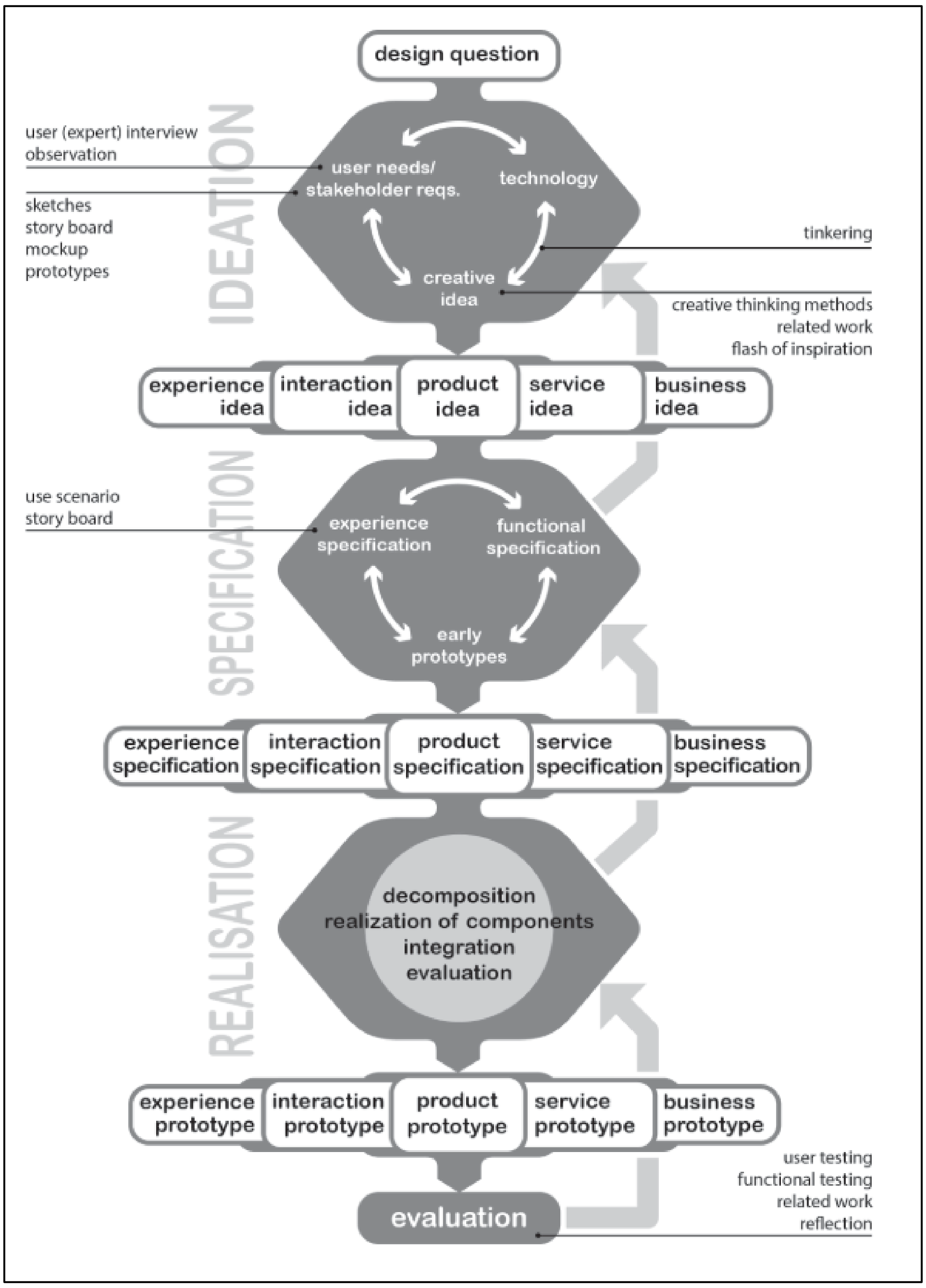


Figure 1: Graph from 'A design process for Creative Technology', presenting a possible graph of a process to be used in the bachelor.

Chapter 4: Ideation

4.1: Requirement Analysis

4.1.1: Interview with an operator within a manufacturing facility

An interview with an operator that has experience within a manufacturing facility was conducted in order to get a better understanding of someone who works in the industry using AGVs and industrial robots. A person working in an industrial environment working with AGVs will be defined here as an operator. An operator's activities in such an environment can be linked to a few things. First, if the environment is a production environment, an operator could adjust or control a machine, or control the transport, or input rather, of material of such a machine. It could potentially be along a conveyer belt or transport system of some sorts. However, automatised production often entails more observation and indirect control in contrast to actively controlling robots on a lower level in machine terms. Their main objective is to keep the flow, as it is called, going in a manufacturing facility. This means that the production efficiency is always optimal, that the factory is always going at its best potential at that moment, so that products and transport are the most optimized in terms of time and money. Which, as one can imagine, is very important in manufacturing and business in general. The AVG's task is to (mostly) drive around and perform transport tasks. It depends of course on the facility itself and its needs.

An operator in this regard spends time with higherups and planners to outline future events for the facility. This means that any available information has to do with the flow and its optimization. Efficiency is key to such a process. Resource management of both people and material is crucial to the success of the optimization of the flow. If an AGV is standing still for some reason, this should be immediately highlighted and brought to attention of the operator. This is because a standstill of an AGV can cause a buffer effect within the facility, creating a delay. If this is at the front of the facility, this is less of a problem, since the buffer cannot affect the rest of facility in the same way a buffer can affect it when it is all the way at the back of the facility. This means that production time can increase, which is bad for the factory when both time and money are of the essence. Taking this into account, the level of anatomy is vital to making sure that the facility does not run into problems. The level of anatomy refers to a number of tasks needing to be done, and AGVs currently doing that task. In a facility, if an AGV happens to stop working, a system should be responsible for handling task assignment in the way mentioned above, be it via human interaction or only based on computers. This is based upon the tasks of the operator, since their assignment and handling of the anatomy of the facility determines the system used to assign tasks to the AGVs in the facility. If the tasks are handled automatically, then the operator does not do much direct interaction with the AGVs, and vice versa. However, the task of the operator is and remains the fact that the flow must be optimized, and at the very least approximated in terms of available data. As an operator, you should be able to view information about starting and stopping tasks, and you should be able to get into control, steer and change the AGVs and the facility in order to fix the problems that need to be fixed.

Say that an AGV gets into a difficult spot by blocking other AGVs or human personnel, or that it gets itself into a spot it cannot leave, what happens then? First, the AGV should be moved into a spot where it cannot bother other machines or personnel. More importantly however, the AGV should give out a signal, both visually and to the operator in some way, that it is stuck. In all cases, damage minimization should be held in high regard. If the AGV is able to find its own way out, this also means that the surroundings should be kept in close observation. It is important to the facility and the people working there (as well as all the robots), that the AGVs do not block paths from each other or personnel. On top of that, such a vehicle at any point should not create more danger or disturb others. If the AGV is in a position, but its position is blocked by some other workstation, it is detrimental to the success of the facility that it does not disturb this workstation or in some other way cause damage, either to itself or the workstation or its parts. As such, it is of importance that these vehicles have some way to determine their possible disturbance and damage control, either by control and repositioning via an operator of some kind, or by them somehow understanding their environment in a more detailed way. This means that there are certain limits the AGV should keep itself to. Trying can be allowed, as long as the limits are kept in check. Safety and prevention of disturbance of the production are key to the success to the solution of such a problem.

This may all depend on the size of the facility or company hosting the production. As such, getting stuck should be avoided as much as possible, not just for inconvenience. As an operator in small facility, one often will need to walk throughout it in order to observe and control the situation. Not just for the AGVs in the facility, but the personnel performing their tasks there as well. With larger facilities, this gets more difficult considering the size and scale of the factory. If a facility works properly 100%, then that is good for the facility. With less that is annoying to say the least. If the technology within the facility fails more than once, and this happens often, the acceptance for this technology will decrease. If people see the technology fail more and more often, the trust within the technology will become brittle. However, if the opposite is true, people instead will gain trust within the technology. The reason that this is relevant is because if people trust in the technology, they will be more inclined to work with it. The TurtleBot brings flexibility. Speaking of trust, information can change the way an operator interacts with an AGV. It depends on production, but priority is often an indicator. If something goes wrong, or if there is an emergency, changing AGVs based on priority is one way that information changes what happens at a given time. Planning influences a facility, so information that influences a planning influences the facility as well. The flow has an influence as well. In a sense all information can have an influence on a decision to configure an AGV. The decisions one makes as an operator are based on time, capacity, planning and priority. Everything is linked to money. Money, time and growth are all linked.

From this interview, a few conclusions can be made clear: First, the operators most of the time do not directly control or manipulate the robots they work with. Often, they instead provide a way for the AGVs to resume or change their task. This can be done more directly, but manipulation of the robot itself often is not done in their job. Secondly, trust should be taken into account. This sounds odd, and very visible of course, but the meaning of this is that the technology provided to both the operators and the people on the work floor should do its job in such a regard that the technology will not viewed as distrustful, and instead should make the employees trust in order to build morale. Third, but certainly not least, self-control and awareness. The AGV, when stuck in a certain spot or in some other kind of trouble, should be able to measure its own necessity in moving compared to its surroundings. What that means is that an AGV cannot just ride into some sort of other working station if that is the only way to solve one of its problems. It should be able to measure its ability to ride at a certain location with the cost of disturbing surrounding areas. Of course, when doing all this, it also should give out some sort of signal, both to the server that it connects to, and perhaps visually, that it is indeed stuck in one way or another. It could perhaps in this way also ask for permission for certain routes that normally would be considered a no-go. Fourth, The AGV should be designed in a way that optimizes flow in the facility. What this means is that the production quality, or logistics measurement, or any kind of optimization of product and transport or some other service should be pushes to its maximum potentiality. This means that the AGVs have to be built and programmed in such a way that it allows them to optimize their part of the flow, in their sense of self, and in the sense of the facility as a whole. This also means they also should be able to estimate their own impact on the facility if they get stuck.

4.1.2: Introduction to the analysis

From the state of the art research and the interview with the operator, a preliminary requirement analysis can be created. From the conclusions made in the background research, an initial position is given for some starting guidelines. But this does not give enough information to give an initial assumption. A closer look has to be taken into the perspectives of the users. But before this can happen, an initial state must be configured to test it with the users and their perspectives. This will done, firstly by some initial ideas, and secondly, by forming a use case scenario based on the background research to form a more or less accurate scenario of an operator and AGVs within some sort of facility. Based on that, some final ideas can be formed

4.1.3: Initial Ideas

In order to get some initial ideas for the prototype, some ideation methods can be used. Several exists, such as generating 50 ideas and mind mapping. First, lets start with generating 50 ideas, and pick the most promising. Appendix 3 contains a list of the generated ideas using this method. Due to the nature of this idea, some ideas may be good, some ideas may be bad. The 10 most promising ideas, also marked within the appendix were:

- 3D-Model representation on a 2D-screen
- Small 3D representations on a grid of AGVs
- Status bars and panels displayed over everything else
- Haptic feedback on body or device that can be worn or carried
- Visual light/sound on AGV in case of emergency
- Task assignment system that uses nodes to indicate what is being done at what time by what AGV
- Overview of the facility map with AGVs and their location as well as small panels of data
- Multi-camera view
- Status update log that contains all status updates like a chatroom
- AGVs having colour and shape codes depending on what they are doing

These ideas were based on potential appeal to the operator based on the background information located within chapter 2. These ideas, either together, or separately could form an interest design within the specifications chapter. These ideas can now form the basis of the filter that is the user test. What this means is that the ideas will be evaluated by a user of some sort to not only test their enthousiasm, but more importantly, to test the viability and user experience of the prototype. Before we can get into that, a use case scenario must be created based on background research to discover how exactly these ideas should be integrated within the workflow of a facility, an operator, and an AGV.

