MÄLARDALEN UNIVERSITY
SCHOOL OF INNOVATION, DESIGN AND ENGINEERING
VÄSTERÅS, SWEDEN

Thesis for the Degree of Bachelor in Computer Science - DVA 331

VR/AR and Digital Twin for improved visualization of overview and debugging of live hardware in next generations industry.

Kristian Jansson Room
krm17002@student.mdh.se

Joakim Karlsson
jkn17010@student.mdh.se

Examiner: Moris Behnam
Mälardalen University, Västerås, Sweden

Supervisor: Leo Hatvani
Mälardalen University, Västerås, Sweden

2020-06-12

Abstract

This thesis describes the implementation of a Digital Twin tied to Virtual Reality environment that could, by easy means be expanded to Augmented Reality-solution. The field is of interest due to the fact that movement into Industry 4.0 puts the traditional operator in a new seat of work. Previous hands-on tasks are replaced with system monitoring and supervision roles. New interconnected industrial hardware allows for extensive data collection, while interactive technology like VR/AR helps monitoring live systems in completely new manners. An operator can overview and debug industrial systems while not even being in close proximity of the physical system. This provides the opportunity to increase the level of system information presented to the operator. The Cyber Physical Factory created by Festo was targeted to be represented as a digital twin. The question asked: *What are the advantages and/or disadvantages of monitoring and troubleshooting a Festo CP-Factory by means of a digital twin-driven visualization?* proved to be extensive and the work included mapping of important factory data and DRM-research to find visual improvements between the provided solutions by Festo, and an implemented digital twin. The solution we produced focuses on overview and debugging and it connects to OPC-servers on each mapped module of the Cyber Physical Factory and acquires the data. This data can then be used to expand, test and debug previous sessions. Each real implementation of a factory has some type of logging of data, our solution allows visualization of those log entries as close to reality as possible, reducing the need to search databases for indications of problems. The Unity 3D created software also handles dynamic connections where the operator can modify which nodes to connect to in an intuitive way, this enables our software to abstract and modify information outside of the source code.

Table of Contents

Table of Resources

1. Introduction

This thesis describes the implementation of a Digital Twin solution tied to Virtual Reality that in easy means could be expanded to Augmented Reality-solutions, which we argue is the future for machine-human interactions. This is an interesting field due to the fact that movement into Industry 4.0 now allows us to map and understand live systems in completely new manners, allowing an operator to even overview and debug hardware in completely new ways, while not even requiring the operator to be close to a physical factory. As VR/AR technology gets cheaper and better we can actually use it in real industrial solutions to improve upon work-factors. Previous generations of the technology were not powerful and cheap enough but today the technology is even widely used by the general public, the industrial solutions are still in its early phases where most solutions are still only theoretical, we aim to implement these theoretical aspects in a short timespan. We used a module based system available to us at MITC (Mälardalen Industrial Technology Center based in Eskilstuna, Sweden) called Festo Cyber-Physical-Factory. The hardware and its platform is developed for research and education which was perfect for work like this thesis.

Digital Twin as an expression has gained interest the last 5 years in an increased manner but NASA brought the expression to the public attention already in 2010. Previous work tries to represent reality as close as possible and even allow for interaction between physical and digital twins. Festo already has a Digital Twin implementation in VR called CIROS but it uses strict definitions of DT-solutions. We aim to improve upon this software in a more generic fashion, applicable to most systems where we can read and understand data in a better fashion. We instead focus upon altering reality if it is incremental towards understanding, overview and debugging by abstracting the physical system in ways not always possible with strict definitions of DT-systems. Festo also provides an AR application usable with handheld devices as an example, we never tried to implement something better but instead used our work as proof of how the technology could be used instead, since the usage of a hand-held device will hold the industrial operators work-flow back. While it certainly works as a quick overview of smaller parts of the system, we instead want to aim higher and revolutionize the industry towards Augmented Industry where machine and human becomes more connected and we need better solutions as explained in this work.

We found interesting related work in our literature study: The thesis "*Virtual Production Line -Virtual Commissioning"* explains the development of a digital twin in welding using Unity3d together with OPC communication but the work does not include the VR/AR aspect. In an attempt to find benefits in simulation applications, *"Using Unity3d as an Elevator Simulation Tool"* investigates the possibilities of using Unity3d as a modeling tool for elevators versus a Discrete event simulator. In the field of industrial data communications, the paper "*Benchmarking of existing OPC UA implementations for Industrie 4.0-compliant digitalization solutions"* found out to be useful in our search for a OPC-UA framework to be used in Unity3d. Our definition of digital twin does not allow for controlling the physical twin, the work "*Digital Twin Based Synchronised Control and Simulation of the Industrial Robotic Cell Using Virtual Reality"* however, researches the possibility of controlling an industrial robot through a digital twin.

This report and our work aims to give examples of and prove how real Digital Twin solutions can work and what they should focus upon when being created. This means any organization that has I4.0 compatible hardware, mainly tied to production, will find the work done here incremental towards how their solutions could be developed, especially if it is tied to overview and error handling. And by implementing our ideas we should see improvements to productivity tied to understanding of live hardware (or previous/saved runs) as well as decrease fatal downtime of real factories.

The question asked: *What are the advantages and/or disadvantages of monitoring and troubleshooting a Festo CP-Factory by means of a digital twin-driven visualization?* aimed to map important backend data and then improve upon the aspects as we presented above. The reason the question is relevant is because we need to define the impact of DT-solutions and their pros and cons before any organization is willing to invest in the technology.

We first defined and mapped the backend systems (Festo modules) and then used DRM-research to find improvements between the current solution, called MES4, towards our implemented twin. The solution we produced focuses on overview and debugging and with that perspective we also analyzed the current solutions that come with the Festo-CP-F system. Our software manages to connect to OPC-servers on each mapped module and save the data. This saved data can then be used to expand, test and debug previous sessions which is powerful. Each real implementation of a factory has some type of logging of data, our solution allows visualization of those logs just as close to reality as possible via VR and the Digital Twin, no more searching datatables for explanations after fatal errors. The Unity 3D created software also handles dynamic node-connections where the operator can modify which nodes to connect to in an intuitive way, this enables our software to abstract and modify information outside of the source code.

We found in our work that the impact of a DT-solution can be great if it is developed with the right focus. Our work focus upon overview, understandability and error clarity so that is the perspective we developed from and argued in this report. We also found some real technical improvements over the original, MES4, solution. A VR solution tied to DT's is a great way to simulate the real world and this can be important for replays of captured sessions, simulations of underlying data and test the security and other aspects of systems without ever touching the real twin. Also the impact of this technology in hazardous environments could be substantial, especially if we allow a connection back to the PT from the DT so the operator can steer and operate from a safe environment. Our work also arguments that AR as an technology could be revolutionary "on the work floor" of real implementations if it is tied to holographic lenses where we can expand upon reality in a non intrusive way, if it helps workflow, understandability and productivity.

2. Background

Companies are interested in discovering possibilities and advantages of technologies such as VR/AR for industrial applications. We aim to improve upon these in new ways of thinking by adding a digital twin for debugging and overview that can be interacted with. The industry is looking into digital twins to improve various aspects of manufacturing but it is still in early stages. The following sections describe some important terms to fully understand this thesis work. First the Industry 4.0 paradigm is described which is in the field of this thesis. Later we are inspecting the components of Industry 4.0 and several focus points emerge like the Cyber Physical Systems, Digital Twins and other means like VR/AR that play a significant role. As one of the key factors of Industry 4.0 is connectivity, we describe the communications standard OPC because this is frequently available in the industry. Below follow some important topics that let us achieve the creation of an interactive digital twin in a virtual and augmented environment.

2.1. Industry 4.0

First defined at Hannover Messe, Germany in 2011 [1],[2], the overall industrial trend is aiming for its next phase, called Industry 4.0, where factories are intertwined with computers and each other to a greater degree for improved construction. The focus now lies in increased access and control with the help of communication between all parts of an industrial environment, all in real time. This is the next step in our industrial evolution and thus work towards this field is important. Smart factories can and will use data to learn and predict machine's needs for maintenance and costly repairs. This means that a production flow also can be supervised to a greater degree where every component could report its status in the production chain. We strongly believe our work could be beneficial towards this part of Industry 4.0.

Industry 4.0 also stands before a great problem regarding security [3], since these systems are often connected with each other through IIoT(Industrial Internet of Things) they are vulnerable to outside attacks. Scientific research promotes [3] that digital twins could help prevent disasters by preventing and testing these problems with digital counterparts, without affecting a real system. Thus, our research is important to the future of these major systems and Industry 4.0´s success.

2.2. Festo and the Cyber Physical Factory

Festo is one of the companies that is leading in digitalization together with physical products. Their primary focus lies in systems connected to education and research and achieve this by developing Industry 4.0 and smart solutions.

Cyber Physical-Factory (CP-F) is a Cyber Physical System (CPS) that is developed by Festo and is a fully functional system where the customer can build a factory on a lesser scale [Figure 1][Figure 2] for research and learning purposes. It is module based and is customizable to the user's needs. The system uses conveyor belts to transport carriers, see [Figure 58], towards different stations, as is common in production flows. The different module-roles range between anything from storage units, drilling connections, assembly in different stages, product control and transport between factories/modules via robots and all of these units do these tasks along with available sensor logic.

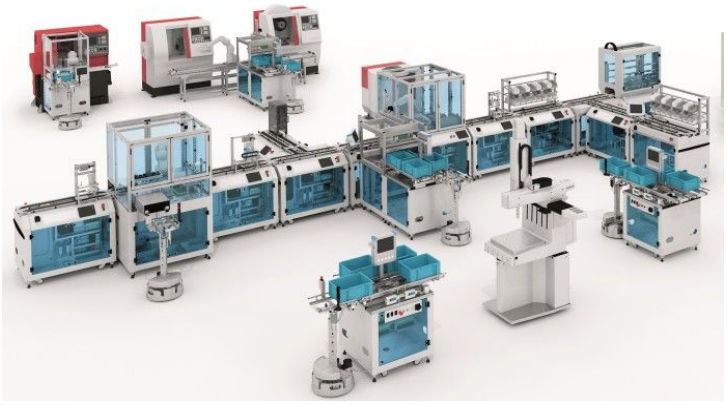

Figure 1, Festo modules:
Modules from Festo CP Factory. In this picture can we see how different robots send units between different modules [4].

Figure 2, Festo real size:
CP Factory in real scale to understand the size of the system [4].

2.3. Digital Twin

A Digital Twin (DT) as a concept is important for the upcoming discussions and will be our primary focus. The different perspectives of digital twins are widespread and thus we need to define our viewpoint, this is discussed in more detail in section 3.1 and we make the definition in 4.2. The Digital Twin was created in the context of aerospace (NASA) in 2010 (finished by 2012) [1] and were first defined as: a multi-physics, multi scale, probabilistic simulation of a flying twin or system connected to the latest sensory data, physics models and historical data (not live). In 2012, the same year when the work was finished [1], new definitions arose and a new perspective, interesting to us, are the one for prognosis and diagnostics activities. The next big step for DT was in 2015 when [5] defined DT with more generic uses, it still targeted aerospace but the concept became interesting for other fields. In other words, over the last 5 years, DT's has been more and more adopted as a topic in other fields, such as industry. This late development is one of the reasons for the many different perspectives. One should note that previous research was made before 2015 as explored in 2.6. The Digital Twin generally aim to mimic a real world system, the Physical Twin (PT), as close as possible for improved reality. It is often used for different simulations, such as aerodynamic simulations in aerospace to understand and improve upon how the physical system would behave in theoretical and real situations.

2.3.1. Early days of Digital Twins and their definitions

According to [1, Table 1. Definitions of Digital Twin in literature] the early structure of DT and previous work were:

2.3.1.1. 2010 - 2012 NASA creates the Expression

NASA defined DT in 2010 publically and its definitions were worked upon until 2012. They defined the system as a multi-physics, multi scale, probabilistic simulation of a flying twin or system connected to the latest sensory data, physics models and historical data. It was meant to specifically mirror the life of its flying twin. This made most of the following research first to focus on aerospace use-cases.

Other work in 2012 saw DT as a cradle-to-grave model of an aircraft structure. Which means it was meant to map a physical system from its creation till its destruction. It included submodels of the electronics, the flight controls, the propulsion system, and other subsystems for complete mapping. Ultra-realistic definitions also arose tied to aircraft structure to make sure the system could handle mission requirements.

2.3.1.2. 2013

New definitions and areas were created which meant other fields began to see uses for digital twins as well. Cloud-platforms became coupled with DT-systems in research, in other words, real units. They were simulating the health and condition of the machinery inside the network systems and used analytical algorithms together with physical knowledge.

Research also went into ultra-high fidelity physical models where focus went into the materials and structures that control the lifespan of a system, mainly vehicles. Noteworthy in 2013 is also structural models that included quantitative, in depth, data of materials with high sensitivity.

2.3.1.3. 2014 - 2015

More work towards other fields gets done and more wider perspectives arrive, for example we have very realistic models of a process and its current state. Mapping also went towards system behavior in interaction with the environment in the real world. A more generic approach also gets defined, digital counterparts of physical products is a reality [5]. Aerospace also gets ultra realistic models mapped to each unique aircraft, generally this was not the case previously, this was combined with known flight histories. Research also came up with high- fidelity structural models, they mainly focused on fatigue damage and presented a fairly complete digital counterpart of the actual structural system.

2.3.1.4. 2016

Work similar to ours emerges from multiple sources, for example Internet of Things (IoT) gets integrated by [6] with virtual substitutes of real world objects. Focus was on communication and virtual representation of the objects on the other end. Together the IoT network made up networks of smart nodes creating a twin as a whole. Another focus incremental to our own definition were: digital representations of real world objects, but focus were on the real life objects themself [7]. In 2016 research also went into simulations of future states of systems, in other words trying to predict states and state-machines.

Finally CPS, Cyber- physical systems, gets focus by [8]. Virtual representations were in focus and Industry 4.0 is directly connected to the definition, along with CPS.

2.3.1.5. Post 2016

The work and positive sides of DT gets focus by a broader audience, today it is heavily connected to Industry 4.0 if we look at the industrial side of things. To map more work than those above after 2016 will be too broad for the target of this report, mostly due to the scale of work, both scientific research and implementations, that has been done. Instead we use Festos own sources for definition when they ask the question if their digital system, CIROS, is a digital twin or not (see 3.1 below). We also want to note that we can only find CIROS as the closest work available compared to our problem, especially connected to Festo assets.

2.4. Industrial VR/AR

According to the principles of Industry 4.0, see 2.1, factory communication is not limited to the machine to machine field only [9]. The human resources and knowledge can be further strengthened by taking advantage of wearable technology such as Virtual Reality (VR) headsets or handheld Augmented Reality (AR) powered devices like smartphones, tablets or holographic headsets. Both VR and AR are considered key components of the Industry 4.0 paradigm for training and visualisation [9].

VR transfers the operator to a virtual world that can be completely modeled to the current applications needs. With a headset the user can explore the environment in a safe way that can be used for many different scenarios such as for simulation, monitoring, interaction or educational purposes. This technology can be a great help when for example investigating future investments. The layout can be built virtually for testing and simulations which would allow for detecting flaws that could be extremely costly to fix afterwards.

The AR technology can be used to project digital content in relation to the physical world. A screen from a smartphone, tablet or smart glasses acts like a see-through tool that renders this digital content on top of what the user sees. This can be useful in industrial applications where someone might need to find information that is not naturally available on site in a factory for example. That can be assembly instructions for some part of machinery or a blueprint that can be displayed in direct relation to the part itself with interactive information. Also AR technology can be used to visualize information about the current state of machinery that is not easily available in the real world since AR won't adhere to the rules of the real world (time, space and gravity for example).

2.5. Open Platform Communications and Unified Architecture

Open Platform Communications (OPC) is a standard adopted by the industrial automation business, especially in Europe, for secure and reliable communications [10] based on the TCP protocol. The standard is maintained by the OPC Foundation and is built upon a series of specifications set by the members of the foundation, more specifically vendors, end-users & developers. Some of the specifications describe the communication between server/client or server/server, while others describe the access to real time-data/historical-data or the monitoring of changes/events in a network. In the beginning the OPC standard was restricted to the Microsoft Windows platform but has developed into a completely platform-independent standard. This is of high necessity for the further use of the standard because of the different equipment and embedded devices used in industrial applications.

OPC-UA (OPC Unified Architecture) is a protocol based on a client/server architecture. Because it is platform independent, any client can access any server regardless of the system it is based upon. A server can collect data from its underlying sensors in any way it likes and then this data is available to clients through the OPC protocol. The protocol supports the auto discovery of servers in a specified network as well as the data it supplies for a client to browse. A client can place subscriptions on particular parts of the data to be notified on specific intervals or on specified events or changes. The OPC-UA protocol leverages a wide range of security features, where some are the use of client/server authentication certificates, message signing and session encryption.

3. Related Work

Related work will present our solution in comparisons between previous work done in the same field. We will begin with Festos current systems and how they define their products compared to research in 3.1 below, the chapter then brings up and discusses other research and connects our work towards each one of them.

3.1. Festo and its VR/AR definitions.

We bring up this section under related work since Festo has two implementations themself that our work needs to be compared with, CIROS [11] and its AR-app are Festos own approach to the DT's evolution. How they compare themself towards research and which definitions their work adheres to is important to understand how to relate where our work fits in a bigger picture, especially compared to Festos own solutions that are closest to ours.

As touched above, Festo provides their own implementation towards AR functionality that allows for displaying information of the running modules of the CP-F such as actuator and sensor states. It can also be used for visual guidance in commission of the modules [12]. This targets handheld devices and is optimized for tablets. The vision for the AR discussion around our system is to overview a greater part of the CP-F at once in real time and for this we target devices that are less disruptive of the monitoring workflow, like smart glasses or the Microsoft HoloLens [13].

Festo also provides a system called CIROS (Computer Integrated Robot Simulation) [11] that focuses on teaching the CP Factory system via simulations in VR environments. CIROS has live overview and visualization of the physical hardware, but it does not augment the visualization in any way, we instead aim to improve understandability via UI elements and tweak the twin for improved understandability at a glance, even if this means we have to add elements that does not exist or change parts in the real world. Our own definition of DT below in 4.2 allows us to improve upon reality which we took advantage of.

Our development of these AR and VR features were towards the connection between our digital twin and an identical physical system where we aimed to use parameters of the physical system to improve understandability of a running system, all in real time. Festo asks the question if the CIROS system is a digital twin, short answer is both yes and no depending on the criterias, they present them as:

- Bolton et al, 2018: "A digital twin is a dynamic virtual representation of a physical object or system ... using real-time data and interaction between them" [14]
- Söderberg et al, 2017: "Using a digital copy of the physical system to perform real-time optimizations" [15]
- Bacchiega, 2017: "A digital twin is a real-time digital replica of a physical device" [16]

CIROS does not fulfill the first criteria but the other two they do, we will find below in our definition, in 4.2, that we will only fulfill the last criteria in specific ways if we only focus on the same definitions as Festo do, which is logical in our situation since we want comparisons between the solutions.

3.2. Virtual Production Line -Virtual Commissioning (J.Persson, J.Norrman 2018)

The authors present their work in the Industry 4.0 domain with a digital representation of automation hardware [17]. Through the software RobotStudio by ABB and CoDeSys, a digital twin is implemented to verify and validate PLC code, before a physical station is commissioned. The paper is related to our work as it involves OPC data communication between a virtual and a physical representation of industrial machinery. Although their work is focused on the core connectivity for testing and simulation, whereas our work aims for enhanced supervision and debugging of a Festo CPF with the help of a digital twin. Also the tools for visualizing the digital twin differs as we use Unity 3D to build our virtual world to produce a more generic solution. The authors findings are that virtual commissioning is costly both in the way of gathering proper prior knowledge and the high time consumption.

3.3. Using Unity3d as an Elevator Simulation Tool (J. Polcar et al., 2017)

The paper describes the research regarding a virtual elevator simulator [18]. The work is related to our research in the sense that the simulator model is developed using the Unity3d game engine just as the virtual simulation in this paper. Research is performed to find out if Unity3d can be suitable to create a simulation model compared to traditional Discrete event simulation tools (DES). The tool is purely a simulator and no data from a physical counterpart is captured and processed, which differs from our work in that way. The authors found that the simplicity of using Unity's object oriented programming system and the use of 3D space, compared to DES is of great advantage for specialized situations. The steeper learning curve and price range of DES tools could be a key financial factor in this decision. Although for the overall situations DES found to be the preferred simulation modeling tool.

3.4. Benchmarking of existing OPC UA implementations for Industrie 4.0-compliant digitalization solutions (H. Haskamp et al., 2017)

There are many different implementations of the OPC-UA protocol available for various programming platforms on the market. OPC-UA is considered one of the key technologies that complies to Industry 4.0 specifications. The authors of the paper are scanning the most common variants developed by corporations, foundations and private persons to compare their features and categorize them in paid vs free-of-charge groups [19]. Their findings indicate that there are comparable solutions in the two groups and some of the free implementations provide approximately the same features and documentation as the commercial ones do, or at least fulfills the criterias set up by the authors. This relates to our work because we needed to find an OPC-UA solution to develop a client in the C# language that is compatible with Unity3d. We found the “OPC UA .NET” solution provided by the OPC Foundation to be well suited to our needs.

3.5. Digital Twin Based Synchronised Control and Simulation of the Industrial Robotic Cell Using Virtual Reality (V. Kuts et al., 2019)

Digital twins can play a significant role in the automation industry in terms of simulations and training. The research [20] by Kuts et al. tries to find the benefits of using a digital twin alongside a physical robotic cell. The work aims for the possibility to not only monitor the physical counterpart in a Virtual Reality environment, but to also be able to control it by the means of the digital model. The DT was modeled in the Maya 3d software and the virtual environment set up in Unity3d. Part of the research aims to visualize movement data such as angles of the robotic arm and also some user directives are

visualized. The research concludes as a successful and viable solution which was tested by students and specialists in the robotic automation industry. The paper relates to our work because of the modeling tools used in the development process and the need to reduce model polygon density. The research also focuses on the digital twin concept in the same way as our work, although we do not find it necessary to control our physical modules through the virtual world in this stage of the research. The use of Virtual Reality is a common factor and because we strongly focus on the status and overview visualisations of our digital twin, these researches relate in regards to user interfaces in a virtual environment.

4. Problem Formulation

This chapter aims to present our research question, we will also structure the greater question into smaller segments in a roadmap manner in which the report later will answer. We also define our digital twin in this section since our problem also impacts definition of the twin, the expressions will be greatly intertwined.

4.1. Question formulation

This thesis aims to answer the following overarching question:

What are the advantages and/or disadvantages of monitoring and troubleshooting a Festo CP-Factory by means of a digital twin-driven visualization?

The more specific problem that we want to answer is if we can achieve improved overview and debugging by introducing a digital twin. We will also introduce VR and AR along with standard Desktop-solutions to the discussion for a wider perspective. Our question means we need to improve overview of the system and its data and we need to define how an optimal solution would look like and function. The Factory we want to focus on is the Festo Didactic system called Cyber Physical Factory and we aim to compare our solution to the overview system that is shipped with the product. We choose the system due to the fact it is built for research as ours and it can produce data towards generic discussions tied to Industry 4.0 and smart factories. As we will see in this reports upcoming chapters nothing will hinder our solution to be applied in a greater scale or problem, in other words a generic approach has been taken in discussions and implementations thanks to the Festo Cyber Physical System.

The visualization tag that comes with the overarching question means we have to look into and define the exact goals that is needed for improvements towards the targeted aspects. We also have to define and present how an effective solution would look like and function, as is common in DRM. This is directly tied to the pure visual and technical aspects of our solution which will be formed in the DRM-research, see 7.3. Visual aspects can be but not limited to: viewability of elements in the environment, improvements upon position of UI-elements or general clarifications and improvements. To finish we then present advantages and disadvantages compared to our result with the current solution that is shipped with Festo CP-F. This software is called MES4 which does not have a digital twin built in to the standard system that controls the hardware. The question and the discussion here is presented and used in the Results section when presenting the work, see Chapter 8 for answers and discussions around the results and its impact can be found in Chapter 9.

4.2. Extension upon Digital Twin definitions

As were evident by Festo's own definitions, which are the ones this report build upon, of a digital twin presented in 3.1 we could only achieve the definitions Bacchiega, 2017 named in our timespan and it is upon that definition we build our discussions. Compared to the other two we argue that our Digital Twin is not meant to alter or change any parameters of the Physical Twin, not even for enhancement which is the case in the other two sources presented in section 3.1. The data the DT produce could be used for improvement, but it's system should not affect the PT, this is for security reasons [3]. The primary purpose of our system is to improve readability of an active physical system, mainly for debugging and status viewing operations. Because of these aspects we needed to extend the digital twin defined by Bacchiega[16] and by doing so we could focus research and development so they were able answer our problems as described in 4.1.

This is how we describe the Digital Twin system used in this thesis research:

> "A Digital Twin is a digital-system that is connected to a physical counterpart that is meant to mimic reality in greatest possible fashion and is meant to expand upon human-machine interaction. Integration of IIot (Industrial Internet of Things) is an important factor since it allows extensive, digital, knowledge about a physical system. Without the sensors that are connected via networking, the DT would not be effective compared to the real world due to the lack of data. Real time data should be in focus connected to overview and debugging as the goal is to keep the delay between physical and digital unit to be as low as possible, this is imperative. The digital system should also look like the physical counterpart when it comes to: scale, relation between modules (such as relation in space) and other "physical effects" that interact with the system, such as gravity or lighting which will improve realism in virtual realities. With realism in mind there is nothing holding the Digital Twin back to abstract away parts of the physical system that hinder understandability tied to overview. Examples of parts can be security attachments or window-panes that are meant to create a secure work environment as well as defending the hardware. A digital twin does not need these security-measures since the operator cannot get hurt or damage the system which means visibility in the virtual space can be greatly improved upon by removal of said parts. The physical twin will always represent the real world better than any mimicking twin and to make the Digital Twin as an available option it has to evolve and we need to improve upon factors increasing productivity. Thus should the software be able to represent the real world twin as close as reality as possible, but its focus for usability should improve upon important factors where it is necessary. This means that any Digital Twin should have more than one mode, at least two: Digital Twin Mode where we aim to be as close to reality as possible which is important for education, secondly we need an Improved Mode which is imperative for overview, debugging and understandability."