4.1.4: Use Case Scenario

In order to create a proper environment for the ideas to test in, a use case scenario will have to be created. Two things are required, a persona, and an environment. For our purposes, our environment will be a facility of some sorts, set in terms of production. The persona is a bit more complicate however. Both will be described like so:

Gary is a 33 year old operator within Marco Industries, a company that focusses on producing specific machine parts. Gary has been taught in the arts of developing and maintaining logistic technology, and focusses his thoughts on optimizing things. In the facility that he works in, plenty of AGVs drive around and do their tasks, and Gary focusses on optimizing the production within the facility. Gary has plenty of experience with computers, and moderate experience with AGVs and their interactive programs, but does not work at this facility for long. Gary does this by observing and adjusting the production and other processes within facility, together with planners and higher-ups of the facility. Gary has received word from one of the planners that one specific set of machine parts has a fault produced in them, so he receives an assignment to temporarily adjust the facility to make those parts earlier than everything else. Gary walks to his computer and uses the system integrated with the facility to create a new number of tasks that need to be done. He assigns remote machines and AGVs new temporary tasks that they will need to do for a relative short duration. He does this by creating a new assignment or production request, and assigns a few AGVs with high-priority transport requests. While this production is going on, he quickly checks if none of the machines are standing by, in other words, are not doing anything besides their jobs. It turns out that one AGV had a problem handling the request and needs to be reactivated. Luckily, Gary can easily select the AGV from a particularly handy menu and chooses to restart the AGV and to then reassign its tasks. Now that the facility is handling the urgent production request, Gary sits back for a bit, closely keeping an eye on all the data and panels in front of him. All seems well, until one of the AGVs suddenly gives off a loud signal as well as a red flashing light on the screen. It turns out that it cannot leave a certain area, and must disturb a local workstation, a location where machines or people do their work, if it needs to get out of the unfavourable location. Gary quickly talks this over with the planners, and decides that the best course of action is to temporarily disturb the station to save time, in order to prevent long waiting times as well as a disturbance of the flow of the facility by the AGV not doing its job. Gary assigns another AGV to temporarily take over its tasks while he commands the AGV and the nearby station to quickly allow the AGV to drive out and reorient itself. After that has happened, the workstation is enabled again, the AGV can continue its tasks and the temporary task assigned AGV continues its own tasks. Luckily, the program that Gary used was intuitive enough to allow him to do this with relative ease, even though he has a lot of experience working at a new facility. Now that the temporary production has been completed, Gary assigns the AGVs their old tasks and quickly gets some coffee before another things happens.

This scenario shows a few things. First, the system used must be intuitive and quickly to get used to, as well as provide the operator with several tools to not just check in with AGVs, but allow them to easily see their statuses and potential error codes. It turns out that location is important, but mostly when an AGV is in peril. The system should be plenty visual and sound based, but sounds mostly may come in handy when there is an urgency of some sort. 3D-models do not seem to be required. Data is absolutely crucial (and by extension its display), as is assigning different things to AGVs.

4.1.5: The MoSCoW Method

The MoSCoW method [15] is a method used to describe certain features of a product and how important they are. They are classified in Must-Haves, Should-Haves, Could-Haves and Will-Not-Haves-Right-Now or Won't-Haves. The priorities and features discussed within this chapter can too be classified using these terms using the table below.

Must-Haves	- Proper display of information, possibly via visuals, sounds, and haptic feedback. - Indirect control over AGVs within a facility, allowing one to reassign tasks to them. - Status control and error messages of the different AGVs.
Should-Haves	- A way for the operator to know where exactly an AGV is located. - An intuitive and easy to use interface to properly interact with all the AGVs and other remote machines.
Could-Haves	- 3D representations of the AGVs, perhaps in some form of grid. - Facility maps as a layout for the locations for the AGVs. - A good way to distinguish AGVs from each other. - A system that automatically assigns tasks to AGVs. - Multi-camera view
Won't-Haves	-

Figure 2: Table containing the different priorities in classifications in terms of the MoSCoW method.

4.1.6: Conclusion

In this chapter, various methods of ideation were discussed and used, and a preliminary set of requirements has been synthesised from background research. Using the '50 ideas' and the 'MoSCoW' method, potential ideas have been established to give a first form to the requirements. In the next chapter, these ideas will be evaluated using a structured interview, and will give a definite design in terms of verified requirements given by the target group.

Chapter 5: Specification

5.1: Introduction

In order to provide more detail both as part of the design and as part of its creation, this chapter will give a more technical insight and description of how to create the prototype discussed so far. By ways of technical descriptions, the prototype will gain its first form and first version of the development cycle.

The first part of this technical description will be the design choices being realized into more concrete features of the interface. The second part will be the features laid out in more detail in terms of programming, hardware and software. To start, the several features discussed in chapter 4 in table 1 shall be transformed into more elaborate features.

5.2: Information display

First, the proper display of information. It is possible to do this via several mediums, such as visual, audible, and haptic feedback. As abstract as this statement is, it does allow one to narrow the field that these concepts entail. Visuals can be defined as anything that is reflected light being caught by our eyes, but of course, this more physical definition of the word does not help in terms of design. As such, a more design-friendly definition is required. It can be defined in sense of efficiency and clarity even. It could be defined in terms of how good an object on, for example, a screen is at conveying certain information. This could be a thought, an emotion, or some other information. As an example, let us look at two things in specific. Colours, and shapes. Colours often can convey information. Looking at applications that provide users with feedback, such as Duolingo, it can be seen that a correct answer is rewarded with a green colour, and a wrong answer with a red colour. This connection of colour to being right or wrong is one such application of colour. The app, however, not only uses colour, it also uses other visuals. One such example that in more recent versions of the app, Duolingo has started using animated characters to display feedback in terms of the answer. Characters will be in an idle state, and upon completion of the question, a certain other animation will be performed. A positive one, for example a cheer, can be used to give feedback to a correct answer while a sad animation could be used as the opposite. These visual designs together with the colour convey either of two thoughts: "You were correct", or "You were wrong". In this way, it is not just clear, but very quick at doing so, as a user can instantly see what happened.

Continuing with conveying information, it can also be done via audio. Audio is important, because it does not necessarily require a person to focus on one specific spot with their eyes. Instead, it can be used as another way to convey information to a person that may be attending some other activities, allowing that person to be aware of a certain information, such as an alarm of some kind, say a car alarm. The alarm alarms the person, surprisingly, and informs that something is going on with their car. It can also serve another purpose, for example scaring of anyone who may be messing with the car, but in context of this chapter serves as one medium for information. In this case, ears and air. Audio lacks the visual advantages by lacking, well, visuals, but makes up for this for its ability to convey almost anything completely via the air. Sure, images might be a bit difficult, but spoken sentences and certain alarms, as well as notifications can be extremely effective. One disadvantage though is that even though audio has some major advantages and strong points, if someone is already listening to something else, they may not be able to focus on it, thus audio losing its strength.

Third, but certainly not least, is haptic feedback. Haptic refers to motion feedback, such as feeling vibration once there is a notification on one's phone. Vibration shares some of the advantages with audio, such that a person is not required to focus on it to receive information. As such, it could be used in parallel with video and audio.

Taking all these different aspects in mind, it is clear that they need to accommodate the requirements listed in table 2 in a specific way. It requires knowledge of design, unsurprisingly, and requires the creator of the interface to implement specific features. Hence, the following specifications for this part are proposed:

- Warning and emergency events for the user should be labelled and created with bright colour associated with negative effects, such as red and orange. These should also be able to be labelled with a severity, such as orange for critical situations and red for the most emergent emergency situations. Similarly, positive situations should be labelled with a green colour, and perhaps yellow in case of certain warnings or errors, but not as crucial as an emergency.
- Buttons within the UI should be given certain symbols often associated with their functions. Such as a cog for settings, and a door for exiting the app. These symbols and their usage should be extended to individual inspecting panels related to the AGVs connected to the interface.
- Warnings and emergencies, as well confirmations should have, short but indicative sounds played when the events are triggered, as to both notify the user when something is happening or has happened while the interface is out of view, as well to establish a connection between the sound and a certain emotion or thought from the user.
- Any input from the user should have some sort of feedback. This can be as simple as changing the icon of the mouse, or could also have additions of sound and colour. Error sounds when something invalid has been clicked and satisfying sounds when something correct has been clicked are examples of this.

5.3: AGV control and status messages

Another required feature is the indirect control of AGVs, by assigning tasks to them. A system that assigns tasks them needs the user to assign tasks, but itself needs to decide what can be assigned. Tasks could be defined from the beginning by the user, but also could be defined based on the knowledge of the system. In any case, the user should be able to assign tasks to any AGV at any point using the interface. First, in order to get some more concrete requirements, let us narrow the field. First, tasks need to be defined.

Tasks, in the context of this research, can be any objective that is not yet completed, which requires completion, and can be repeated. This objective should be done by an AGV in context of the prototype. A task can be any objective an AGV is capable of doing, but mostly entails transport and logistics. Tasks should be able to be done of AGVs by themselves and should not require any help from humans in context of the task itself, though in case of error or emergency humans can be of help. Task assignment done by the operator can be done in a few different ways, such as selecting from a menu, or typing it in, but perhaps the most straightforward way is a mixture of both. There could be a menu that allows the user to search for a specific task if they know what they are looking for.

On the other hand of tasks there is status messages of the AGVs. The AGVs, handling a task or not, should be able to indicate what is happening at any given time. If there is an emergency of some sort, an interrupt could be issued to notify the operator. Otherwise, it is possible to create a grid or log of status messages from the AGVs, allowing the operator to know what is going on, and what was going on in the past with an AGV. Filtering options could be added, to allow an operator to focus on one specific AGV. An AGV could also be inspected in more detail, maybe by clicking it, or perhaps clicking a log belonging to it. The AGVs could be arranged on the screen to allow the operator to easily click and select each AGV and gather information about them. More detail could be granted if the operator wishes so, allowing them to see a better representation with all sorts of information.

Given these statements, a more clear requirement list can be created, as seen above with the information display. These are the following:

- First, the interface should consist of two main parts. First is the oversight of all the AGVs currently connected to the interface. They contains cells and an image of the AGVs, as well as having some background colour to indicate their status if one quickly looks at it. The second part consists of a log, a view that contains AGVs and status updates in a central chat. These logs will be outputted by AGVs when a certain state is reached in their programming. This could be picking up something, delivering something, or being in a spot that they cannot get out of. This log should mention the designated name of the AGV.
- Second, the AGV logos and or names in the oversight should have clear visuals and colours. If everything is okay, the colour of the AGV should be green, otherwise red or some other colour of emergency. Audio should accompany this in case of a highly critical situation.
- Third, the logs should display information clear and should highlight the most important words, so that any operator looking at can get a quick and clear view of what an AGV is doing at a given time. The rest of the text should still be visible, in case more information is required, but the important things that are critical in times of emergency should be highlighted to facilitate the ability to react to such a situation.
- Fourth, hovering over an AGV in the oversight should allow the operator to see more available information, and clicking an AGV or one of its logs should highlight the AGV and its logs.