As is apparent with our description we could in Digital twin mode achieve the definition of Bacchiega, 2017, see 3.1. When using Improved Mode we will lose all these similarities since the system will not, to the greatest degree, be a replica towards reality, thus we extend the definition. Nothing is hindering the implementation by our extension to allow for other modes where the operator for example could control and impact the physical system from the digital counterpart, this would allow even more and stricter features to be available to us. We did not add this to our digital twin due to the security risks this connection would bring and our timeframe would not allow the implementation.

Even though we aim to use Virtual Reality (VR) and/or Augmented Reality (AR), we don't find that directly related to our description. Theoretically a 2D program tied to Desktop solutions can work under the same description as the one we plan to use, but work in VR would mean that the user could move around and monitor the system as one could in the real world. AR is where the user could use a handheld device to get more information about parts of the system, it would basically act like an overlay to the physical object when viewed through the device that could improve understandability of a live system. It would also allow for quick verification of system status to limit human introduced errors such as misplaced parts.

5. Method

This section aims to present how we tackled the problem via DRM and gives a short explanation of its methodology and what factors we used to define success or failure.

5.1. Festo Cyber Physical Factory

To answer our questions we used the Festo Cyber Physical Factory (CP-F), a training and research station with the capacity for real solutions, located at MITC in Eskilstuna. The CP-F is a module based system built with various units that are connected to each other. Each unit/module has its own state that can be monitored and data can be received from it via communications. This backend is the data we visualized for the user in a VR environment in relation to a digital version of the CP-F system provided. The virtual world was built in the Unity3d game engine that has extensive support for VR/AR development where our foremost focus was on VR but Unity3d enables quick and easy tools to convert these solutions to AR as well which enables a richer discussion. Previous literature was studied regarding VR and industrial implementations and how we could interact in these types of applications such as but not limited to: [21],[22],[23]. This work provided us with a good starting point in our research and we could define what we needed to focus upon tied to success parameters.

5.2. DRM

Our research was structured according to the Design Research Methodology (DRM) that suits this type of interaction design well [24, p. 112] as the research drives parameters as they surface via assumptions, experience and evaluations we focused these parameters upon optimal solutions based on overview and debugging. DRM is an iterative process and we needed to evaluate each step and then go back to reiterate the whole process and improve upon our previous solution where it was necessary, which means during all these iterations new functionality was added or revisited for evaluation according to our findings. The DRM emphasizes four steps to work towards a research result [25] that can be overviewed in [Figure 3].

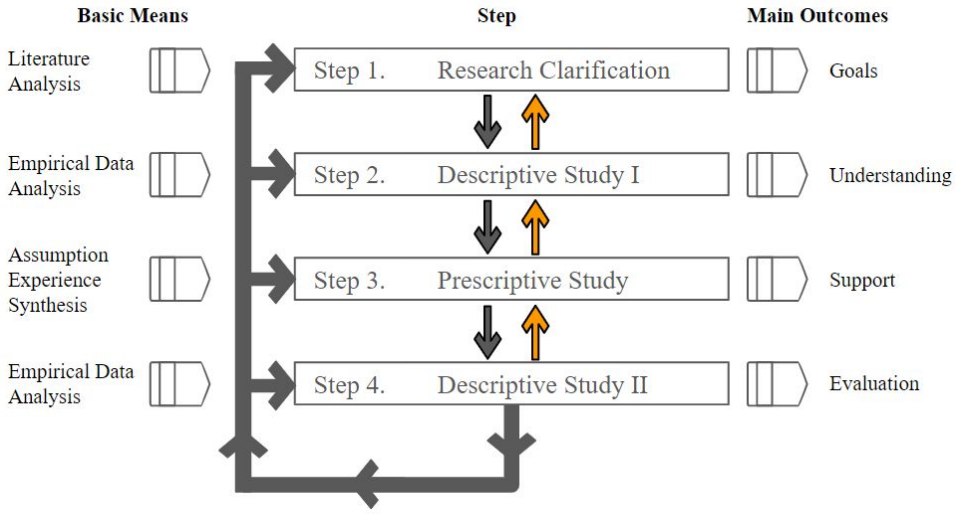

Figure 3, The DRM research process

5.2.1. Criteria definition

Goal:

- Identify the aim that the research is expected to fulfill, and the project focus.

This means we first needed to map what we aimed to achieve by the DRM-research. Since our question aimed to focus upon overview and debugging we defined the goals with these two aspects in mind, see 7.3.1 for the targeted parameters as we defined them. Since our solution was meant to be compared to the existing desktop application along with the HMI-screens (Human Machine Interfaces) that are attached to each unit/module we also had to define problems with our definitions in mind versus the current solution, see 7.3.2.1 for this process.

In each iteration we got more information about the systems, problems and what we needed to focus on which is the proper strength of DRM as we could begin early and by previous faults correct and evolve the solution and our targeted parameters in 7.3.1 and focus on design-parameters as defined in 5.2.3 below. This is the power of DRM since we can shift focus according to experience and the best practical solution as we do the research [25].

5.2.2. Descriptive Study I

Goals:

- Find factors that contribute to or prohibit a successful solution as defined in 7.3.1 and 5.2.3.
- Find details that can be used to evaluate design

Each of the first descriptive studies in each iteration took the previous iterations second descriptive study and tried to focus on improvement from the data produced. The very first iteration focused on the current solution shipped with the Festo CP-F and tried to define what was a success and what was a failure within the system tied to our initial criteria, see 7.3.1. This first step inspected the desktop solution (MES4) along with the HMI-interfaces that exist on each module.

5.2.3. Prescriptive Study

Goals:

- To develop a model or theory, based on the reference model or theory from the descriptive study stage, describing the expected improved situation.
- To develop prototype in a systematic way (Proof of concept)

The prescriptive studies in each iteration aimed to implement and try the improvements upon the data found in the first descriptive study. In other words the prescriptive phase is where the implementation and/or mockup work got done which in turn delivered valuable insights towards the second descriptive study that broke down the implementation studying success or failure, note that each of the parameters from the implementation we looked upon are the following:

1. Is the solution easy to use?
    a. Here we focused on feed forward [26, p. 126] (Does the operator need to be trained in the software to use it?)
    b. How easy it is to get a complete picture in one go? (Maybe there are multiple tabs with relevant information which might be a bad or good thing)
2. How clear does it handle errors?
    a. In what way does the solution flag errors?
    b. Are they intuitive?
    c. Are they easy to miss? (If operator has focus elsewhere)
    d. How do they look and can we draw any conclusions from that?
    e. Is it clear how can we recover from the errors?

3. How can the operator get information?
    a. How quick/easy is it to navigate to the relevant information?
    b. Are there any problems with these navigations?
        i. Such as the operator have to move in physical space to get the information
        ii. Not easy to use or great hierarchies to reach information
        iii. Not clear what everything is
4. Does the solution design (visuals) follow and improve upon the previous categories in this list?
    a. Color schemes
        i. Contrasts
        ii. Are they easy to see in the environment?
    b. Orientation/transformation of menus and UIs
    c. Are the relevant information grouped in intuitive ways?
5. What do we gain/lose from VR compared to desktop each iteration?
    a. Does VR hinder the workflow or improve upon it?

5.2.4. Descriptive Study II

Goals:

- Find out if the prototype can be used for the intended situation(evaluation)
- Is each implementation a success compared to the focused design questions in 5.2.3?
- Are there any side effects present?

These stages are similar to the first descriptive study, but instead they focus on the improvements made in current iteration from the prescriptive study. The questions were the same as we used for development and we could use the data produced to check where problems still existed in each iteration. The deliverable at this stage was then used as data in the first descriptive study in the next iteration, unless we were finished with the research. Our study had five of these DRM iterations before we felt done with the result and had a strong proof of concept.

5.3. Research

All work needed to be grounded on current scientific data which also meant we needed to review and read up-to-date sources that touch improved design. The peer-reviewed sources we studied are mainly interaction design sources connected to VR and previous digital twin definitions. We also wanted to look at the human interacting with the system and how we can create applications that are better emotionally, we aimed to use [27] among others that discuss emotions connected to design-choices but there was no time to include these into the thesis and we recommend future work to look into these parameters as we believe them to be beneficial. What we mean with improved design and interactivity here is a better overview that could help prevent hazards (very clear warnings for example), new ways to interact with the hardware (for example focus on a small part of a greater system, abstracting all other information) and better feedforward [26] for new users [28],[29] together with an experience that feels natural. We also needed to investigate the OPC-UA protocol where the paper *"Communication Technology for Industry 4.0"* [30] was useful to find more information and to define the metrics of the protocol compared to other popular communication protocols such as ROS, DDS and MQTT [30],[31].

There was a need for studying how certain areas of visual feedback are implemented for industrial machinery, in particular for the Festo CPF. We needed to review the solutions given for visual information like monitor positioning and content as well as light indicators with its colors and positions that the physical system at MITC is equipped with. Then we needed to compare our own result by listing what is better and/or what is worse with our implementation, all discussions grounded in current research with focus on our question: *"To what degree could VR and/or AR connected to a digital twin visualize data to improve supervision and debugging of a Festo CP Factory in real time?"*

6. Ethical and Societal Considerations

This is a brief section where we discuss some of the ethical effects our work could produce as we completed the research.

6.1. Engineers honor codex

If we focus on the Engineers Honor Codex [24, p. 20] we can't see that we, our work or its result could be in violation of these ethical guidelines. The closest to problems we could be is if this research would harm any of the corporations behind the technologies that we aimed to use such as MITC, Unity Technologies, Festo or Microsoft, but we find this highly improbable as our work finds the complete opposite to be more likely.

6.2. Industry 4.0 and its ethics

Our application aims to improve upon Industry 4.0 which is introducing some ethical problems. Since I4.0 could be replacing personnel-labour with robotics and digital solutions this work could have some catastrophic consequences if this were to come true. But with that said we argue that our section of I4.0 is made to improve upon the workflow for operators by increasing human-machine connections which should lead to that industry won't need to replace them, but integrate humans with new digitized solutions that could change the possibilities towards a role of supervision and monitoring [9]. Thus we argue that our work and technical progress could be beneficial for the future industrial worker where jobs can continue, not become a replacement factor for organizations.

7. Description of the work

We had 3 major phases in the workflow, first of we mapped the systems, this proved to be the greatest hurdle and it took most of our time. Secondly we had the DRM research phase and lastly the conclusions and result phase.

7.1. Selecting the data protocol

The modules of the Festo CP-F expose a lot of data about the states they are currently in. We needed this data to be transferred from the modules to our system to have a correct representation in the virtual world. Because there are a lot of different data transfer protocols such as OPC-UA, ROS, DDS [30],[31] and MQTT that are used in industrial communications, we needed to decide which one was best suited for our system. We found the answer would be the OPC-UA protocol, mainly because of the current system in place. To completely use another protocol would take too long to implement and bridge, but this was not a bad thing: The OPC-UA protocol is considered one of the cornerstones of Industry 4.0 [31] and because of this the Festo CP-F has extensive support for it. As this is an open protocol there are open source libraries available for us to take advantage of in our implementation [32], although it had to be tweaked a bit to work in Unity3d. Previous research [19] by *H. Haskamp et al*. guided us in the right direction to find suitable tools for developing the data communications part of our software targeting the .NET framework supported by Unity3d. All Festo CP-F modules have OPC-UA servers integrated into their Programmable Logic Controller (PLC) that we could connect to directly with our software which acts as a client to receive available sensor data. This reduced the complexity of our program because with other protocols we would have been forced to develop some kind of intermediate bridge to handle the data we needed to get a hold of. The clients (each module of our Digital Twin) signs up for subscriptions of every sensor on the server that provides data. Upon changes in the server the client gets notified via async protocols, and with the incoming data-stream we could incorporate the changes into our solution in any way we wanted.

7.2. Mapping the Physical Twin

The mapping of the physical twin proved to be more extensive than we initially hoped. There is no extensive documentation over the MITC-specific implementation so we had to dissect every part of each module and its server nodes, for example does the ASRS32 contain nearly 2000 nodes. Our first thought was to subscribe to all nodes and then build a parser logging the important data. This would make saving/reading the data easier, see 7.2.4, and we could just throw away the unneeded data, but as one might understand this was not a good approach. Due to the extreme amount of data that is produced the client won't get all data since both server and client are flooded (read/write in Unity must be done between frames). As an example, the ASRS32 produced 21 362 rows of data in a timespan of 14 minutes where we only needed a very small portion of the data and a lot of information was missing due to these bottlenecks as it needed to collect and write all data between engine ticks. The problem is also in the first few moments when we subscribe where the server will send the complete status of the system and this proved to be extensive and expensive. The reason we tried a “log everything” approach is to mitigate problems introduced when the user wants to produce the logs, they would have to subscribe to very specific nodes nested in a complex hierarchy and the work would quickly grow tedious, so a “log everything” approach would be easier for the user, but as we learned did not work.

Due to the fact of the extensive nodes count we had to focus on fewer and relevant nodes and make sure to build our own OPC-client in the Unity-application. This would be beneficial since it could subscribe automatically to the correct nodes. This approach also lifted all work from the user when it comes to connecting and saving the logs, but this would be at the cost of maintainability and extensibility: to expand functionality the user would have to extend the source code. To mitigate this problem we landed in using a settings file with all paths and node names collected, see 7.2.5 below for in depth explanations. The program will read these lines from a text file and parse them as subscriptions in our software. This approach was the best available and we did not need to map everything and the operator can add functionality as needed, outside of the source code. Note that the software will show the raw data stream and its values, no logic will respond to this data if it was to be added in this fashion. For that to work the source code needs to be altered, logically, see 11.5 for discussions around this problem and how to mitigate the problem. The settings file also enabled the software to be more dynamic in the other spectrum, imagine that the operator wants to abstract away unneeded nodes, they will just have to comment out a line in the document, and the system will not subscribe to the node.

Below we will look closer to the modules we chose to capture data from, the factory assets called: CP-F-ASRS32, CP-F-BRANCH and CP-F-RASS (Robot Assembly), see [Figure 4]. The nodes were first mapped and overviewed by an OPC-Client program called Prosys OPC UA Browser [33]. Since Festo could not provide up to date documentation of the MITC-Festo implementation we had to find the nodes we wanted manually by subscribing to them and view their data in real time. We then had the CP-F running and compared the subscribed values to the actions produced by the physical factory, with this method we could map values and it was of course not optimal. We should also note that on some parts of the physical factory there are labels of sensors that sometimes could be mapped to the OPC server which somewhat speed up our process along with Festo general documentation [34]. Sadly this documentation in some cases did not help us with logic or was wrong compared to the implementations. The hardest part was to find the error handling done by the modules, which was missing from documentation completely and we naturally wanted that data because of our softwares purpose, we had to manually find them. Most nodes work with string values (that can represent the normal data types such as booleans, floats or integers) and each error and sensor has its own node in the server. We can not to our current knowledge understand what all the different nodes, such as all errors, represent exactly but sometimes the naming and namespace of the node gives us a hint. We still present every error, but the more obscure ones will flag with its node name instead of having specific logic attached because of this documentation problem.

To sum the OPC-logic this means each error and all other data-nodes that have logic in the backend have to be manually subscribed to. This turned out to be a lengthy process and we can't always give a string output with an explanation to the user at this moment in time, but as an approximation we have mapped and created specific output for 90% of the subscribed data and to increase that number we will need an expert on the PLC logic in the MITC implementation, something we sadly not had time for and it is thus tagged as future work.

Also note that all software shipped with the solution such as MES4 and each module Human Machine Interface(HMI) will be presented in the DRM section of this report, see section 7.3.2 and not in the mapping of each unit below. The upcoming sections will also shortly explain the purpose of each module and how they work.

7.2.1. Mapping CP-F-ASRS32

ASRS32 is the storage part of the MITC-factory, it handles all pallets, see [Figure 59], and stores them in a hierarchical fashion. It has two loading stations where carrier ID and other data is read by an RFID sensor, the second of these stations will be triggered when an order is queued by the user. A carrier will stop and the arm of the unit will lift down one of the specified pallets (decided by the production-order from the MES4 software). When done the carrier will be released and transported to the BRANCH-module via the conveyor belt. On the first station the module will, with the help of the arm, lift back an empty pallet (or finished product if defined by the order instructions) when it arrives back after the completed production flow.

We chose to map the two loading stations and if there is a carrier present at them. These are composed of two nodes and they will also feed which carrier ID we have on these stations. We also mapped the nodes *"diONo"*, *"diPNo"*, *"iOPos"* and *"iOpNo"*. These will update with information when a station is loading/unloading a carrier and if a carrier has an order attached. "*iCarrierID*" is reporting which carrier is present at the station. The OPC-server has different node sections (can be compared to folders in an file system hierarchy) and as an example we use "*dbTransport*" to know in which direction the belts are moving, "*dbRfidData*" has the carrier data explained above and "*dbError*" has some (far from all) error-data that can affect the module. The Unit will also measure all raw sensors that are in the machine such as data to present what the gripper arm is doing, for example fetching a pallet. This is where the Festo documentation was of great help, we could use their own definitions of these nodes and add an explanatory text on the Machine UI (a feed showing what the machine is doing, all modules we mapped have them) thus increasing understanding of what the operator sees.

The module also has "*dbAppCtrl*"-specific data and in here we can find how the storage situation look like, but there are some underlying logic hindering these nodes from updating unless the physical system has a specific menu open (probably OPC functions), thus will our software read other lines instead of these nodes from the settings file defining what is present in storage from start, just as proof of concept. This should also be future work to dissect this problem and feed the data to our software dynamically.

7.2.2. Mapping CP-F-BRANCH

The CP-F-BRANCH is the module that handles input and output to the collective modules seen in [Figure 4]. It also has a CP-AM-MEASURE attachment that via two lasers checks if each carrier loaded with pallets have them at a correct distance (depth) and raises an error if the values are outside of the correct scope, this error is only visible via a light pillar on the unit presenting a green/yellow or red light depending on success. It also handles transport between the ASRS32 and the RASS along with output/input to and from the Robotino robot (CP-MR-C) which handles transport between the two Factory sections.

We chose to ignore the raw laser reading (which otherwise would be interesting) because it produced too much constant data and it flooded the OPC-connection, especially our logging. We decided to abstract this away in favor of a system that is easier to run and get files that are easier to read, of course there are workarounds to this problem but we instead monitor: if a pallet is being measured on the station, if we have any error on the machine and if something is present in some of the different stations. The stations mentioned are Input or Output (the robotino connection) along with the Laser station and a output station from CP-F-RASS. We should also note that the underlying OPC system of this unit is completely different from the other two and this produced some problems when we were trying to map and understand the underlying structures, but we could still find some of the raw sensors and use Festos definition of them as readable output to the user.

7.2.3. Mapping CP-F-RASS

The Robot Assembly (RASS) module has a Mitsubishi robotic arm attachment [Figure 5] and when a carrier with an order arrives into the machine it will deflect the carrier to the assembly conveyor (otherwise it will just pass straight) and in current configuration the arm will grab the back piece of a mobile phone that is stored on the pallet. It will then place the back piece on a lightboard for measurement and assembly [Figure 6]. The robot will then switch heads (the attachment mounted on the end of the arm, basically it's gripping tool) and mount a PCB-board on top of the back-piece laying in the assembly station. The arm will again switch heads and then install two fuses on top of the PCB-board now mounted to the product. When done it will switch head again and put the assembled pieces back to its carrier waiting in the station. The carrier will then be released and move back towards the BRANCH, which in turn will put it on the output conveyor of that module (ready for the robotino transport) or back towards storage in the ASRS32, depending on specific order logic.

As this is a very complicated module we did not have time to map and overview all of its nodes and it seems the Robot specific OPC-nodes are not active as described by Festo so future work will have to be done to increase the dataoutput. We eventually managed to read if the Robot is working on something, errors that could be produced (even from the arm) and where the carriers are along with other general data as mapped and explained by the other two, previously described, modules.

7.2.4. Logging the input

Since we chose to build our own OPC-UA client and subscribe to all relevant nodes we had an easy opportunity to log all incoming data to files. We think this is a good practice since the user can go back to runs that are not running live as well and try to debug sessions that might have had problems, this is commonly true for most real implementations tied to productivity. This also made our work slightly easier since we didn't need to be at MITC as soon as we had to run and develop the software, we could even simulate problems or logic that never actually happened and it can be used as a tool for teaching when it comes to different problems. We would have liked to have some tool in the software with pause, play, stop, rewind and move forward capabilities when replaying a stored file, but our timetable was too limited to implement this, but it is a good tool to keep in mind for future work in 11.2.

The logging capabilities also presented us with one of the greatest possibilities of the software we built, imagine the physical system breaking down and the software has logged what has happened in real time as OPC-compatible data. The operator could still read the logs produced manually as is common in today's industry, but our software can actually replay the actions of the machine in a visual representation with the same logs. The operator can literally walk around the machine in VR and see what is happening. This is a really powerful way to understand historical/saved data and it might even be groundbreaking. Imagine when parsing information this way we could also as an example show what has happened to the company managers or “investigators” after a catastrophic breakdown. Since many of these individuals might not have specific machine understandability we could be bridging

understandability between different levels of knowledge and experts to an ease by showing what has happened as close to reality as possible.

If we focus on the logging itself it is saved in the CSV-file format by using an external library called "CsvHelper" and a NodeData class implemented in the software to automatically append data to the CSV files. The CSV structure can be seen in [Figure 7] below and as said, as soon as an OPC event is triggered, the software will append its data and as well send it into the input stream of each FactoryModule.cs script. Since we then get all the information and specifically its name, source, value and timestamp we can later recreate the run by using the CSV file as an input stream instead of the OPC live stream, we just have to recreate the events and make sure the timeline follow the different CSV files as it was done in the real-time run. There is a CSV file for each module, mainly because of the possibilities of loads of data so it is easier to understand what each module is doing. To read or write OPC data the user can change the settings file for easy configurations, see 7.2.5.

7.2.5. The settings file

We decided to implement a file where the operator can set different parameters to make the software more dynamic. This is an important part when it comes to portability, modification and expandability of the software. In the Data folder we put an ordinary .txt file and it is read by Settings.cs script in the Unity3d scene, it loads the text-file and iterates the data stream into a switch-statement. As of now the user will have to modify this document to change program behaviour, but future work to create a menu in the software (some settings already exist, but no pre launch settings) or an external program would be beneficial, see 11.5 for further discussions around this. The input will ignore lines that start with '#' thus disabling that line (comment). First lines have the following fields as examples of file structure:

```
#PREVIEW EXAMPLE START HERE

#                                   #### Software Settings ####
#                             Should the software connect to a VR Device
UseVR=False
#                             Should the software look for and use OPC Servers as input stream
UseOPC=False
#                              Should the software look for and use CSV files as input stream
UseCSV=True
#                             Should the software Log the OPC data (not recommended to use with
UseCSV=True)
LogCSV=False
#                             The animated arrows on the belts, should they move?
UseArrowAnimations=True
#                             Should the Modules unnecessary(Top/upper elements) parts be removed from the start?
AbstractMachineTops=True

#                                   #### VR Logic ####
ShowControllers=True
UserSpeedVR=1,1
RayLength=5,1



#                                   #### Keyboard/Mouse Logic ####
UserSpeed=7,1
MouseSpeed=2,393206
MouseSmooth=2,1
.
.
.#PREVIEW EXAMPLE END HERE
```

By using this logic we can parse information in a way that makes it possible to change logic outside of the game engine and source code, this enables our solution to handle more than one solution since we

for example can start in VR or Desktop mode, only one application is needed to be installed. Other important elements are the logic on each factory module such as the Branch settings as seen below:

```
#    #### BRANCH ####
ACTIVE-CP-F-BRANCH=True
UI-CP-F-BRANCH=True
ADDRESS-CP-F-BRANCH=opc.tcp://127.0.0.1:REDACTED
DELAY-CP-F-BRANCH=400
NAMESPACE-CP-F-BRANCH=4
.
.
.
# END OF EXAMPLE
```

First we can set if the UI should be active at start or even if the unit should be active. We also have the OPC connection ip, this is important if network changes are in effect. The delay between each update in the OPC protocol is also changeable from here. Namespace is also important for dynamic OPC logic, it is the main hierarchy in where we should find the nodes (much like which “hard-drive” to open). The user can comment out the ADDRESS fields, leave that field empty or type NULL for clarity to disable OPC logic completely if a whole module or OPC target is unneeded in any given situation.

Another important field of the settings file are the OPC-nodes that each unit should subscribe to, below follow an example of the first rows the ASRS32 will connect to:

```
#                       ####ASRS32 DATA ####
ASRS32
#"xK1_F2_KF1"#                      FALSE = Emergency stop button pushed (does not seem to be used practically)
#"xU1_F2_KF3"#                      TRUE = Safety doors closed (does not seem to be used practically)
#"xS_F2_FQ1"#                       TRUE = Emergency stop button pushed (does not seem to be used practically)
"xK1_MB20"#                         TRUE = Move stopper 1 down
"xK1_MB30"#                         TRUE = Move stopper 2 down
"xG1_BG36"#                         TRUE = Carrier detected at belt 2 entry
"xG1_BG37"#                         TRUE = Carrier detected at belt 2 exit
"xG1_BG20"#                         TRUE = Stopper 1 in lower end position
"xG1_BG21"#                         TRUE = Stopper 1 carrier ident code bit 0 detected
"xG1_BG22"#                         TRUE = Stopper 1 carrier ident code bit 1 detected
"dbRfidCntr"."ID1"."xBusy"#
"dbRfidCntr"."ID1"."iLen"#
"dbRfidData"."ID1"."iCarrierID"#    INT = Station 1: Carrier ID
.
.
.
#END OF EXAMPLE
```

By this method we let the user/operator have the possibility to abstract away and add logic as needed. As seen above we have commented out the three first lines because they might be unnecessary for our logic or not even active in the Festo OPC server at all. Because of these segments the solution is even more dynamic and it should be easier to port to other OPC servers, most nodes also have a comment explaining what it is and what it does.