5.4: Technical details

What has been described so far in this chapter are all technical specifications with a multitude of details. In order to put things into perspective and to wrap it up into a neat little package, the table below is used to keep track of and summarize the individual requirements.

Original Design Principle	**Technical requirement**	**Additions**
Proper display of information, visually, audibly or haptically.	Feedback should be labelled and coded with colours and shapes in mind. Audio should be used in accordance with this, and should use positive and negative sounds based on the association.	Using associative colours for warnings and good situations is key to understanding the interface, as well as using audio to notify the operator.
Indirect control of AGVs, via task assignment	A task assignment system should be created to allow the	This menu should also allow the user to change tasks even

	operator to assign tasks to the AGVs. This should be done via combination of dropdown menu and search menu per AGV.	if it is not required, as control should not be taken from their hands.
Status control and error messages	A log should be created as part of the interface that all AGVs can send data and status updates too, including errors. Clicking on a message or an AGV in the oversight highlights the AGV and the error messages.	These messages should highlight important words to give a first impression to the operator of what is going on.
Operator knowing the location of an AGV	Location and views may be represented into a more options tab that is located near the AGV in the oversight menu.	Perhaps this could also be integrated within the status log system, in addition to the more options tab.
Intuitive and easy to use interface	Using clear and associative buttons and colour should make the interface intuitive and easy to use	Often used symbols are symbols such as a cog to indicate settings. Using symbols that have already been used should hopefully reinforce the button's function.

Figure 3: Data in a table containing the design features transformed into technical features to be created in the prototype.

5.5: Game Engines

A representation of specific and environmental information can often be done using simple lines in a text file, but a much more interesting way is the usage of specific software to represent such information. One of these ways is a game engine. One way to define it is as a collection of codes that indirectly describe the behaviour of a modular game [14]. In this way, the behaviour of the game depends on the input of the player. This means that in essence the player is responsible for the game's executions and calculations. This means that if a game engine was used for a different purpose than games, say an informational interface, than the logic behind the graphics could deal with interpreting signals from an AGV instead. There are several game engines available, all with differences.

5.5.1: Godot

The first game engine to be examined is Godot. Godot (https://godotengine.org/features) is a program that allows a user to create games which, unexpectedly, is a typical feature of a game engine. Godot has tools that make it accessible to people to create their games and applications with relative ease. It supports an array of scripting languages, such as GDScript, a script not unlike Python.

It also supports C#, a programming language also used in Unity. Godot also supports C++. It also supports multiple platforms, such as Windows and Linux.

5.5.2: Unity

Unity (https://unity.com/) is another game engine, one based in C#. Besides containing similar features to Godot, Unity contains its own-made asset store, allowing user to create and publish code, models and other assets to be used for any user. Free and paid assets are both available on this store, allowing any user with a Unity account to get and download assets directly into their project.

5.6 Conclusion

In this chapter the design choices and principles explored in chapter 4 have been transformed into technical requirements for the creation of the prototype. It has been summarized in a table for a clear oversight of what needs to be done to realize the design. Unity has been chosen for this project because it offers a game engine that is not too difficult to work in, as well as its wide use and variability. On top of that the researcher has some experience with Unity, making it a choice because of familiarity.

Chapter 6: Realisation

6.1: Introduction

Now that the specification is properly defined, it is time to create the prototype. This is done by describing the requirements and transforming them into the working prototype. This is divided into a number of steps. The first step is the program structure and the setup used in order to create the prototype. Then there will be more sections in detail covering each part of the setup. Finally, there will be a part detailing the interface design and creation in Unity.

6.2: Requirements into realised parts

In order to note down what parts of the technical requirements are realised into actual parts of the prototype, the technical details table, Figure 2, is copied and edited.

Original Design Principle	**Technical requirement**	**Additions**	**Realisation**
Proper display of information, visually, audibly or haptically.	Feedback should be labelled and coded with colours and shapes in mind. Audio should be used in accordance with this, and should use positive and negative sounds based on the association.	Using associative colours for warnings and good situations is key to understanding the interface, as well as using audio to notify the operator.	Text in the interface displays some information, while a yellow colour is used to indicate a warning. Associative colours are used a bit less in this regard.
Indirect control of AGVs, via task assignment	A task assignment system should be created to allow the operator to assign tasks to the AGVs. This should be done via combination of dropdown menu and search menu per AGV.	This menu should also allow the user to change tasks even if it is not required, as control should not be taken from their hands.	Task assignment system that the user can click at a specific TurtleBot to then select a new task for them, or dismiss the window.
Status control and error messages	A log should be created as part of the interface that all AGVs can send data and status updates too, including errors. Clicking on a message or an AGV in the oversight highlights the	These messages should highlight important words to give a first impression to the operator of what is going on.	A status control window has been created that allows the user to see detailed information about the TurtleBot, and a log has been created that

	AGV and the error messages.		creates a new entry whenever an update happens within the interface.
Operator knowing the location of an AGV	Location and views may be represented into a more options tab that is located near the AGV in the oversight menu.	Perhaps this could also be integrated within the status log system, in addition to the more options tab.	Not implemented in details, only an error message and a log update gets created with the message that the TurtleBot is stuck.
Intuitive and easy to use interface	Using clear and associative buttons and colour should make the interface intuitive and easy to use	Often used symbols are symbols such as a cog to indicate settings. Using symbols that have already been used should hopefully reinforce the button's function.	In the interface plenty of symbols are used, most of them to indicate a certain part in the task assignment window.

Figure 4: Table that looks like Figure 2 with the addition of the realised parts in the right most column.

6.3: Proposed setup and program structure

The proposed structure for the prototype exists out of 3 main parts. First is the part that deals with ROS and Linux, the second part is the MQTT broker, and the third part is Unity. The setup will be like so: Ros will be setup on Ubuntu on a pc. This pc will then connect to Ros to control the TurtleBot. Then, a separate program on Ubuntu will be installed to host an MQTT broker. This broker will connect the terminal of Ubuntu to Ros. Finally, Unity will be installed on another pc and connect to the MQTT broker hosting on the other pc. In Figure 4 the structure of the program can be viewed in a more visual manner as opposed to a written description. The choice for this system stems from the difficulty of finding a starting point. That means that it was difficult to know where to begin in creating such a system. A student working on a similar project suggested to use an MQTT broker in an online message platform. After having been shown the system in action, this starting point for the writer had been reached. As such, the creation of this system can be traced to this interaction. This system may not be the best or easiest system, but in certain respects it is certainly provides structure and ease of use. The choice for a system at all also stems from the limitation that Unity cannot be installed on a Linux Operating System, such as Ubuntu. This had to be overcome and as such this system was used.

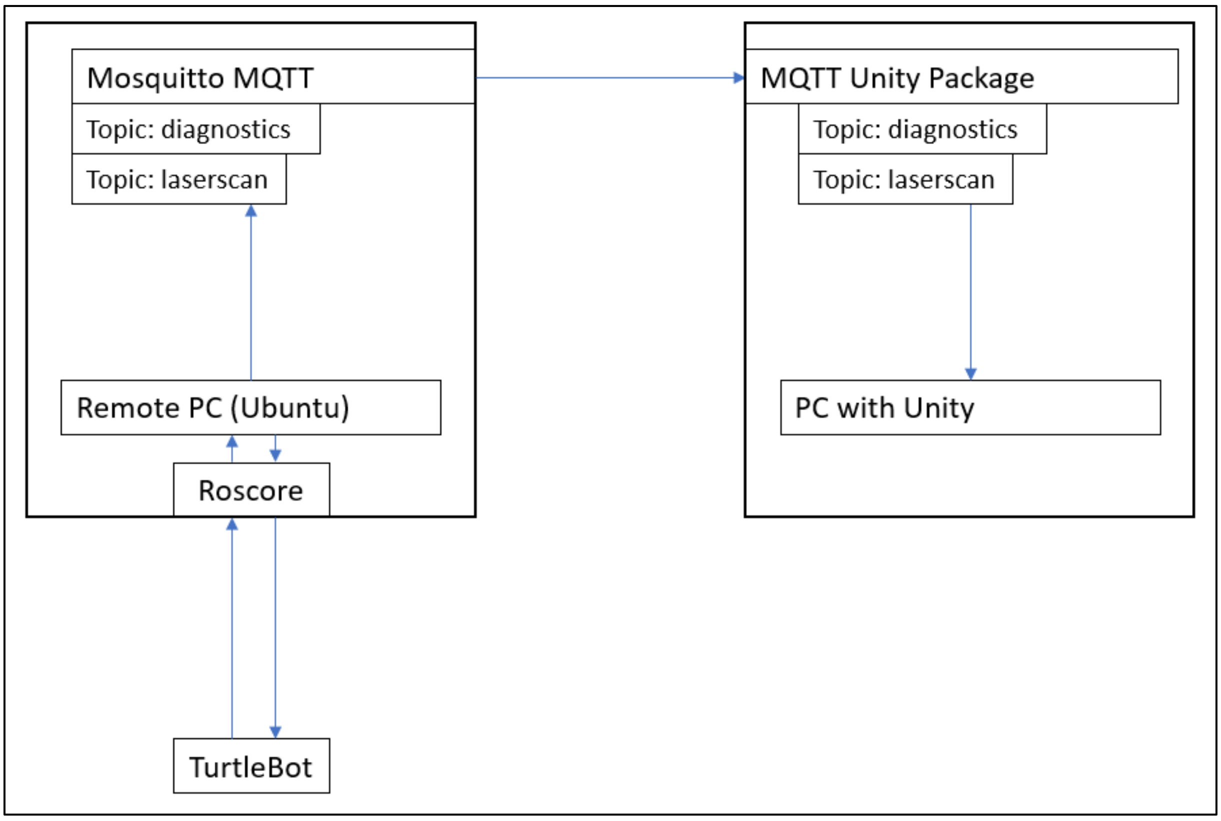


Figure 5: Image representing the program structure of the prototype.

6.4: TurtleBot

The TurtleBot has to be set up in order to use it for the prototype and to use it with a pc. In Appendix 1 this is explained in more detail, but in essence, the TurtleBot needs to be connected to and configured both on a remote pc and on the TurtleBot itself. This needs to be done by first connecting a screen and keyboard and configuring the Wifi. Then, it can be accessed remotely via the remote pc that has Ubuntu on it.

6.5: Ros

First, Ros will need to be set up in order to communicate and get data from the TurtleBot. To do this, first Ubuntu will have to be installed, as well as Ros afterwards. In Appendix 1, it details how not just the TurtleBot can be setup, but how Ubuntu has to be configured as well. Ros is a language that communicates data in robots.

6.6: Ubuntu

Ubuntu will be the platform to host Ros and by extension the program Roscore. Ubuntu is a an operating system that is Linux based.