7.3. Research phase

When we had implemented the main logic needed for the various systems and modules we started the DRM research phase. First we needed to define the criteria to work towards and we learned from some problems when we built the backend of the software, also note that the design criterias mapped in 5.2.3 will be used in upcoming development as targeted aspects in each iteration and its implementations.

Since DRM-is design driven will images/figures in this section below be important for the discussion and its analysis. Because of this there will be plenty of figures with detailed explanations in appendices as they drive and impact the discussion, it is thus recommended to have another version/copy open of the appendix section on the side at all times for quick referencing towards the figure referenced in the text.

7.3.1. DRM: Initial Criteria and mappings

The solution needed to be effective when we focused on overview and debugging along with other important parameters as defined by the extended DT-description in 4.2. This gave us some initial technical criterias that we aimed to improve upon:

1. Software need to be realtime
    a. This means we have to build a solution that is not too complex when it comes to the architectural sides of the software. There might be too much logic which would result in the system not being able to handle incoming data in a satisfactory way.

2. The solution needs to be easy to use.
    a. We do not want the user to have any other knowledge than what they already need to know about the physical system. The digital twin should be easy to meaning that no information about how to operate our software should be present in the solution itself.
    b. The system is meant to complement, even improve upon, reality, and a software that is complex to use might not live up to that criteria, meaning the industry will keep old solutions instead of allowing progress.

3. It should be easy to see, find and understand problems as they arise and other UI-elements as described in 5.2.3.
    a. Intuitive color schemes.
    b. Relevant positioning of informal elements.
    c. Locations, reason and information should be intuitive.

4. Overviewing should be easy to do simultaneously without missing any relevant information because of the fact that focus might be elsewhere.

5. System should be able to abstract away unimportant information, but it should still be available if it is necessary for understandability or a complete picture according to the expanded definition in 4.2.
    a. This could be both physical counterparts in the virtual world and information from the machine. Physical counterparts can be security attachments on hardware meant to increase safety for the operator, in the virtual world these obstacles can be removed for improved viewability since the risk is removed.
    b. It should be easy to switch between these modes and each of the modules should have its own configuration.

6. Our solution should also be able to be ported to different systems, this does not only mean Festo systems but also other factory solutions, targeted hardware and functionality-modes.
    a. This is generally internal workings and tied towards backend logic, but it might affect UI functionality and how it is presented. We mainly wanted to create a universal solution that could be used everywhere no matter the physical system.
    b. Solution should also be portable between Desktop,VR and AR modes in an intuitive manner which means all elements need to translate between different mediums in a logical manner.

7.3.2. DRM: Iteration One, the metrics of the original solutions

In the first iteration we looked at the existing solutions and thought about how we could improve the design, this helped in creating the initial criterias as seen above in 7.3.1. Success will be measured by each iteration and what factors we look at. We used the definitions above and the factors mapped in 5.2.3 to find what was deemed as successful elements and problematic areas that we could improve upon.

7.3.2.1. Descriptive Study I: The existing solutions.

The first DRM iteration overviewed and analyzed the existing solution of the Festo CPF, this is mainly the software and solutions we needed to improve upon and in the end this comparison defined if our work was successful or not, see Chapter 8 for the results. There were three parts/sections of the physical system and its current software we had to focus upon: First and foremost is the desktop software that controls the modules (7.3.2.1.1). The software is called MES4 and its job is to overview, debug, customize logic, tweak functionality and produce orders. The second section is also part of the MES4 system (7.3.2.1.2), but it is instead tied to the different module's own interface (HMI's). Each module has a Siemens touch-screen attached for interaction and overview. The third section is how the physical twin looks in the physical world and what information we can get from it. This can be lights flashing when something is wrong, sounds, locations of physical assets and simple overview by looking at the system, these parameters will be presented as the iterations move on and have no specific section in the report.

7.3.2.1.1. MES4 - The Desktop software

The MES4 software [36] is shipped with the Festo CP-F system and it handles all input and output towards the physical modules. It acts and works as most software used in these types of systems and the user uses it to set up the modules and produce orders as well as error handling. The focus for us will be on error handling and how the software can present data in real time. Festo themself present the system as:

> "MES4 is a specially prepared manufacturing execution system (MES) with a new design for Industry 4.0 learning platforms. In MES4, orders can be started or finished at every station.
>
> The database is open and can be written to and read from via SQL commands by external programs (e.g. order entry from ERP system). Work instructions for manual workstations can be created or adapted at any time. The individual controllers communicate with the MES4 via TCP/IP." [36]

As a side note SQL-logic is interesting since with these SQL queries via TCP/IP our software should be able to also control the modules, but this is out of scope for this work and it would expand the description of our twin. But as future work it could be beneficial, if the security aspects are acceptable [3] as seen in closed networks for example.

If we focus upon the visuals in the current solution we can in [Figure 8] see the main screen and see connections to each module, their IP-endpoints, what mode they are in etcetera. The problem is that there are no warnings other than the color-codes when something goes wrong with the physical system. [Figure 9] presents the watch-tables that is used to see the raw data of the backen. This is one of the most useful screens when we overview the technical data. One can also see the logic from the OPC protocol. The problem is viewability, no complete picture can be given, no historical data is accessible in this mode and some elements are not even updated (blue icon under value-tab). As values change they will just flip values and no other notification occurs. This is what we aim to improve upon: the readability and overview of this data. Note that all comments are empty, we aim to bring explanations of these sensors as well. [Figure 10] build upon previous figure and shows one of many sections of the dropdown menus that are in the RFID1 section at the moment. As previously stated, a complete picture

is impossible to achieve when overviewing the modules since no feed exists at all, especially since this is only one of multiple modules.

MES4 has support for outputting data tied to uptime and productivity as seen in [Figure 11]. The view is presenting uptime, idleness, error and working timetables by pie charts. We will not add these types of overviews because of abstractions, but this is all data MES4 gives when it comes to productivity. But this is something we should be able to improve upon in future work as the impact together with a DT could be substantial. [Figure 12] might show one of the most important screens when we focus on debugging and overview. This is how MES4 shows and presents errors, the explanation text where it tells the operator how to solve the problem of the door error is descriptive, but the X-Axis warning as seen in the picture has no such information, only that it should be fixed. We have no location data at all so we know where to to look, not even colours to improve understandability. Also note when this screen is active, viewability of other modules are suppressed and limited, but multiple computer screens might improve upon this where GUI-windows can be dragged out.

To sum this section we can see that the MES4 desktop-system is quite complex and unforgiving if we focus on overview and understandability, but also note that this aspect also makes abstraction and focus stronger. The user can focus on what is important as of now and this is a strength that is hard to achieve when we want it all in one place. We saw that abstractions should be available in our software to try to bring this strength to our solution as well. Also note that all menus are in a complex hierarchy (hard to find) and just to load the watch-tables [Figure 9] or the Error menus above [Figure 12] the user needs to load a new web-based interface each time, which is not optimal. See [Figure 56] until [Figure 59] in the Appendices for further figures upon the desktop solution and its analysis.

7.3.2.1.2. Factory attached Human-Machine Interfaces

Festo also presents the current solution with a HMI-screen (Human Machine Interface) attached to each module. It's job is to handle some important input towards each module, such as clearing errors and settings of the module. We will abstract the input part away from our evaluations since we only focus on overview and debugging in our report due to previous definitions. Below we present pictures of each relevant UI tied to all of the different modules we mapped, this work will focus on the ASRS32, the CP-F-Branch and the RASS unit as they seem to be the most extensive units. Under each picture we will comment what we initially looked at.

As seen in [Figure 13] we have the ASRS32 settings for the gripper arm (it loads/unloads carriers with pallets from the storage), we also get some information about sensor BG53 (it is showing true). But no further information on what it is, what it does and so on. This is a problem and it seems to be a theme in most of these menus. Thus the user needs to be educated on the system before he/she can use this information. Another factor is that to reach this information the user has to first press setup mode (blue tab on top), this is hindering the overall overview. We can, on all of these screens, in the top left see the name of the unit. To the top right we can see system time and date and if the unit is working (Automatic mode) and if it is connected to the software (MES4) controlling the units (MES Mode), this also applies to all screens.

The storage menu [Figure 14] has cards with each index of storage position to the right. In the middle top of the card is the code showing what is stored, 25 means empty, 210 is a black front-cover where a 510 is a red-front cover for example. This is a menu that needs to be open if we want to update this information via the OPC nodes, which is a technical problem. Again the user needs education to understand all these numbers. The screen also has another tab in the bottom if we want to show the back-row of the storage which means complete overview is impossible. The belt overview in [Figure 15] is probably one of the best screens when it comes to live information, we can clearly see the sensors and their position tells us what they are in the picture (they will be green when the sensors read a pallet). We can also see the belt movement direction, but a problem is that regarding the picture we don't know what is forward/reversed (left and right are actually wrong here). Thus it is easier to

physically look at the module to understand these variables, which can be a problem if we only have digital access.

The Stopper menus shown in [Figure 16] are quite extensive with a lot of information. The menus will show data when a carrier is present at the stopper and information about its payload. This is mainly data that is appended when an order is tied to the carrier. We can also see the sensor names here, but a lot of them are unknown to what they do. Again, the user needs extensive knowledge, and as this menu might be one of the most informative, the menu hides all other data and only one of the two stations can be viewed at the same time. Another factor here is that we cannot see any history of carriers at all, this is only live data and unless the arm is doing work the carriers are only present under one second which means information is easy to miss.

[Figure 17] is also the first menu we show of how errors or warnings are presented on the module, the red text on top tells the problem and will be present in any screen that the user happens to have active. By clicking the top that shows the problem the menu shown in [Figure 17] will be presented. It has a list of current errors and clicking list items makes it enables the user to clear them (if the reason for the problem is solved). A red message here will result in an error and the module is stopped (such as emergency stop triggered for example). Since this application is using standard colours for presenting: Error = Red, Warning = Yellow and Working = Green, we mimicked this in the software for understandability, consistency and to reduce relearning and faulty reads.

The greatest problem with the physical assets is that the errors only show up as a red line on top of the HMI's, no light or other information in the real world (except for the CP-F-Branch and Robotino that has a light-pillar mounted on top showing red, yellow or green light depending on working mode). The only other physical alert the user will get is the sound the module produces when it stops, not a signal, just the sound of machinery shutting down. The only sound for errors and warnings in the real environment is that the RASS unit will beep repeatedly, but only if the emergency stop is pushed, all other machinery is silent. Sounds like these are easy to miss in a noisy environment and if it was to be louder it could even be a work-hazard. We aim to improve upon this understandability and overhaul how errors are presented to the operator, this is too important information to be displayed in this fashion and it is really easy to miss.

If we instead focus on the CP-F-Branch we first look at the mounted attachment, the laser measuring unit [Figure 18]. It stops carriers when they have a payload and via two lasers measure if the pallet has products placed the correct way. It shows us the allowed differences in measurements and also the latest measured element. This is one of the few menus that show history data in any fashion. The two gauges point to current data, this is constantly reading which proved to be hard for us to map in a discrete fashion. Constantly changing values are a problem if we want an effective solution, thus our software won't show the lasers raw data over OPC in real time.

The RASS (Robot Assembly) takes a carrier and loads the payload onto a light-table where it will assemble a PCB-board and two fuses before loading it back on the waiting carrier. This menu [Figure 19] is another one of the few with some historical data. ST 2 is the station that is under the robot arm and here we can see the latest data read: what carrier it was that had the payload, order number, product number and other information, this will be overwritten if a new order arrives at the arm. This might also be one of the more pedagogical menus as most information is self explanatory with clear pictures for information, at least tied to the picture when we overview stations. But again, it hides a lot of other relevant information, mainly what the robot is doing. The RASS arm in [Figure 20] is probably the most relevant information on the unit itself, thus it has its own UI-menu as well. There is a lot of information to gain here but it isn't necessarily explained in a good way, again the operator will need education. Also note that the screen has a lot of information in a small area, which is hindering better information and looks a bit cluttered. We are also missing some sort of feed that shows what the robot is doing at the moment.

7.3.2.1.3. General comments about MES4

As we can see with the pictures tied to the first descriptive section we had some thoughts and effects that we could focus on. First problem we saw was the overview of most information, we could never get the bigger picture at a glance. On the physical modules only a small number of information can be shown at the same time which for overview is not a good attribute, especially as we needed to move in physical space. Also note that this negative aspect from our perspective also is working well if we want to abstract away parts of the data. We wanted to mimic this abstraction in some ways without losing overview as we understood why it was developed in this fashion. We also set out to mimic colour schemes when it comes to warning, working or error logic for familiarity. History was a problem and understandability of what the system has done previously is in many parts missing when viewing the physical machine and its different menus. We wanted to create some sort of feed so we could store interesting history over the system. We also needed to get information from the physical world where it was possible, for example: storage, what the arms are doing, orientation of belts and deflectors, what the carriers are doing such as where they are and what they have loaded (if any) as examples. See [Figure 60] until [Figure 63] in the Appendices for further figures upon the HMI-solution and its analysis.

7.3.2.2. Prescriptive Study

We started to implement some of the activities from the information gained in the previous step and we started to focus upon the ASRS32 module first for proof of concept and narrowing down our initial workload.