6.7: Mosquitto

Mosquitto is a package that can be downloaded for Ubuntu. This package works as an MQTT broker, which can handle connections between machines. Mosquitto will be used on the pc that contains Ubuntu, making this pc the address that Unity will connect to. It creates a connection between the computer with Unity and the computer with Ubuntu. It does this via a bridge of some sorts where it acts as a broker between the two computers, by hosting itself on one of them and allowing connections from and to other sources.

6.8: Unity

Unity is a game engine and as such is used for games. However, it has plenty of other uses, since in essence it is a platform for rendering and UI options as well.

6.9: Unity MQTT Package

The package used to facilitate the MQTT broker within Unity is called Web API Kit: MQTT for IoT by Haptix Games and it takes care of the MQTT connection to a certain server and topics.

6.10: Connection

Now that the elements are there, the connection between the machines can be established by launching Mosquitto from the pc containing Ubuntu.

6.11: Interface

Now that the backend system is working, the actual interface as well as the systems that send data to it can be created. There are a few parts. First, the interface in Unity as well as the scripts required for actually making it work. Secondly, the scripts and processes running on the pc with Ubuntu that collect and send data to the broker, where the interface picks them up. The interface will be described first. Then the systems and scripts will be described from the Ubuntu environment.

6.12: Log

One of the larger parts of the interface is the log. The log, whenever an update happens within the interface, will be sent this data as well as other relevant information. Other scripts can refer to the instance of the log because the instance is one that is within the scene but static. As such a script can call the function to add a new entry to the log from anywhere. The new entry will then be displayed in the log. The entry of the log contains the type of update, the name of the unit affected or related to it, some information, potentially a reference image and the time and date. In the figures from 6 till 10, the log updates can be seen in the right side of the interface.

6.13: Task Assignment

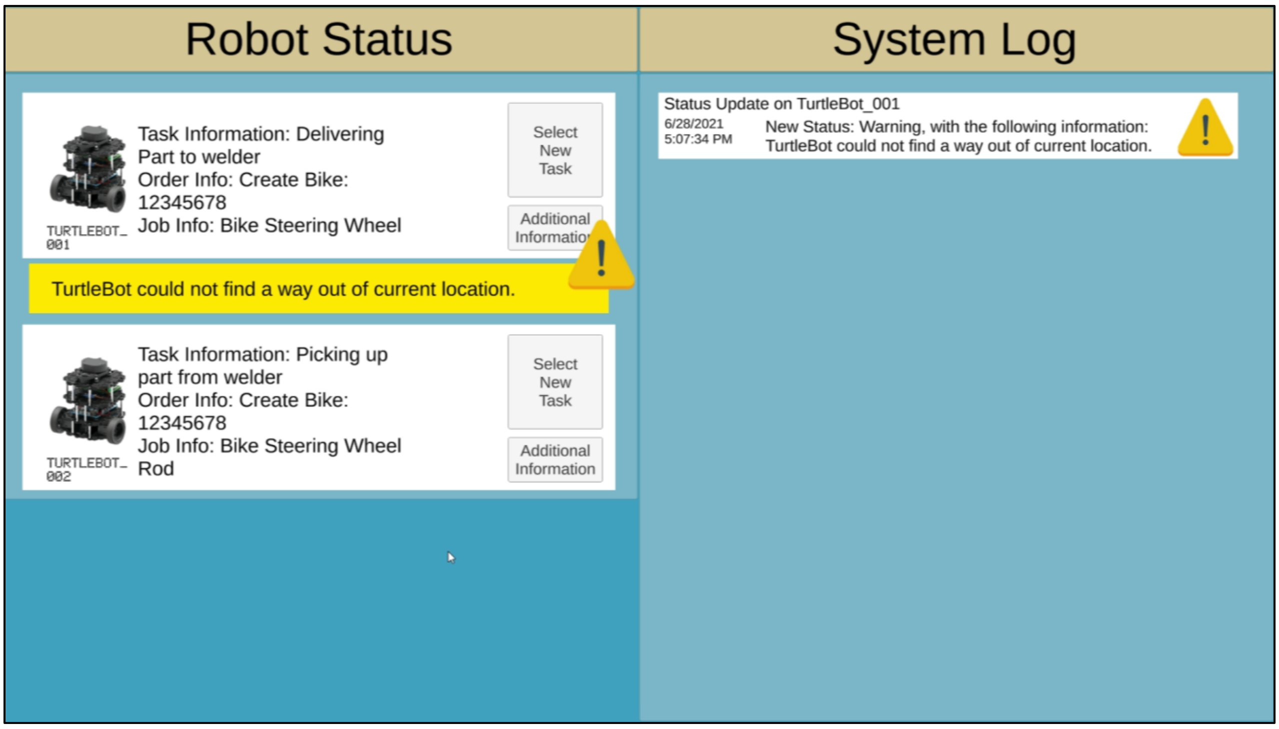
The second window shares its location with a diagnostics window, and this window allows the user to assign or re-assign tasks to TurtleBots. These TurtleBot will not actually execute these tasks, but it at least allows the user to select them. There are a few predefined tasks available, and choosing a new task for a TurtleBot will then list that task as the active task for that TurtleBot. In figures 9 and 10 this can be seen in the interface.

6.14: Diagnostics

This window shares its location with the task assignment window. This window, when clicked on a certain TurtleBot, will show some diagnostics about the TurtleBot's systems. This, like other parts of the interface will cause a log entry to appear with the information. In figure 8 this can be seen in the interface.

6.15: TurtleBot status

The status portion of the interface shows the user the status of the TurtleBot and what it currently is doing in terms of tasks. It also provides the icon of the task that it was assigned if it was assigned a new task. A special feature of each TurtleBot status in this area is if the first the first TurtleBot gets stuck, or rather, the average distance to its environment is lower than a certain threshold. If this happens a yellow warning bar with information and a warning icon. This can be seen in Figure 6, and it gets updated in Figure 7.



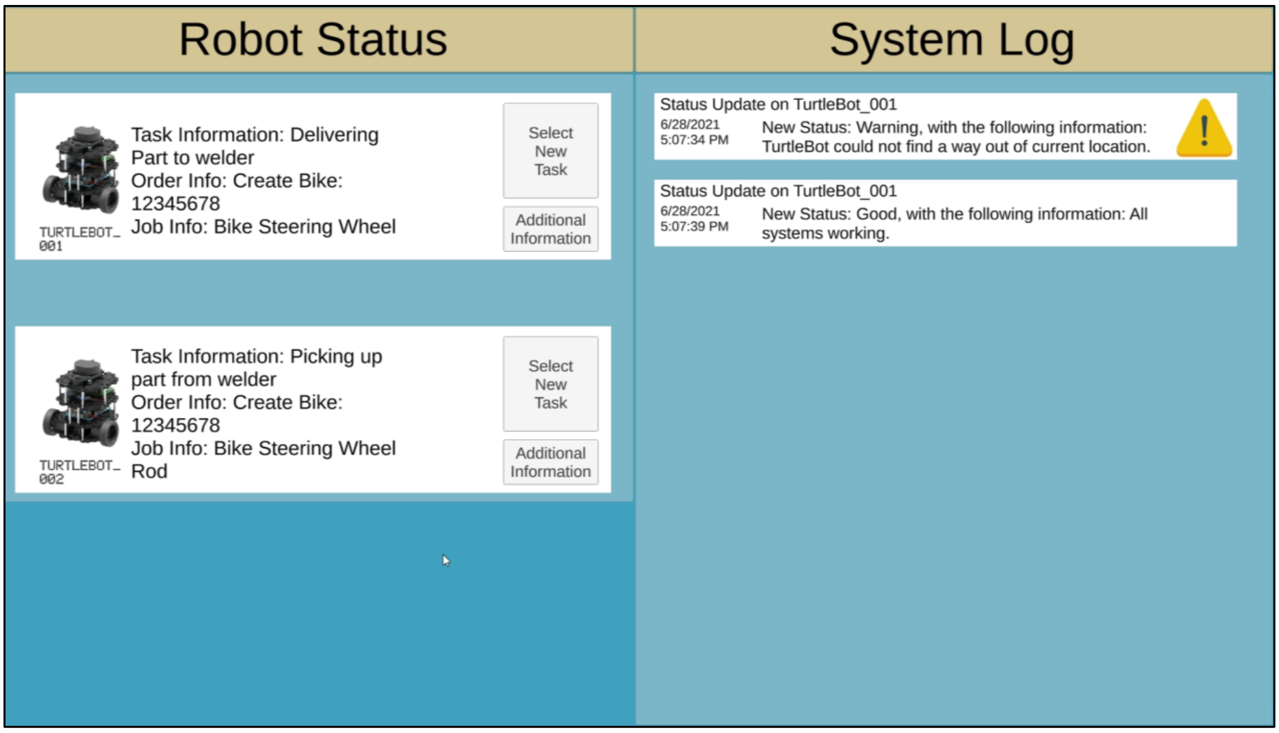
Task Information: Delivering Part to welder
Order Info: Create Bike: 12345678
Job Info: Bike Steering Wheel

TurtleBot could not find a way out of current location.

Task Information: Picking up part from welder
Order Info: Create Bike: 12345678
Job Info: Bike Steering Wheel Rod

Status Update on TurtleBot_001
6/28/2021
5:07:34 PM
New Status: Warning, with the following information:
TurtleBot could not find a way out of current location.

Figure 6: Interface displaying TurtleBot status and log update. A warning has appeared.

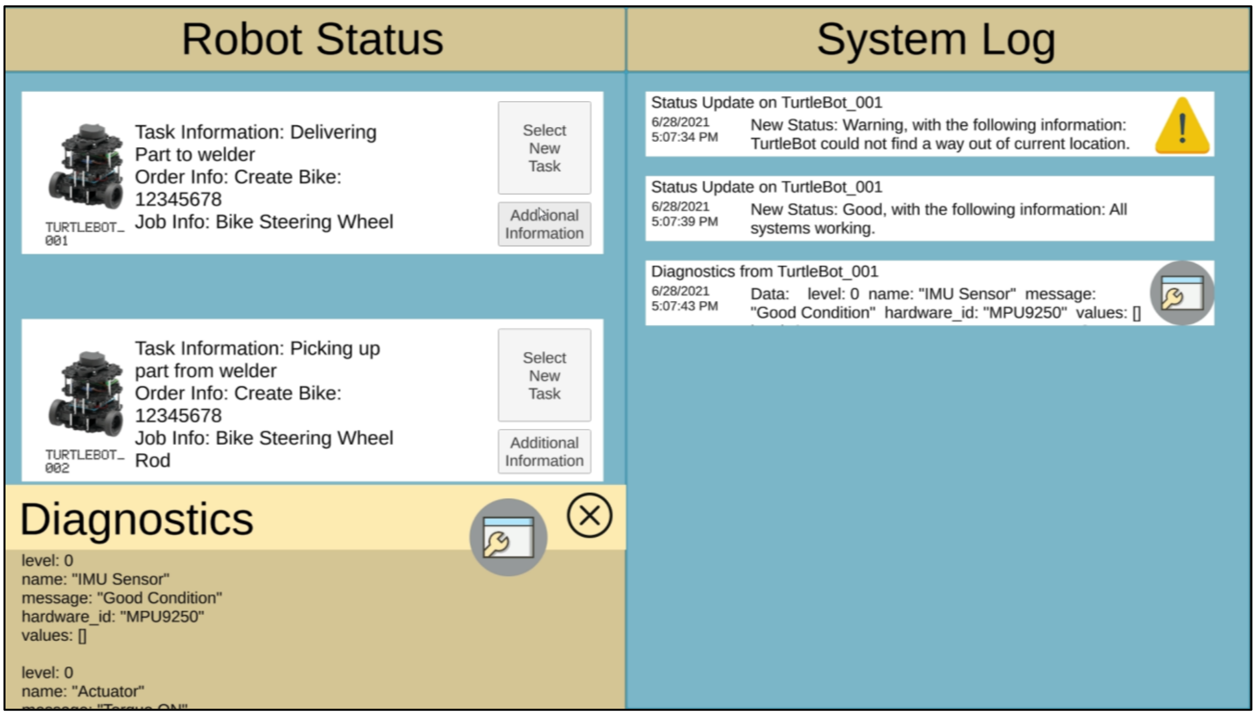


Task Information: Delivering Part to welder
Order Info: Create Bike: 12345678
Job Info: Bike Steering Wheel

Task Information: Picking up part from welder
Order Info: Create Bike: 12345678
Job Info: Bike Steering Wheel Rod

```
Status Update on TurtleBot_001
6/28/2021    New Status: Warning, with the following information:
5:07:34 PM   TurtleBot could not find a way out of current location.

Status Update on TurtleBot_001
6/28/2021    New Status: Good, with the following information: All
5:07:39 PM   systems working.
```

Figure 7: The interface where the warning was in Figure 5 has now disappeared, and it has been updated in the log.



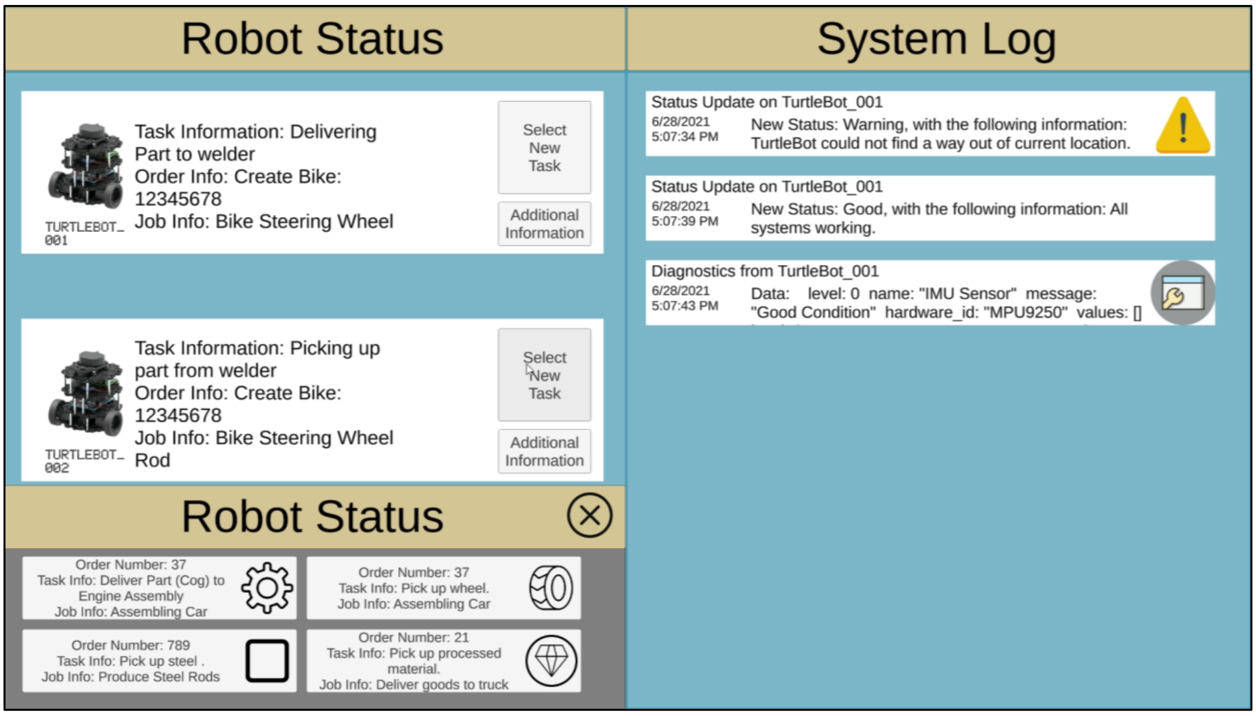
Task Information: Delivering Part to welder
Order Info: Create Bike: 12345678
Job Info: Bike Steering Wheel

Task Information: Picking up part from welder
Order Info: Create Bike: 12345678
Job Info: Bike Steering Wheel Rod

```
level: 0
name: "IMU Sensor"
message: "Good Condition"
hardware_id: "MPU9250"
values: []

level: 0
name: "Actuator"
```

```
Status Update on TurtleBot_001
6/28/2021    New Status: Warning, with the following information:
5:07:34 PM   TurtleBot could not find a way out of current location.

Status Update on TurtleBot_001
6/28/2021    New Status: Good, with the following information: All
5:07:39 PM   systems working.

Diagnostics from TurtleBot_001
6/28/2021    Data:   level: 0  name: "IMU Sensor"  message:
5:07:43 PM   "Good Condition"  hardware_id: "MPU9250"  values: []
```

Figure 8: In the interface the user clicked the additional information button, and some diagnostic data has appeared along with an update in the log.


Robot Status

Task Information: Delivering Part to welder
Order Info: Create Bike: 12345678
Job Info: Bike Steering Wheel
TURTLEBOT_001

Task Information: Picking up part from welder
Order Info: Create Bike: 12345678
Job Info: Bike Steering Wheel Rod
TURTLEBOT_002

Robot Status

Order Number: 37
Task Info: Deliver Part (Cog) to Engine Assembly
Job Info: Assembling Car

Order Number: 37
Task Info: Pick up wheel.
Job Info: Assembling Car

Order Number: 789
Task Info: Pick up steel .
Job Info: Produce Steel Rods

Order Number: 21
Task Info: Pick up processed material.
Job Info: Deliver goods to truck

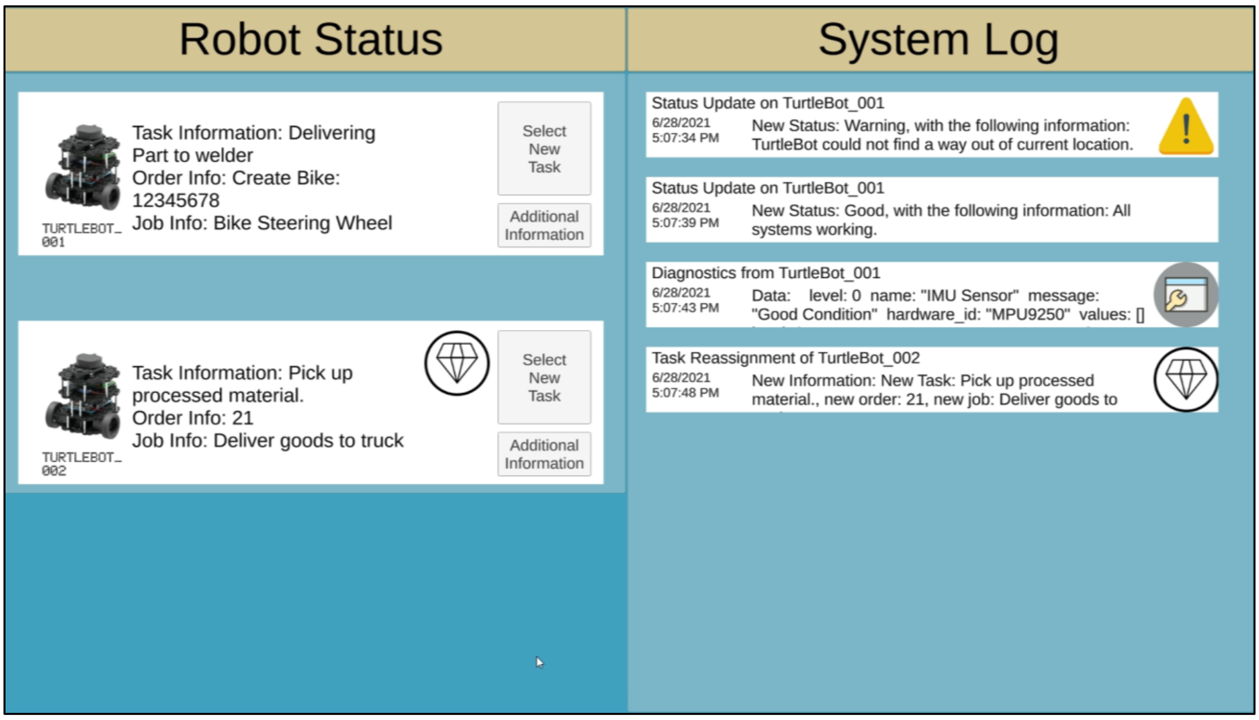
System Log

Status Update on TurtleBot_001
6/28/2021 5:07:34 PM
New Status: Warning, with the following information: TurtleBot could not find a way out of current location.

Status Update on TurtleBot_001
6/28/2021 5:07:39 PM
New Status: Good, with the following information: All systems working.

Diagnostics from TurtleBot_001
6/28/2021 5:07:43 PM
Data: level: 0 name: "IMU Sensor" message: "Good Condition" hardware_id: "MPU9250" values: []

Figure 8: In the interface the user has dismissed the diagnostic data using the X button and is now ready to select a new task for a TurtleBot after having clicked the select new task button.


Robot Status

Task Information: Delivering Part to welder
Order Info: Create Bike: 12345678
Job Info: Bike Steering Wheel
TURTLEBOT_001

Task Information: Pick up processed material.
Order Info: 21
Job Info: Deliver goods to truck
TURTLEBOT_002

System Log

Status Update on TurtleBot_001
6/28/2021 5:07:34 PM
New Status: Warning, with the following information: TurtleBot could not find a way out of current location.

Status Update on TurtleBot_001
6/28/2021 5:07:39 PM
New Status: Good, with the following information: All systems working.

Diagnostics from TurtleBot_001
6/28/2021 5:07:43 PM
Data: level: 0 name: "IMU Sensor" message: "Good Condition" hardware_id: "MPU9250" values: []

Task Reassignment of TurtleBot_002
6/28/2021 5:07:48 PM
New Information: New Task: Pick up processed material., new order: 21, new job: Deliver goods to

Figure 10: Interface where the user has selected a new task for TurtleBot_002, picking up some processed material, along with an updated log.

6.15: Mosquitto

In order to send data to the interface, Mosquitto has to be used to send messages to a certain topic. These topics are called “diagnostics” and “laserscan” and by using these topics, the messages can be easily identified by their topic. One could consider them lanes on a highway that are all going the same direction, but on slightly different routes. On the other end is Unity, who still receives and sends the messages to the same point, where they can be read and parsed. The choice for creating a central script for handling all these messages at the same point stems from the choice of wanting a central point that all messages can be sent to and received from, to create an easier to work with system where all things are handled at the same point. From this point, the messages received are checked to what data they contain and are then parsed and sent to the specific TurtleBot.