As we can see in [Figure 21] we implemented a UI that is as big as the front of the ASRS32 unit, we could cover the whole unit in a big UI. We also implemented a feed (middle right, empty in picture). This is meant to show some historical data upon the unit, we used the sensor name and its value for quick information at a glance. In the bottom right (green) we can see if the belts are active (moving) or not, red means stop or inactive while green is active. The two red images at the bottom show if a carrier is present at corresponding stations at any moment in time, we did this so the operator could see from far away in the environment if carriers were passing the station. Another historical aspect are the Lastest Outgoing and Latest Incoming menus on top of the UI where outgoing will show the latest data when an arm has loaded an order upon a carrier. The incoming will be updated when a pallet has arrived back to storage, thus completing a run within the whole factory. The user can also see timestamps on these menus so they can calculate how long a complete iteration takes. As of this stage the colors were chosen with a darker background and brighter text for clear contrasts. The greyish background worked well to blend into the environment since it is not important but still give support to readability, the text and colored-images are where focus should be when an operator is overviewing the system.

This DRM iteration also saw a lot of work in the backbone system which is not directly tied to the visual aspects of the DRM methodology, but it explains why the Prescriptive study might not be as extensive as in following iterations. We found some real problems with portability along with scalability and wanted to address these before continuing the DRM process, mainly so it could grow and change at a faster pace. We also had problems finding documentation and mapping of the live system (as mentioned in 7.2), which had to be corrected before we could begin properly.

7.3.2.3. Descriptive Study II

We did learn a lot in the first iteration, especially about the backend of both our software and the Festo system. Tied to the DRM-phase we had the UI straight on the machine as shown in [Figure 21], it covered the ASRS32 front window but there were some problems with that approach. First of the machine got covered, the digital twin parts were unnecessary since the UI covered the digital implementation. As seen in [Figure 21] we hid big parts of the inner workings which would make an ordinary desktop application work just as well if not better. We also realized that the images and

historical data hindered real time information, thus we wanted to implement other logic where transformation in the simulation was important (foremost carrier positions) in real time. This approach would not make that possible so we experimented with alpha channels (see through materials) briefly, but this hindered viewability on the UI itself instead since we needed the darker background to see the text information in a clear fashion.

We implemented a simple button to hide the UI, but an apparent problem as we saw it was that all information vanished and we could not fulfill the criterias we mapped. But we liked the idea where we could actually hide the UI, it could be beneficial if the operator wanted to abstract away information and it would actually allow us to represent the system as close to the real world as possible as is the normal approach to DT´s. The upside with this first approach is that it is clear to which unit the UI was paired and we needed to keep that explicit information in some fashion.

7.3.3. DRM: Iteration Two

7.3.3.1. Descriptive Study I

We had to correct the viewability problems found since we could not cover some of the most important parts of the system. The button we implemented in the previous stage was working well so we made that a feature with some basic cleanup. We also needed to move the main User Interface so it was not in the way. Now the other machines were introduced so we could see a more complete picture of the flow. We found some problems with technical information about the software we built, we had no information on how the connections to the OPC endpoints went, if settings were properly loaded and so on. The biggest goal at this time was to create information about where all carriers are positioned since we thought that would really improve understanding of the live system. Modules should represent this logic when the carriers pass certain locations and if they had their own UI-elements we could enhance real world viewability and understandability. When we focused on other parts of the system and realized that UI elements should be placed at relevant positions on the DT to improve understandability, this could be anything from storage to the orientation of carrier flow (conveyor movement and orientation).

7.3.3.2. Prescriptive Study I

According to [Figure 22] we can see how the work evolved. The ASRS32 menu was now moved to the side for improved viewability of the inner workings of the module. We also started to move UI elements to the relevant position on top and inside the module, this worked great in many cases as its orientation acted as strong reference as to what the information was linked to. This works just as the menus on the physical units (the siemens HMI-screens) to give information without explicitly pointing it out, as discussed previously with [Figure 19] in focus. One of our examples are the arrows on top of the BRANCH module (closest to the camera in [Figure 22]) which worked really well to show if the belts were moving and in what direction. This was one of the big problems when overviewing the menus from the first iteration, it just had some colors as representation (still viewable on the ASRS32 menu in [Figure 22]). Another example is the light white-grey small menus attached to the ASRS32-module, they will show what is stored at each pallet position along with a text-color that is strengthening the overview.

We can also see a carrier [Figure 22] in the input line of the BRANCH-unit, we could now translate carriers depending on the stream we got from the OPC data. This meant we could abstract away sensor information by showing effects in the world itself instead, all in real time, just by translating and showing/hiding carriers. We did because of that knowledge not implement a whole UI section of the BRANCH, everything is instead floating on top of the unit since we found that to be a more effective approach. The only bigger UI element is the live feed showing what the system is doing. It is strategically placed to avoid it disrupting the physical overview.

The buttons (green user interfaces) on top of each unit hides all UI-elements linked to that module for a real world twin approach, but information will be lost if the operator does this. It is still a viable solution to abstract away all information and only focus on where the carriers are at any given time, this works well with different work-modes as presented in 4.2. We also implemented two feeds in the user Heads Up Display (HUD). As seen in [Figure 22], the upper right one handled all information from the modules no matter where the operators were looking. The HUD also projected startup information about the software itself in the left corner, such as connection endpoints and loaded settings etcetera. This will also improve understandability in what the system is doing at the current time.

7.3.3.3. Descriptive Study II

We located one big problem in this phase, and that was that the output from feeds on each module used raw data, mainly raw names as they are named in the OPC nodes. Generally they have the same name as the ones on the physical interfaces, but we wanted to take the opportunity to increase understandability of the system which we found the current solution had troubles with as explained in 7.3.2.1. This would also lower the knowledge threshold of the operator by increasing self explanatory solutions into the software. Errors are also hard to recognize as we wont get audio feedback as in the physical world (a machine stops “humming” when not working for example), and the feeds are hard to understand when an error arises in the current solution. This means the operator would have to go to each module and read the cryptic feed of what is wrong in our solution.

We also noticed that when we implemented the translation of carriers in the world together with smaller UI-elements attached to each carrier, the machine itself could be in the way of sight. It resulted in the operator having to stick his/her head inside of the machine to read all data in a clear way, this was not optimal in any way and this needed to be corrected as defined by the extended definition in 4.2. Another problem was the HUD-feed, when we started to increase data flow into our solution, it got populated fast and the user can't read and understand the data, it was not a good solution, especially in VR.

7.3.4. DRM: Iteration Three

7.3.4.1. Descriptive Study I

The negative effects we had experienced by showing data in the virtual world needed to be mitigated and one of the strong possibilities with digital twins is that we can alter and improve upon the real world if it is well founded. For example, many of the physical modules have safety measures preventing the operator from getting hurt, this can be everything from safety doors, locking mechanisms, sensors stopping the machine and the likes as mentioned before. All of these aspects could be abstracted away completely since the user cannot get physically hurt or damage equipment in the virtual world, which is one of the strong aspects of a digital twin.

We had to look into how we presented errors when they occured, this is one of the greatest problems we see with the current solution (see 7.3.2.1, Descriptive Study I: The existing solutions). Since this work focused on overview and debugging we needed to enhance the debugging capabilities. This was especially important for the factory or any production flow to recover as fast as possible from a fatal error in live production.

7.3.4.2. Prescriptive Study I

[Figure 23] presents the top of the modules were we implemented UI-Improvements to mitigate debugging problems and to clarify errors. We also added modes to the software and [Figure 24] shows the viewpoint when you enter the solution, which is in mixed mode.Tied to the BRANCH unit in [Figure 25] we can see overview mode compared to [Figure 26] that shows the Digital twin mode. We added arrows on top of the deflector arms showing how they affect the system flow of carriers. The robotino connection also got an arrow on the BRANCH so we can know when the unit is paired.

[Figure 27], [Figure 28], [Figure 29] and [Figure 30] shows comparisons between different modes and their strengths tied to the ASRS32 unit while [Figure 31], [Figure 32] and [Figure 33] instead gives the data from the perspective of the RASS unit. Notice that the feed still in some of these pictures are using raw data, that is because of the OPC logging. We had to send dummy data to each unit due to a bug when we recorded the OPC data to CSV files. But otherwise we had mapped the raw data and could now print out special messages according to the data. This allowed us to increase understandability of what the system was doing and it also gave a more production flow feeling than the raw data could ever achieve. We also removed the right on-screen HUD interface showing what all machines were doing in real time due to the increased data flow as found in the previous iteration.

Since we now could remove parts of the machine we had to implement more buttons for these actions on each machine. We also made sure to move these digital elements to logical positions on the DT, for example we can see in [Figure 33] that the red button actually replaced the Siemens-screen perfectly and the green button was placed on panels made for additional attachments on the physical twin. This improved the feeling of the elements and gave a more worked look and feel, especially in VR mode.

The greatest additions in this DRM iteration were the panels above each module [Figure 23] since we could now see from far away the health and status of the system. We also get a custom made message here whenever an error comes up, this was really handy as it provides more information than any error-code ever could give.

7.3.4.3. Descriptive Study II

The feedback on top of each unit did a lot for overview and health of the system [Figure 23] but if the factory was bigger and if we implemented other parts of this factory, all elements would be impossible to see at any given time, we had to think about scalability. Thus the operator could still miss relevant information depending on current orientation. We also saw that some data was hard to read in the virtual world on the UI elements, we needed to focus on readability so the user would not need to be extremely close to read the information or to see if something gets updated. We should comment here that better hardware could improve upon these problems as well and viewability in VR-mode is greater than these figures in this document can display.

We also saw that carriers could be hard to spot in the world as their color somewhat gets camouflaged by the environment (the gray background), especially if the operator has not abstracted away physical elements in Overview-Mode. A better overview of the carrier flow would be a good approach. A new operator might also have a hard time to understand the different stations that read each carrier position and there is no history on what carriers have passed each station in our software. This was problematic since there could be one second between carriers in the real world, and if the operator was trying to find a specific one it could be hard. These aspects were also some of the first problems we had with the original solution in 7.3.2.

7.3.5. DRM: Iteration Four

7.3.5.1. Descriptive Study I

We needed to present errors and debugging even clearer, we also had to increase viewability in the simulated world in some fashion. Before we started the new implementations we evaluated the UI in [Figure 34] which gave us information on what was working best in our environment. Black or white backgrounds, where we favor black, seemed to have the best contrasts, where white or black text on top also seems to be best. The picture was taken from the VR rendering for greatest realism in how these colour-schemes worked in the real application. We did not rely on research outside of this application when it came to this decision. Because of the technology we used, there could be some differences between theoretical and real solutions, which also allowed us to abstract this part of the

DRM-research. We also wanted some elements to be easy to see in the software and we argue that the cyan background with black text or the opposite are among the best.

7.3.5.2. Prescriptive Study I

This iteration saw great improvements upon readability, history and debugging features. We used the information we got from the previous iteration, but also the UI-color schemes we found to work best between the two iterations. As seen in [Figure 36] we changed the look of the carrier cards to be easier to perceive in the environment. It is also apparent by these two pictures that we lowered the UI resolution scale, this improved readability by a fair amount from further away because of pixel logic. It also lowered the load all UI-element puts on the application. It might not look as pristine on a closer look but the positive effects were hard to ignore. The stations on each module (where carriers could be read by the physical system, [Picture 16] as an example) also got "card-holders" showing the name of the station, but also some history of which carriers that previously visited each station. This improved understanding of where the carriers are, where they have been and how they move in correlation to each other. Abstracting away parts of the hardware became more important for a complete picture and readability for the operator.

The error cards floating in the environment [Figure 35] follow the users orientation so they can be read at any given time no matter where the user is in the environment, they also improve upon where the origin of an error is via the line connected to the card towards the module. This proved to be very powerful when we for example could draw a line to the emergency stop button paired with a visible card on top with information, this feature is very powerful and apparent when in VR mode. We also implemented a minimap [37] similar logic as seen in [Figure 38] and [Figure 39] so the operators can see errors in the world even if they are not looking in the object's general direction. The rendering minimap-camera is culling everything except specific UI elements and the physical assets of our choosing. The minimap UI-element sits on the user's left hand in VR and is up in the right corner when in desktop mode. The colors change on each module according to current working status, so in other words the user can get information on what module is offline straight away when this happens. This might also be a good approach if we scale this software to greater solutions in size. The ring in the middle with an arrow also shows user orientation and location within the scene, this is also good for orientation, especially for new users. In [Figure 39] we see that the three modules in front of the user are working, all assets behind the user are offline (not tracking, thus grey).

7.3.5.3. Descriptive Study II

We could see some obvious problems within this iteration and it was mainly connected to the Carriers and Station cards. They did not feel complete and the cyan colour felt misplaced in an application of this type. But we still needed the properties that comes with the colour-scheme, it needed to be revisited. The same goes with the other UI-elements, they did not feel complete and they did not fit the theme so to speak and we thought the whole experience would benefit from a universal feeling. Other than these problems we started to feel the solutions and new logic we introduced were beneficial to the complete picture.

7.3.6. DRM: Final Iteration

7.3.6.1. Final Descriptive Study I

Since we felt we were closer to proof of concept we had to rework how all elements look and make sure the whole solution had the same feel. Thus we had to redo all UI-elements or tweak them so they felt more uniform with each other. We also wanted to use the minimap-logic more extensively and add information so we could know what problem had flagged on the machine by just looking at the map.