6.16: Scripts

In order to collect and send the data, a few scripts must be created in the Ubuntu environment to do these tasks. The tasks can be divided into 2 main parts, collecting and sending the data. Collecting the data requires roscore to be active on the pc. It also requires the TurtleBot to be up and running and to be connected to Ros on the pc. The script can then ask for information on a certain rostopic from ros. This data is then stored into a variable and then sent to the topic it needs to be sent to. Additional information is put in front of the data to identify what kind of data it is and to what TurtleBot it belongs in Unity. Then using Mosquitto, it gets sent to the interface.

6.17: Start menu, icon credit and asset credit

The several icons used within the interface were not mentioned during the actual evaluation experiment with participants. However afterwards the appropriate credit was listed in a specific credit window, found in the application. The participants did not have access to this, but it was added some time after that experiment. It lists the usage of what icons were used and who made them. On top of that, it lists the creator as well as the name as the MQTT asset used in Unity. This is contained within a credit window that is located in a start menu. The participants as mentioned before did not have access to this, but was added after the experiment.

Chapter 7: Evaluation

7.1: Interview and requirements

The interview and the requirements inspired from it discussed in chapter 4 will be discussed. In order to assess the quality of the requirements and a certain sense the prototype, the interview and its requirements will be examined and evaluated to assess the quality of these concepts as they are applied in the interface and thus the prototype. In Figure 10 these evaluations can be seen in table form along with the initial design principles as well as the realisations seen in the figure.

Original Design Principle	**Technical requirement**	**Additions**	**Realisation**	**Evaluation**
Proper display of information, visually, audibly or haptically.	Feedback should be labelled and coded with colours and shapes in mind. Audio should be used in accordance with this, and should use positive and negative sounds based on the association.	Using associative colours for warnings and good situations is key to understanding the interface, as well as using audio to notify the operator.	Text in the interface displays some information, while a yellow colour is used to indicate a warning. Associative colours are used a bit less in this regard.	Information is not always displayed in the best manner, as not many shapes and colours are used in association, and information is only really conveyed via visuals.
Indirect control of AGVs, via task assignment	A task assignment system should be created to allow the operator to assign tasks to the AGVs. This should be done via combination of dropdown menu and search menu per AGV.	This menu should also allow the user to change tasks even if it is not required, as control should not be taken from their hands.	Task assignment system that the user can click at a specific TurtleBot to then select a new task for them, or dismiss the window.	The user can assign tasks via a dropdown menu but not via a search menu.
Status control and error messages	A log should be created as part of the interface that all AGVs can send data and status updates too, including errors.	These messages should highlight important words to give a first impression to the operator of what is going on.	A status control window has been created that allows the user to see detailed information	The log located on the right side of the interface can receive all kinds of

	Clicking on a message or an AGV in the oversight highlights the AGV and the error messages.		about the TurtleBot, and a log has been created that creates a new entry whenever an update happens within the interface.	updates and display them. However clicking on a TurtleBot does not highlight the messages specific to that TurtleBot or vice versa.
Operator knowing the location of an AGV	Location and views may be represented into a more options tab that is located near the AGV in the oversight menu.	Perhaps this could also be integrated within the status log system, in addition to the more options tab.	Not implemented in details, only an error message and a log update gets created with the message that the TurtleBot is stuck.	Not implemented in detail and thus not really working in the interface.
Intuitive and easy to use interface	Using clear and associative buttons and colour should make the interface intuitive and easy to use	Often used symbols are symbols such as a cog to indicate settings. Using symbols that have already been used should hopefully reinforce the button's function.	In the interface plenty of symbols are used, most of them to indicate a certain part in the task assignment window.	Symbols are used in the interface but do not always represent usual depictions otherwise found in other interfaces.

Figure 11: Copied and edited figure 3 to account for the evaluation on the right side of the table.

As seen in figure 11, not all the design principles were inherently fulfilled. The design principles that were strived for in the specification phase were not or not fully created in the realisation phase of the project. As such the interface is limited in what it was originally designed to do.

7.2: Evaluation of the communication interface

Besides the user interface, the communication interface or technical interface needs to be addressed too. The idea of the technical interface was to create a method to transmit data between computers and operating systems. The MQTT broker system solves this issue. It has the minor disadvantage that it is a bit over the top if only used on one computer, causing a bit of inconvenience in terms of setting up the system. That does not deny the system its usefulness in terms of communicating data over a network, local or more widespread. The performance is decent, as the system is able to send and receive messages in a second or two at most. As such, assumably, sending messages in a local network will have the best performance in terms of transfer speed. Security of the messages is perhaps basic, but existing nonetheless, as messages that are published and subscribed too can be set up to use a password and username, which has to be set up. This increases security. One minor problem encountered specifically with the Unity project is that when the Unity instance sends messages back to the pc hosting Ubuntu, the message was a JSON-formatted line of text. This could be solved by using a program on the Ubuntu pc that decodes a string and returns the piece of data the user can specify. All in all, the program and the interface structure are decently user friendly, and the set up is not the most difficult thing to do. That said, the infrastructure itself is quite straightforward in terms of overall function; send a message and receive a message. To get used to it is a bit of a different story. It is not the most difficult thing, however a bit of experience with Ubuntu and perhaps VirtualBox for hosting a network locally might help with a setup of the system. Other than that, the infrastructure is quite straightforward and functional in its usage.

7.3: Interface testing overview

In order to assess the quality of the prototype itself a test was developed that can concretely say to what extent the prototype does what it was designed to do. This ensured that the prototype is at least somewhat known in its qualities. In Appendix 4, the precise details of the research protocol can be found, here we will provide an overview. The purpose of this research protocol was to detail how the research had to go about performing the experiment with the participants. The target group for this experiment was initially expected to be operators, but turned out to consist of people who have experience with such an environment.

The participants were guided through four steps, two of which were an introduction and a conclusion to the experiment. The other two steps are the participants interacting with and reacting to the interface and a semi structured interview to evaluate their opinions on this. The results of this experiment were quite controversial as the two participants both had very different opinions about the prototype and what it should do. However, the silver lining on this is the fact that the two participants provided a wide spectrum of opinions. This means that the prototype can be judged rather widely, although it doesn't have a large sample size. The two participants had quite different opinions about the prototype, but they also seemed to have slightly different professions. It means that the opinions are maybe not as reliable, but it certainly does not render them useless. One of the participants found the prototype quite functional while the other one found it probably around the opposite. One very debatable issue within the prototype was icon use. A couple of points of interest. In the prototype, a cog icon was used in order to display a certain part in a task for the task assignment window. However, in usual UI design, a cog can also refer to settings. This disconnect between one person's perceived intention of one potential use of an icon and the designer's intention for the icon provided a real problem in interacting with into reacting to the interface. It caused difficulty navigating the interface because it attempted to create a new connection between an icon and its intended use while there was already a widely reinforced definition present.

7.4: Interaction with the interface

In the interaction part of the experiment the users had to interact with and react to the interface. It had mixed results, like mentioned before. The task to find additional information was not interpreted as the task was originally intended, as the status information from a TurtleBot could be considered the additional information. The question was of course framed in a way that the participant had to find or locate additional information. The additional information button as such did not exactly display additional information but specifically diagnostics. As such the purpose of the button and the question were interpreted in an unforeseen way that the designer did not intend or design for. The log (and in hindsight the diagnostics window as well) also did not have a scroll bar, which was a problem considering there was far more information than the limited space of the window could show. Another issue that arose during the experiment was the issue of not having enough information and/or information from different orientations such as logistics. That means that during the experiment a mention was made that the information in the status window of the TurtleBots needed more and different information, such as production or article numbers, as well other details about the robot itself, such as the number of deliveries and what deliveries it made or its configuration. Mentioning a date and time with an order could also be useful. Important to note is that a potential improvement for error handling would be additional information such as a date and time as well as a specific production moment. On top of that, the production as well as the date and time of the production itself could be tracked as well. The production number as well as a date and time could be tracked as well the machines, because the production number can be the same but the actual production itself does not have to be. The error being resolved could have some improvement as well, since at the moment it is not entirely clear that the error is resolved, for example the lack of a green light. The button for diagnostics was not clear either. Some further comments were given in regards to potential other improvements, such as locating the buttons and information shown when such a button is clicked so that the user then knows what TurtleBot they are working. It is not clear from the first view what robot is selected, and this should be made clear to the user. Colours could be used in accordance with this. Colours could also be used in the interface, such as warm colours for errors, or blue for the system being frozen for instance. The system log was good, but could also use colours in a similar way as described before. A system can have a black background, like a terminal.

7.5: Summary and analysis of the interviews

Now to analyse the interviews. Both participants, as mentioned earlier, had different views on the interface. One found the interface confusing while the other one found it relatively intuitive. The interface was confusing to get used to because there was a lot of text. A comment was when another question was answered where the response was that the interface looked functional, where the comment was made that the system log may have been too big and may have had too priority. Another comment was made in regards to the overall look of the interface, where a lot of software used to look like the interface. The interface looked like older software, and meanwhile there’s been a large increase in terms of UX specifically designed for robotics, and interfaces get adjusted to look more improved. The interface, similarly to being mentioned before, had similar colours between several parts of the interface, which made it confusing to differentiate between the status and other components of the interface. Suggestions that came with the answer of the question were subjects like colouring the buttons for diagnostics and assigning tasks. Having the choices down on the bottom of the interface makes it more confusing. It does not give the participant reassurance on what robot they are currently working or which one is active in regards to the task assignment or diagnostics window. A user interface should always be intuitive and user-friendly. Here, an interface being intuitive should come in front of an interface being user-friendly. A user always knows what the specific button does because it is described with an icon, such as settings. As mentioned before, colours changing could also be used to indicate something being active. The interface would need to be more intuitive, since it is not quite that in its current state. Simple and clean is good for an interface. Let there be a decrease in complications and text, and let there be an increase in visuals and colours. It could also be possible to add a legend of some sorts to the interface in combination with icons, where this legend could also be made a bit smaller and could even have an option to be hidden if users do not think it to be required anymore. Buttons could also be assigned names. An important thing to remember is that users do not think the same way. Users also do not have the same experiences or the same ways of getting the interface. It is important that the user knows what they are doing. The legend and the other options could make the user more confident.

When the users reflect on the question relating to the warning event in the interface, the answer was that it may or may not be clear on what the error is exactly, but that it was unclear on how to solve it and that it can cause a user to lose focus. In order to solve the encountered problem, the participants noticed that extra information would be required to solve the problem. Information within a given specific area that it could be caused by, not just a notice of a warning. A description should be given in terms of what is going on with a robot not just in terms of its sensors, but what is going on overall. On top of that, it should be ensured that no information is missing, in the interface itself, which is always important of course. When a user loses focus they would need to look at the log to get some information. What would need to happen is to establish a connection between what a user does and where they need to look afterwards. One such way could be colours where the user clicks on something and both the thing they clicked on and something related change into the same colour. Something like this, this need for a connection, should be visible in the interface at the first moment that this happens. The entries in the log looked extremely similar. The log entries all need a date and time, and now they are not much noticeable or visible. Because of this disconnect it was difficult to resolve the situation, the warning that the TurtleBot was stuck, more quickly. Another suggestion was to add a search button to the log. If the data cannot be used, it is not a good idea to leave it like this. Another idea was an export data button for the log.