7.3.6.2. Final Prescriptive Study

As seen in [Figure 40] we updated the UI elements in the world so they are even easier to see and give a more worked feeling, but at the same time remove the complete cyan theme on the carriers and its stations. In [Figure 41] we can see how other UI elements started to get the same thematic rework, we also revisited the UI on the ASRS32 module, we can see a bigger feed and we created better cards and logic for input/output and real time station data. The colour coded images on the bottom were removed due to new logic on the belts, see [Figure 43] and its discussion. In [Figure 42] we can see the error logic on the newer system, mainly the minimap (top right) now also has smaller sections showing red or yellow smaller sections, these are error specific elements and its position tells the user what error that has flagged (only the common ones are shown in this manner).

First we chose to go back to the general color that we had previously, but at the same time use the cyan properties as borders and text for that improved visual feedback. This made all elements easy to see and define in the world. The Stations and Carrier cards got remade as well, we increased the size of the ID number and highlighted it so the user could easily see where each module is, even from further away. The problem with the cyan color is that it doesn't really fit industry and it feels alien in this environment, but we still wanted elements to be visible in the digital world. Finally we used the color-scheme found in [Figure 43] for a more thematic feeling. This colour was chosen because it fits our personal connections in obvious ways (MITC and Mälardalen University has an orange colour in their logos), this makes the solution also fit in a wider perspective and improves thematic feelings inside a corporation. This highlights a possibility for company branding of the environment.

As seen in [Figure 43] the active card (with payload) has the previous cyan coloured borders, we did this to improve visibility of carriers that had important information, they will simply stand out compared to the standard cards. One of the greater strengths we added in the last iterations was the introduction of animations upon the arrow cards on top of the modules. The red-background crosses are stations that are currently stationary, thus the red-cross UI-cards are stationary as well. But on the BRANCH-module we can see a green-background arrow and it will move along the belt path to show it is moving and we also set the speed here to try to match the real belt speed for improved connections to the physical twin. This little step made the solution feel a lot more alive and worked. We also added some ease-of life solutions to the settings file where the user can shut these animation off, due to some users might feel motion sickness if parts of the environment is moving, and to abstract away unnecessary parts (top elements) of the DT at start since it will be tedious to click the button on each unit every time, imagine 100´s of units.

7.3.6.3. Final Descriptive Study II

We felt we had proof of concept so this became our last iteration. There are of course hundreds of improvements that could be made tied to the DRM methodology, but as a complete suit we felt we had something to present and discuss. The complete software is presented in 7.4 below for a more complete picture and walkthrough of the solution we created. The negative aspects with our software will be discussed in the Result section (Chapter 8) of the report and because of this will the last Descriptive II study only link where the information is given tied to this phase.

7.4. The implemented Research: Final Software

Some of the information given here will already be discussed in the DRM research of the report (see 7.3) but here we aim to give a complete picture from the finished products perspective instead. The content here will build upon the DRM research and future content in the report will use this section as backbone instead, mainly for the Result (Chapter 8) and Discussion (Chapter 9) sections of the report.

First off we want to comment on the color we chose for the thematic feeling. As is now we tried to use colours that are easy to see and stand out where needed in the environment. We also finished with an orange colour-suite because it is heavily recognizable within the organization this solution is meant to be used. We think and recommend that future work using this report as reference should think about this when implementing the elements in a DT solution and what works best in that situation. Also note that according to the description of the DT, we are allowed to enhance upon the reality which might not be true for future work, this needs to be reflected upon if this work is used as reference.

We will showcase the complete software below in the text as it is after the DRM iterations. Also note that these pictures were taken from a previous run made with an modified CSV file to showcase different aspects of the software, because of that some logic might seem strange and it would never occur in a proper OPC-run (such as different belt movements, carrier locations, representation of data and errors). This is also one of the greater strengths with our solution, we can create scenarios that never actually happened by using a proper run of the physical twin as backbone, this is especially good if we want to teach in specific scenarios or learn how the system work, or as we do here, showcase the solution and its systems. Also note that we alternate between VR and Desktop mode in these pictures for a broader understanding (pictures with the minimap in the upper right corner come from the desktop version).

In [Figure 44] we can see the finished solution from further away and all physical parts that are unnecessary are removed for improved visibility, this we call overview-mode. Compared to [Figure 45] were modules are seen as closer to the real world twin, we define this as Digital Twin-Mode. Our solution is easy to modify if we also want to remove carrier-stations, errors and belt movement cards to be even closer to the real world representation, the elements just have to be added to a list in the engine which increases possibilities and modifiability. The RASS system, as seen in [Figure 46], has a pallet with order number 2261 tied to it, the deflector arm is now diverting to the robot-belt. We can also see the historical data upon the robot station, the latest order was 1396. Also notice the pallet has another colour now, this is to let it stand out in the environment in a clearer way that an order is present. The orientation of the production flow as seen in [Figure 47] has the red crossed UI-elements on the belts which tell us that the belt is still, the green arrows shown will animate and move in the direction they are pointing along the belts. We can also see that Carrier 6 is on its way to the robot in the figure while carrier 16 is moving forward back to the ASRS32 module.

The ASRS32-storage is presented in [Figure 48] were the unit has just gripped the pallet (see the feed) on the offloading/incoming station. We can actually see what the arm is doing along with other data in these feeds. We also showcase that the outgoing station has sent carrier 6 with the same order, see [Figure 47], it is still in the system. This pallet in view (with the finished product code 214) will be stored in index 1. Note that the ID is missing here on the main-UI, we show how unchanged values look compared to changed values, the difference is meant to draw attention to important fields in the environment. In [Figure 49] we see how the front doors on the ASRS32 module has been opened. These floating cards (red) have the warning on top animated to draw attention. The card itself will follow the operator's orientation in the world so it is always pointing towards the user. Also tied to Error-handling we can, in [Figure 50], see how the User Interface is mounted on top of the users left arm. We can also see that the red line from the error is pointing towards the emergency stop, this is also apparent by the location of the extra box (on the minimap) in front of the main red UI (biggest box). We can also see the users main orientation with the arrow in this picture which helps with location information, this could prove important in greater (scaled) solutions.

Tied to how the solution works our right hand controller has a ray (generally longer than shown), the blue line with the ball at the end in [Figure 51]. This line can trigger elements in the world. The user just has to point the handheld device and trigger a button to interact from as far away as they want. Mainly buttons such as the green ones on the ASRS32 in the background listen to these interactions. The desktop solution has this ray attached to the camera instead and the spacebar will trigger the elements in the environment. Movement is done in two ways in VR, first is physical movement and this is to recommend. The user should be in a larger room where they can move as in real space, this is sadly not always possible, and longer sessions might even force the user to sit in a chair. Because of this the left hand has a touchpad that enables movement in x/z-plane. By pushing the touchpad (left and right) on the right-hand controller the user can move in the corresponding direction for ease of use. All our buttons are modifiable by Steam's own UI overlay, this can be triggered from the menu if the user is in VR-Mode. We should also point out that we used a VIVE-PRO HMD and Controllers when we developed this software, but all Steam compatible equipment should be able to run the solution as we are using their API.

7.5. AR complement and its potential

The possibilities of Augmented Reality are one of the real goals of our work. Since we can use the UI elements and the backbone of our system to port to an AR application we could use a hand-held device (such as a smartphone) or holo-glasses to actually see these elements in a real world solution. In other words our UI enhancements could be shown in physical space just by using one of these mediums. The only difference between our solution in VR and a ported AR solution would be that the DT is replaced by the PT (Physical Twin). Imagine the possibilities and understanding we could strengthen about machinery if we could use these additions in the real world space, all workflow and current health of the system could be seen at a glance. And if a user has glasses with holographic capabilities (such as the HoloLens [Figure 55] from Microsoft [13]) it would not interfere too much with the operators normal equipment as well as allowing a less disruptive workflow. He/she could even control the environment via holographic buttons and handles. This is where the real power lies within this work. As our definition of DT allows us to enhance upon reality the work will be more impactful in AR as well as VR compared to stricter DT implementations. An Augmented Reality application for the Android system was developed to show the possibilities of the status and overview part of this project, mainly as proof of concept that portability is possible within a small time frame:

Sometimes an operator might need a quick way of verifying the current status of a specific module. To prevent possible disasters due to the human factor, for example if the system relies on a particular set of input pallets but the operator puts the wrong parts there, then the AR solution can be used as a tool for verification. With a hand held device such as a smartphone, tablet or even a head mounted device like the Microsoft HoloLens, the operator scans a QR code that automatically connects to the corresponding module and through the OPC protocol retrieves the setup that the machine is expecting. The setup is then displayed as an overlay onto the physical module for the operator to compare. This makes it a quick tool for verification and to avoid pallet mismatch.

[Figure 52] is rendered from the screen of a mobile-phone and no afterimage changes have been made. We are scanning a QR code tied to the machine, this will trigger the live OPC connection and start the overview and in [Figure 53] we can see the next stage were the OPC connection has been made. We can use the pallets here to compare what the system thinks is present or what it expects. This is again a screenshot from the phone showing what the user is seeing via the camera feed.

8. Results

The result section will focus upon the questions and discussions tied towards the DRM phase we presented in 4.1. We aim to link back the data we produced tied to the overarching question as Chapter 8 moves on.

8.1. Comparing the solutions with regards to debugging and overview

This section aims to define the data tied towards the comparison between the current solutions, we will look at positive aspects on both sides and by doing this we can define if our solution is proof of concept or not by looking at the differences. It will aim to answer the visualization goals but also discuss and present an effective solution compared to the original system.

First of our software has two modes (VR/Desktop) and we need to discuss these in relation to the MES4 (see 7.3.2.1 above) solution. The MES4 also has two different modes, if not three depending on definition. First is the desktop version that the user interacts with at the computer that is connected to the factory. The second is the Siemens touch-panels on each module, thirdly we have the physical aspects of the machines such as sounds, lights and visual overview, this last might not strictly be tied to the MES4 solution but it is included for broader comparisons.

The focus we previously mapped in the criteria definitions (see 5.2) will be used as headlines in this step to define positive and negative aspects of our and the previous solutions, by this approach we can answer each of the targeted problems and its comparisons in a conclusion.

8.1.1. Ease of use and overview

8.1.1.1. Feed forward

Feed forward [26, p. 126] defines if the operator needs to be trained in the software to use it or if it is intuitive without external knowledge. Strictly the wording means that it's clear what actions the user can make and what the effect of those actions will be.

The MES4 software is a complex desktop application and it is not only meant to show system data, but also configure and control the factory. This sometimes makes overview suffer due to the scale of the product. Its primary focus lies in how to operate the machinery and this forces the user to have greater knowledge about the physical system and MES4 and because of that it will be harder to use unless the user has some sort of training. This also allows for complex structures of menus and that abstractions can be made to hide unnecessary elements. But as an effect by doing this, feed forward will suffer and the software will be harder to use. Our software instead focuses on showcasing the system and because no effects can be changed from our software the representation can be kept in a simpler manner and we can actually focus on ease of use. The user will trigger the software and just jump in the virtual world. Our main menu in the game is always present and the user can find help from the Steam API as well for usage in VR as presented in [Figure 54]. The buttons tell us what they will do for example when we look at them and we try to use self explanatory words to limit the need for documentation in how the solution works.

One aspect of our solution that might suffer just as the MES4 solution is our settings file. It has the same type of technical wording and the user needs to know exactly what they are doing, if not, the software will stop working. But we argue that an “expert” can set up the software and afterwards anyone will be able to use the solution. We also chose to let the same software have the capabilities of using VR and not use VR. This will allow one installation of the software to work in multiple ways, and not separate into different solutions for different types of use cases. Such as using VR or not, logging data or not and connecting to OPC servers or listening to saved logs. We argue that this brings modularity to the software and for it to be easier to use in the daily activities by increasing versatility.

8.1.1.2. Complete Picture of a live system

Our solution in perspective of the factory aims to improve overview by showing when something changes in the solution, this is primarily made via colours and big menus. We also use a minimap to showcase where errors are even if the user isn't really looking in the general direction (see 8.1.2 below for more information). If we stand further away in our solution we can see more of the modules but there is an apparent problem with this approach. The more we want to see, the further back we need to be and this is a problem since the resolution will hide information to a degree, especially text elements. Because of this we made sure that production flow could be visible from further away via the Carrier cards and the stations. Also the animations on the conveyor belts help as well. We argue the strength in our software is overview of production flow and overview of error handling.

As we saw in the MES4 deep dive (see 7.3.2.1) the solution lacks overview capabilities. Either we focus on raw data in one section of one machine, or we get a general working/not working interface for the user to overview. But with that said, MES4 is meant to control the factory, not overview it. That should be done on each of the attached screens on the modules, but we also saw some problems there. The errors were small and not always self explanatory, the data related to the module activity was also split into subsections and no complete overview was possible. Even if we inspected this data it could be hard to know what we were looking at. With that being said, one of the greatest strengths of these screens is that some of them have pictures with live data in them. This was something we tried to copy with the locations of some of our UI-elements (such as ASRS32 storage cards, belt movements and orientation of deflectors etcetera).