The next issue is whether the participants were able to do what they wanted to do with the interface in the time that they had available. Because the tasks were directed at them, not a lot of time was spent on discovering and interacting with the interface except in those moments that a task needed to be done. However, despite this, the interface was not entirely able to do all the tasks a participant wanted to do in a specific timespan due to the fact that the interface was not very intuitive and thus it took time to attempt to adjust to the interface instead of doing the tasks. The test was not like a standard interface test as the researcher guided the participant towards the tasks and as such, the interface was not really used as a usual interface. In other words, the interface was used in the test in a way that the researcher wanted the participants to, and limited the standard use of the interface because of the researcher and the experiment focussing on tasks, causing the data to be different than if a different test was used. In terms of recognisability of UI elements within the interface, some parts were noticed were parts like the window edges, closing icon buttons and title texts. The interface was divided in that aspect, making it functional and by some extension also intuitive. The interface can be improved in these areas. Another area that could be improved is icons, since some of them in the interface were confusing or unclear. One such example is the use of the diamonds icon in order to indicate materials in the task assignment window. Another example for an unclear one is a square that indicates a diagnostics window. An example that could indicate a settings menu would be one such as a square. There are intuitive icons for these kinds of things, like a smaller UI icon. The questions are what the users are familiar with, and icons can have very different depictions but mean very different things. A user depicted in an icon could be a logging icon. It may have a different colour or its gender may be switched, but it is always the same main idea for an icon. A home will always be a home if it is depicted as one in terms of icons.

7.6: Limitations and challenges

Sample size is one of the limitations of the research, since only two participants were found in the timeframe that testing was available. This may cause the data to not be entirely representative. Another limitation that occurred was the fact that the interface did not entirely conform to the initial requirements set in the ideation and specification phase of the project.

7.7: Conclusion

Even though there were some problems with the testing, the interface displayed that the main design principles of user interfaces and user experiences are quite important. On top of that, the system behind the interface as well as the interface itself has the ability to create a foundation on which to build other interfaces, as well as improve this one. The interface may not be optimal, but it certainly provides the ability to make it so.

Chapter 8: Conclusion

8.1: Conclusion

Within the project, a few research questions were asked. These are all answered: The communication interface and the user interface were created in a way that facilitated future work, especially the technical interface. Perhaps not entirely intended from the start, but the technical interface itself provides a solid framework on which to build in terms of further research, as it allows future research to use the established connection between a Linux based environment and a Unity created user interface. The user interface itself provided insight into the specific needs and wants of the target group that was related to operators. The data provided by the participants shows that the user interface in its design, although it did not fulfil all of the established requirements, was at least somewhat functional for its design. But we also concluded that it needs a lot of improvement. It does not mean that the research and the project by extension failed however, as the technical interface and user interface can provide a great deal of insight into further research in terms of user design, and specifically for this target group. Among others, it shows that the visual and icon design is extremely important in terms of user interface design, at least for this specific field.

8.2: Future work and recommendations

In terms of potential future work, it is important that icon and visual design is researched in more detail and applied to a user interface to facilitate the users and their experience in their usage of the user interface. The technical interface itself is largely in place, but there is an opportunity to improve on the efficiency and performance on the scripts that were used in sending data to the user interface. Currently it works by constantly receiving data from Ros once, sending the data to the MQTT broker once, and then repeating this as fast it is allowed to. One potential improvement could be to control the rate at which this is sent, which for example could be every second, as well as the efficiency to the processes to how this data is required, by for example not constantly disconnecting from the broker, but keeping a connection and sending the data every now and then. These improvements may be minor though, and focus more on the technical aspect than the user experience.

In any case, it is important that in future research it is made sure to not do extra work than required, as the in the case of this project the technical aspect and getting to know the Linux environment, as well as installing software took up significant time in creating the prototype. As such, using the projects that were created here to establish a foundation can reduce the production time of future prototypes and research.

8.3: Asset attribution and notes for future use

It should be noted that a specific asset used in the Unity project is a paid asset, and should be purchased in use with other projects or further development, unless this is handled (for example), via an appropriate moral and legal standpoint, such as a central development team, or a University with a central account. It should be noted that this is ONLY allowed when the policies of both the asset itself (if a policy exists there) as well as the policies of Unity are not breached. Finally, some of the images in the Unity Project, specifically the icons, were used from a website that allowed one to use them under specific conditions. These conditions can be read on their site, and this site can be found in the credit file in the project. On top of that, the project itself contains a credit window that lists the appropriate credit for the icons used in the interface, as per the sites instruction for attribution. For other images, links were used to attribute credit, as it was uncertain how to appropriately attribute an image from the TurtleBot Robotis website. As long as these points are kept standing, further development using the project should not be an issue.

Appendix

Appendix 1: Setup of the TurtleBot3, Ubuntu and Ros + Unity

Tutorial

To setup the TurtleBot, a tutorial found online was used. The tutorial can be found here: https://emanual.robotis.com/docs/en/platform/turtlebot3/quick-start/#pc-setup. After a quick first look, one may notice that a Linux environment must be used. The tutorial recommends Ubuntu 16.04, one such a Linux environment. The first step taken was to install a Linux environment simulator of some sorts. In this case, WSL2 was used. Then, Ubuntu was installed using WSL2 as its environment (no setup required this was done a while beforehand), effectively simulating a Linux environment for Ubuntu, which can be interacted with using the Ubuntu terminal. However, this did not work.

Virtual Adapter

This was because the environment is connected to the internet via a Virtual Adapter. This is an object representing and "connecting" the virtual Linux environment and the internet connection used in Windows 10. To install and use Ubuntu, this works fine normally. However, for the purposes of interacting with the TurtleBot, this will not work. The reason for this is that the TurtleBot and Remote PC mentioned in the tutorial, will be connected via Wi-Fi. This is not possible with such a setup, unless another source for Wi-Fi is used, such as Wi-Fi USB-stick, although even this may not be a viable solution. As such a USB-stick is not used, another option must be utilized.

Oracle VirtualBox

The second option to use Ubuntu was via a program called VirtualBox by Oracle. This program allows the computer it is installed on to simulate a virtual isolated instance of another environment, such as Ubuntu. However, this also will not work, for the same reasons listed with the Virtual Adapter problem, because this too uses an "ethernet" connection to connect the virtual internet to the computer's actual internet access.

Setup Ubuntu

The third and final option to fix this Virtual Adapter issue is to install Ubuntu on a bootable USB-drive. For this, a program or tool such as Unetbootin can be used (from: https://unetbootin.github.io/). This tool can create a live USB-drive, allowing one to run another operating system from that USB-drive. Using this tool, an ISO-file was downloaded containing Ubuntu 16.04 64 bit and was transformed into a live USB-drive using the aforementioned tool. Then, the computer can be restarted and another boot has to be manually selected. Selecting Ubuntu (and not installing it when the window shows up), will yield an instance of Ubuntu as an operating system, able to connect to Wi-Fi without much hassle.

Now, Ubuntu will be installed. First, a secondary drive is needed, in this case a portable hard drive was used. Installing Ubuntu (and making sure that this drive is formatted in the right way) will yield a hard drive with Ubuntu. The initial USB-drive used for installing Ubuntu is now not required anymore, since everything will be located on the portable hard drive. When starting the computer, entering the boot menu will now allow the operator to choose to boot either Ubuntu or Windows. Editing the priority list of the boot menu will change what operating system is booted by default.

Installing Ubuntu and installing packages

Now that Ubuntu is up and running, it is time to continue with the steps taken in the initial tutorial. First, several packages must be installed on the Remote PC, as it is called in the tutorial. These packages have to do with installing ROS on the computer. Be sure to use commands such as sudo apt-get update and sudo apt-get upgrade. Then, continue installing the other packages mentioned in the tutorial. Doing this will ensure that all the packages are installed and up to date. Rebooting from time to time may also fix unwanted errors and warnings about packages. It is however extremely important to, within the Software Download settings, tick the boxes that allow Ubuntu to download from other sources, as well as the main regions (like main and universe). On top of that, be sure to tick the boxes in another tab to allow Ubuntu to also download from other sources.

Wrong port

After following the tutorial for a bit, the OpenCR board has to be set up. However, an error may come up when defining a certain port. It claims the port to not exist. This may be solved by unplugging and plugging in the USB-cable that contains the OpenCR connection to the Raspberry Pi of the TurtleBot.

Card was not recognized

Attempting to redo the setup will yield another error, that the card of the OpenCR board could not be read. This may be fixed by setting the OpenCR board in recovery mode by pushing specific buttons, as mentioned in the tutorial. First, hold the PUSH SW2 button, than, press and release the RESET button, and finally release the PUSH SW2 button. This should fix the problem and the OpenCR setup should be able to complete successfully.

Keyboard controls

If the setup was successful, it should now be possible to connect to the TurtleBot using the computer, and using several terminals to control the TurtleBot with the keyboard of that PC, using the WASD-keys. Before this can happen however two things need to be done, in order. First, Roscore needs to be launched from the remote pc, and second, the TurtleBot3 bringup needs to be launched from the TurtleBot. Then, teleoperation nodes can be launched to drive the TurtleBot around.

Mosquitto

To connect mosquitto to Unity, we need mosquitto. Sudo apt install mosquitto and sudo apt instal mosquitto-clients will do that for us. Make sure that mosquitto is not already running, and then run it using the command mosquitto.Using the functions mosquitto_sub and mosquitto_pub, you can subscribe and publish messages by specifying a topic and a message by specifying the parameters -t and -m followed by your topic and message between quotation marks.

Unity
Download the Unity project and set the parameters in the setup script under MQTT. Specify the topics under the topics and sent messages by mosquitto_pub should now be received by Unity.

Appendix 2: Setup of ARC on a PC and a TurtleBot

Installing a server on the TurtleBot

For starters, the TurtleBot needs to have an EZBPi server installed on it in order to allow ARC to connect to it. To do this, we download a zip file with these files.

Installing ARC on the PC

To connect the TurtleBot to ARC, ARC of course is required to be installed on the PC. An account is needed, which can be made without much hassle on the website. This can be done before or after installing ARC, which can be found on the website as a free version. The free version offers mostly the same things as the paid version, with one limitation being only being allowed to be used either for education or personal use.

Connecting the TurtleBot to the PC

Now that the program is successfully installed, it is time to connect the TurtleBot to ARC. First, the IP-address of the TurtleBot needs to be found. Connecting a screen and a keyboard to the TurtleBot, or alternatively, using a PC to access it remotely via SSH, typing "ifconfig" in the terminal will display the IP-address where afterwards the EZBPi file can be run from a terminal.

Creating a program

A program can be created by putting in several different nodes, and possibly by writing some code to make a connections.

Appendix 3: Generated ideas using the ’50 Ideas’ method

Highlighted ideas are marked blue.

3D-Model representation on 2D-screen	VR-View showing the actual environment	Digital representation	Camera representation	Panel layout with colours and shapes
Sound based system	Touchscreen vs Mouse and keyboard	Voice and speech recognition based interface	Small 3d representations on a grid of AGVs	Status bars and panels displayed over everything else
Emergency buttons	Auditory warnings using sirens	Sonar based scanning interface for the AGV	Web browser based (both working and UI elements)	Haptic feedback On body or device that can be worn or carried
Annoying sounds in case of emergency to stimulate attention to it	Visual light/sound on the AGV in case of emergency	Task assignment system that uses nodes to indicate what is being done at what time by what AGV	Smell/taste based system that secretes certain odors for certain events	AR based system with notifications on the HUD
Soundboard based layout that has specific buttons for specific purposes	Terminal based interface that shows updates and statuses of AGVs	Overview of the facility map with AGVS and their location as well as small panels of data	Expected path prediction in the interface	Helper assistant that helps the person control the interface
Multi-camera view	Soundboard + map by making the locations of the AGVs light up on the soundboard	Excel sheet with ever changing information	Projector that projects information on the floor	Object recognition relating the object in camera space to a location on a virtual map
Task/location assignment list for AGV	Lighting up lines around facility and in interface to indicate AGVs and their paths	Minecraft representation	Music system per AGV	Predictions of the AGVs will do after they are done with their current task
Interface that selects a target location for an AGV to drive to	Status update log that contains all status updates like a chatroom	Google glass with ‘x-ray’ vision to display AGVs	Waypoint AR glasses that show AGVs paths	AGvs having colour and shape codes depending on what they are doing
Static map with simple LEDs to indicate signals	Spherical room with light indications to allow access to a	Pressure pads to physically tap where an AGV has to go	Follow command that lets the AGV follow a specific user	Hologram of the entire facility and touch-display

	lot of AGVs at the same time			
Singing control of the AGV	Laser pointer control	Easy creation of a task track by queuing items from a list	Tasks to be divided by morning, afternoon and evening	Kingdom of Keflings approach, where AGVs can be taught what do by showing them

Appendix 4: Research Protocol

Basic Outline

Step 1: Introduction and consent form
First, introduce yourself to the participant, let them know who you are, confirm who they are, and make them sign the consent form. Then, move to step 2.
Step 2: Interface navigation and reaction to events
The participant will navigate through the interface and will interact with and react to several events within the interface. Move to step 3.
Step 3: Semi-structured interview
After the interface navigation has been completed and notes have been taken, the semi-structured interview will be performed. In this interview, a few basic questions will be asked to assert the interface of its potential, and to look for any potential changes. Move to step 4.
Step 4: Closure
In step 4, the participant will be thanked for their participation, and contact information will be given to them if they have questions, complaints or other comments. End research.

Refined Outline:

Step 1: Introduction
- Meet in online meeting room via Microsoft Teams.
- Make sure that the brochure and consent form are signed beforehand. Send them via email and make sure that they have been received a good amount of time before the research.
- Introduce them to the research and refresh their memory on the brochure. Ask them if they have any questions, or any comments at all. Then make sure that everything is well understood, and move to step 2.

Step 2: Interface
- Introduce the participant to the interface, and use Microsoft Teams to allow the user to actually control the mouse on the local computer. This can make sure that the user can actually interact with the interface.
- Then, execute the first testing procedure, which is the interaction with the interface test. This test focusses on getting the participant to see how intuitive the interface is to use. The test in that sense focusses on not just getting them used to it, but testing how fast that happens, or if it happens at all.
- The test goes as follows:

o There are 3 situations that the user has to do something about. A better way to describe this would be 3 specific assignments. In the interface, there are plenty of things to do, and a lot of things to take a look at. The user has 3 assignments to be completed.
o The first assignment is to figure out what a certain TurtleBot is doing. The user will be asked to take a look at TurtleBot_001, and be asked to inspect it. Specifically, they will be asked to get relevant information on its processes. Processes are those such as delivery, job, and order number. The user will be asked to see if they can find these statistics and details of TurtleBot_001.
o The second assignment is for the user to quickly check what all TurtleBots are up to, to see if anything is out of the ordinary, and to see if all TurtleBots are doing what they are supposed to be doing.
o The last assignment is to get more detailed information about a TurtleBot. The user will be asked to get some data about diagnostics of Turtlebot_001. The user will have to see if they can find this specific information within the interface, such as battery level of the TurtleBot.
o Then, the user has to do a slightly different thing. The interface will then be tested in terms of response time from the participant. In order to test this, the researcher will surround a TurtleBot

with a box, which will trigger a piece of code in the interface that will show that this TurtleBot is "stuck". The simulation works by checking if the distance from the TurtleBot in every direction is too small. That means that surrounding the TurtleBot with a box triggers this code. The participant will then be asked to see and identify the problem.

- After these few tests, move to step 3.

Step 3: Semi-structured interview

- In order to assess the situation of the interface and to assess its quality, a semi-structured interview is held with the participant. The participant has the ability to ask questions and deliver comments, and if an interesting answer is given by the participant, the researcher may ask further. The questions are divided in the sections
- The following questions are to be asked:
  - 1. How did you feel about getting information, did feel confident in knowing where to find it?
  - 2. Did the interface make sense in terms of location of information, buttons, colours, and visuals? How did the interface feel in terms of relatability to usual interfaces?
  - 3. Where you able, after the error with the TurtleBot, to find the issue in a time you thought you found reasonable within a time limit? If this time was within a facility, would that be optimized and in time to prevent any additional problems?
  - 4. Did the interface allow you to do what you wanted to do in the time span that you were active with the interface?
  - 5. Were the elements in the interface familiar to you in terms of other interfaces, and were these elements relatable enough to allow you to recognize certain parts of the interface as parts in previously used products? If yes, which products were those?
- Are these few questions, move to step 4.

Step 4: Conclusion

- After the tests have been concluded, the researcher will move to conclude the session.
- First, thank the participant for their participation and ask them if they have any questions.
- Give them the ability to comment on anything, and let them know that they can ask via mail if they have any questions or want their data to be removed, and let them know that this can be done within 3 days after the research has been completed. Let them know that their data will be removed at maximum 2 months after the thesis is finished.

Appendix 5: Extra Research

GoBot

GoBot is a program, or framework, that allows one to write programs in a specific programming language, Go, and to use it for robotics, physical computing, and the Internet of Things (https://gobot.io/documentation/getting-started/). It makes it possible to make high-level functioning programs relatively easily. It allows the user to connect several different devices, that in some way have an API.

What this information means for the user is that they can with relative ease (quickly) create high-level applications in ways that bring new meaning to the devices.

ARC

Arc (https://synthiam.com/Products/ARC) is program that allows users to easily create social robotics in a fraction of the time it would normally take them using traditional methods. It contains several 'skills', which can be viewed as nodes. These nodes each have separate functionality and can be interconnected, allowing the user to develop a vast array of different applications. ARC has many of these skills, from speech recognition to cameras, including several recognition criteria with it as well, such as face recognition. ARC has several price ranges, but even the free version allows a user to do plenty of activities within the app. ARC also has a skill store, which might be comparable with a sort of manager of adding and developing new skills to a project. New skills that are normally not present within ARC can be downloaded and put to good use, while also skills can be developed to one's own wishes. For specific uses that do not need a skill, but perhaps a specific function, scripts can also be added within ARC as a separate skill. The skill can then be programmed in a number of programming lanugages, such as Blockly and Python.

However, the important point, maybe the most important of all, is that several devices can be connected to ARC. This also means that a TurtleBot can be connected to it. This has already been successfully tested by the author using a local server on the TurtleBot that ARC can connect to. This is done by using a program called EZBPi. The TurtleBot contains a Raspberry Pi, from which it can be operated and controlled. This also means that installing new software comes with relative ease, as a user can simply connect to it via a remote terminal, or plug in a screen and keyboard and/or mouse, and the interact with it. With ARC, a EZBPi zip file can be downloaded and unzipped within the Raspberry Pi. Then, if the user runs a specific executable, the Pi will host a specific server. ARC can then connect to this server, and by extension the TurtleBot itself.

Sources

[1] M. Slot, P. Huisman, and E. Lutters, “A structured approach for the instantiation of digital twins.,” *Procedia CIRP*, vol. 91, pp. 540–545, 2020, doi: 10.1016/j.procir.2020.02.211.

[2] F. Orellana and R. Torres, “From legacy-based factories to smart factories level 2 according to the industry 4.0,” *Int. J. Comput. Integr. Manuf.*, vol. 32, no. 4–5, pp. 441–451, 2019, doi: 10.1080/0951192X.2019.1609702.

[3] L. Monostori *et al.*, “Cyber-physical systems in manufacturing,” *CIRP Ann.*, vol. 65, no. 2, pp. 621–641, 2016, doi: 10.1016/j.cirp.2016.06.005.

[4] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, “Digital Twin in Industry: State-of-the-Art,” *IEEE Trans. Ind. Informatics*, vol. 15, no. 4, pp. 2405–2415, 2019, doi: 10.1109/TII.2018.2873186.

[5] G. Schuh, H. Rozenfeld, D. Assmus, and E. Zancul, “Process oriented framework to support PLM implementation,” *Comput. Ind.*, vol. 59, no. 2–3, pp. 210–218, 2008, doi: 10.1016/j.compind.2007.06.015.

[6] B. Schleich, N. Anwer, L. Mathieu, and S. Wartzack, “Shaping the digital twin for design and production engineering,” *CIRP Ann. - Manuf. Technol.*, vol. 66, no. 1, pp. 141–144, 2017, doi: 10.1016/j.cirp.2017.04.040.

[7] L. R. Elliott, M. D. Coovert, M. Prewett, A. G. Walvord, K. Saboe, and R. Johnson, “A Review and Meta Analysis of Vibrotactile and Visual Information Displays,” no. September, p. 36, 2009.

[8] H. Lasi, P. Fettke, H. G. Kemper, T. Feld, and M. Hoffmann, “Industry 4.0,” *Bus. Inf. Syst. Eng.*, vol. 6, no. 4, pp. 239–242, 2014, doi: 10.1007/s12599-014-0334-4.

[9] I. F. A. Vis, “Survey of research in the design and control of automated guided vehicle systems,” *Eur. J. Oper. Res.*, vol. 170, no. 3, pp. 677–709, 2006, doi: 10.1016/j.ejor.2004.09.020.

[10] S. Haag and R. Anderl, “Digital twin – Proof of concept,” *Manuf. Lett.*, vol. 15, pp. 64–66, 2018, doi: 10.1016/j.mfglet.2018.02.006.

[11] N. J. Belkin, P. G. Marchetti, and C. Cool, “BRAQUE: Design of an interface to support user interaction in information retrieval,” *Inf. Process. Manag.*, vol. 29, no. 3, pp. 325–344, 1993, doi: 10.1016/0306-4573(93)90059-M.

[12] Z. Zhu, C. Liu, and X. Xu, “Visualisation of the digital twin data in manufacturing by using augmented reality,” *Procedia CIRP*, vol. 81, pp. 898–903, 2019, doi: 10.1016/j.procir.2019.03.223.

[13] A. Mader and W. Eggink, “A design process for Creative Technology,” *Proc. 16th Int. Conf. Eng. Prod. Des. Educ. Des. Educ. Hum. Technol. Relations, E PDE 2014*, no. September, pp. 568–573, 2014.

[14] M. Lewis and J. Jacobson, “Games engines in scientific research,” *Commun. ACM*, vol. 45, no. 1, pp. 27–31, 2002, doi: 10.1145/502269.502288.

[15] Productplan.com. 2021. MoSCoW Prioritization. [online] Available at: <https://www.productplan.com/glossary/moscow-prioritization/> [Accessed 18 July 2021].