If we focus on the Carriers and where they are and what they have mounted, our solution will be better than the real life solution in some fashion. What we can't beat is the live orientation of carriers at all times. We could only get data at sensor positions to display where the carriers are and on these locations we created stations to improve understandability and make historical data available. But our solution can show what is on a carrier such as it’s order information, real life does not have that capability and we argue this is one of the greatest potentials of our software. In larger workflows it could be very important to be able to follow production orders throughout the production chain.

8.1.2. Error handling

8.1.2.1. How does the solutions flag Errors

The MES4 shows what has gone wrong in 4 different places. If we focus on the two relevant places in the MES4 software first we can see that the first of these is the overview screen that has a red/yellow/green colored dot on one of the units if something is wrong [Figure 8]. We also have a window showcasing the specific errors on each module but no real explanation to them, even though a specific field for comments for this exist, they are not always descriptive [Figure 12]. If we shift focus to the onboard screen on each module the error handling gets better. As seen in [Figure 17] we have a dedicated menu for this and we can reach it via the red border on top of the screen (touch it). Here we get the same information as we get in the MES4 Error screen, but it is clearer when it comes to the severity of the error (red/yellow colors). The operator can also acknowledge errors and fix them from this menu, which means the user has to walk to these screens anyway to solve problems. The last error flags the physical factory can bring up are light stacks (red, yellow, green) on the CP-F-BRANCH, as seen at the top of the measure unit in [Figure 22], or the Robotino asset. Another factor here is sounds as previously touched upon. The RASS unit will signal if that specific unit stops, for example when the emergency button is pushed. This will tell the operator that something is wrong, but it will only occur if the RASS module has stopped as well, which is far from the case every time. One might argue that Festo only implemented this for showcase as to how the system could flag errors, but the CPF error handling suffers from clarity in how and when errors occur, especially how to solve them. There are strong arguments that there should be a manual for these situations, but we believe our solution even improves upon that.

Our solution, the DT, is flagging and showing errors in a direct approach. We have three different solutions for this. First of are the top panels shown in [Figure 23] which are colour coded in 4 different manners and we chose these colours to be in line with what was already present in the MES4 solution and the reasoning is familiarity. Bright Green means the unit is working, generally in Automatic mode. Yellow/green colour means the module is doing work, but we do not know in which mode it is working (this mainly occurs before we get the OPC- data showing the xAuto variable from MES4). Yellow means the module has an error, this will only be true if the module isn't in auto-mode. As seen in [Figure 44] the machine is working, but the yellow warning is still true on the card floating (low battery warning), more about these cards below. Red means the machine has entered full stop due to an error, we get a message from each error on top of the board showcasing what is wrong, this is fully customizable in the source code via a string on each error so expandability is thought of.

The Second way we flag errors and warnings are the floating cards as seen in [Figure 49], for example if the door has been opened we can create a line towards the sensor or point of origin of the fault. We argue that this increases understandability by a fair amount without giving explicit information. The cards will also follow the operator's orientation (they rotate towards the user) to always be readable no matter where the user is in the simulation. On top of each card is a warning icon and it will animate by pulsing, we did this to call attention to the cards for improved perception. Lastly we show the errors via a minimap similar logic [Figure 50]. This is a powerful complement when the user is occupied by other activities. In VR the user can see the error on the left arm and we chose this approach so the UI won't be in the field of view unless the operator wants it to be. In the desktop solution we had to have a fixed point for this UI so we chose the upper right corner instead. The UI renders a camera in top-down view that has UI-elements on top of each module. These UI elements are squared in the same outline as the module it is meant to represent and they have the same type of colouring as the top panel UI-elements showing errors. Green if working and Red if not as a quick example and this is a clear way of showing which module is offline and the severity of the problem. We find that this could be powerful in a larger solution (bigger factory) where the user might be very far away. The minimap also has smaller squares tied to each main-square on each unit and these are directly tied to specific errors. For example in [Figure 50] can we see how the emergency button has been triggered, a smaller red square sits in front of the greater red unit and the yellow battery warning is also showing in the back of the same square. We did this to showcase normal problems, the user will quickly tie these locations to specific problems and react accordingly without more information needed.

8.1.2.2. Error intuition and if they are easy to miss

Since we find MES4 lacks in the error handling we would have to argue that the errors are not intuitive in the main desktop solution and because of the hierarchies in the desktop solution they might not be easy to see or mitigate. We do find that the module screens are somewhat better because this is where we acknowledge the errors but in a whole we find that the MES4 have some clarity problems as previously stated. This is also one of the main reasons we did this research to improve upon and find these problems. Our solution has error intuition in focus when it was developed and is built to expand and be heavily modifiable in this regard. Because of this we find that we have better intuition via point of origins, explanatory text and solutions that are harder to miss via multiple solutions as mentioned in 8.1.2.1 above.

8.1.2.3. Clarity in how to recover

How to recover is one of the greatest questions in a real factory since fatal errors (factory stops) cost money. We find as mentioned in 8.1.2.1 above that the MES4 sadly doesn't focus on this. We argue this to be true because of the lack of comments and how hard it is to see the errors with no clear solution. But we might also mention that the Festo-system is built for research and educational purposes, downtime will likely not cost money, it might even be part of its purpose (as seen by this report as an example). A real life solution probably handles this better than the scope of MES4 and Festo CP-F but we argue that the knowledge produced here is vital for any solution with production in

focus. An intuitive way of recovering from machinery stops could be very beneficial for the operator that is not fully experienced in the work environment or when help from colleagues is not easy to find.

8.1.3. Information Representation

This section is important and will focus on how data is represented such as if it is clear what we see and if information is easy to access along with how the solution handles modifiability (can we change what data to use and focus upon).

8.1.3.1. Access and overview of data

The MES4 solution has some problems if we focus on representation but that is not so strange. As previously mentioned in DRM 1 (7.3.2.1) the solution is not developed as an information platform, the users are meant to control the application from the software. This means information is baked in different menus that often is used to modify the hardware. One menu that is meant to see the raw OPC data is showcased in [Figure 9] but a complete overview is impossible due that the different fields are in a drop down hierarchy which result in poor overview. The positive aspects with this are the abstraction among which data can be overviewed and we tried to implement this in our solution by letting the operator choose exactly what data to view via the settings file, more information below. But the information in this watch-table view on the MES4 solution does not even have explanations on most of the data and the reader needs to have extensive knowledge about the internal workings of the CP-F to understand the data. When information in this view updates, the table will just change the corresponding value and no other information about updates or changes in values are highlighted. We can also see that these watch-tables are generic since much of the information has no link to an OPC-node, this is also a negative aspect when we focus on overview. Optimal would be if the solution was hiding these nodes completely because the data that is not used takes up space on the page and might draw attention even if it is not used.

If we instead focus on each module with support from the data overviewed in the human-machine interaction section from the first DRM, in 7.3.2.1.2, we see major improvements to representation of data compared to the desktop version. For example [Figure 60] shows the first stopper on the RASS module. Festo ties data here to a picture and we found this to be a very good approach to give information and how the sensor fit in the environment, this was actually one of the ways we mapped the data to the underlying system. We wanted to incorporate these ideas in some parts of our software (mainly point of origin of UI-placements such as stations and errors as discussed above). This is by far the greatest implementation Festo has for overview and giving explicit data, but one of the problems is that most sensors are not explained to a greater degree than the location. We believe there is a good reason for this and that would be space on the screen, it would simply not fit if each node was named in a good manner, even less for complete comments. This is one of the most important aspects we set out to improve upon. The colours on these screens are also quite intuitive and we chose to mimic the green/yellow/red colored approach for familiarity.

Our own solution is instead reimagining the whole information output of the physical twin (PT). Since our definition of a DT system (see 9.1 discussion) allows us to alter reality if there is a good reason for it, we can actually alter the DT in ways to make the solution improve upon reality. As previously mentioned this strengthens the usability compared to a stricter solution that only presents the PT as seen in the real world. These alterations we made, such as abstractions of security solutions on the physical system and allowing information to float in the environment and be a part of the system via UI elements, we can actually present data in new ways. Our solution has feeds as seen in [Figure 48] and in these we actually had the space to rename all events that occur in the OPC-server. Let's say variable xAuto from the "*mes*" hierarchy turns true, we can actually write out what that means in this feed as pure text instead of variable name and what its value is. This we felt were really important for a continuous understanding of what the machine was up to. One apparent problem became true though, the speed of the feed can in some cases be too great. This means it can be hard to read in real time when the machine is producing a lot of events in a short time. We have some ideas here and one

solution among others could be different feeds next to each other. Let's say one for important data, and one for less important which could prove to be beneficial, we could even expand the settings file (see 7.2.5) to include where data should be output via priorities for example. The settings file also in current functionality allows the user to comment out all unimportant nodes via a '#'-sign in the beginning of the line. This will improve flow by a substantial amount and it is one of the greatest features our solution has for modification, something the MES4 system does not allow at all. We could even expand this file to allow custom messages as output depending if the node data is true or false to even improve upon expandability and maintainability. This means we can alter behaviour without ever touching the source code, it's so impactful we even recommended it as future work (see 11.5 for more information about this thought process). By doing this we could also make the source code simpler where we only have to listen for the events that impact real logic in the DT-solution (which might be 10% of the complete mapping we made).

If we bring up [Figure 60] again compared to [Figure 48] we can see that space in the environment is far more allowing for us and placement as compared to [Figure 60] is even easier for us, for example the storage cards and station placements. We argue MES4 did this in a good fashion, but a DT system as ours are far more suited for this kind of representation. We also want to allow even greater detail in which sensor is lighting up on top of the DT system. As seen in [Figure 48] the feed has the numbered event of 1806 (a carrier has just left the second entry belt) and we could represent this by lighting that sensor in a colour, such as green when true and red when false, in the exact position on the digital twin. We never implemented this logic due to the amount of implementations that would follow, but as a concept we think this could be really powerful. One step further would be small UI-elements pointing towards the sensor, looking somewhat as those that we tied to errors in [Figure 49]. But following research into this we have to be careful with this technology, information could be too substantial and relevant information could vanish.

We have some other aspects to focus on such as the operator has to move in physical space to get the information. This is true in the PT-system and we recommend it to be mimicked in the Virtual Reality mode of the DT as well. Because of the increased realism that comes when the operator move around the DT-system (without controllers input) we increase upon reality and the operator won't be as motion-sick (if he or she suffer from this), a large empty space would be needed for this approach which might be impossible for upscaled solutions. If the operator can handle VR without physical effects he or she could sit in a chair and stay in the same place, these approaches improve upon secure work environments. The Desktop Mode lets the user sit in a chair at all times when using the virtual solution. We bring these aspects up since work environments in some PT-systems are dangerous (such as toxic, high speed machinery, automated systems with lower security defences for humans and so on) and allowing the operator to work in a safer place is beneficial in the next-gen industry.

8.1.4. Design (visuals) follow up and improvements upon the previous solution

Some of these aspects defined here have been visited above in 8.1.3, 8.1.2 and 8.1.1 but we want to make some clarifications in the field of pure design choices.

The colour schemes we chose were targeted for visibility in a simulation (mainly to stand out in VR and especially AR). The DRM research had some interesting choices as iterations went by. The cyan colours [Figure 37] for example did not really fit an industrial solution, even if they were easy to see we had to tone down the feeling that came with the colours. Finally we settled with the orange setting [Figure 48] because we saw similar properties in viewability, but also because it has strong connections to MDH (Mälardalens Högskola) and MITC (Mälardalen Industrial Technology Center) for an improved thematic feeling. We argue this enables suits of software (even from different sources) to be created for specific organizations in a manner to feel more connected. Because of this we believe that future work will think about two aspects: viewability on the medium used to represent the data (Hardware connected to VR/AR and Desktop simulation in our case) and familiarity for the operator. Another familiarity aspect were the warning colours we chose: Red, yellow green. As a member of

society the operator has strong connections to these colours, just think about a traffic light. We also choose to enable a panel on most UI-elements and the thoughts about this is that they will stand out and be clear in an AR solution where we render elements in the real world instead of a digital twin. If we compare our solution to the existing MES4 solution we can sometimes see the blue colour attached to the Festo-logo in the menus on the HMI's, such as seen in [Figure 20]. Their UI-theme is based upon the green colours of "working" on elements that are triggered and the menu-buttons goes in the blue/white theme of the Festo-logo. Our solution could have used the same blue tints for a feeling towards the physical system and there are good arguments for that approach, but we went with MITC and MDH instead.

Festo's orientation of menus in regards to pictures have an optimal approach on each HMI for giving explicit information. We tried to mimic this as previously mentioned by letting our UI-elements float and be attached to parts of the DT to give a grouped feeling while at the same time give information. This positive aspect of the PT is hard to keep up with but we argue we actually improved upon the information and further work can even be better. The relevant information has been grouped in intuitive ways on the physical solutions, but they hinder overview as discussed above so we tried to take a middle-way with more focus on overview.

9. Discussion

As presented in Chapter 8 our solution not only is proof of concept, it even works as intended in the targeted fields. Since this thesis has overview, understandability and debugging as a viewpoint this is how we developed the software and we believe the software fills the gap in these three fields compared to MES4.

Since this thesis aimed to answer the overarching question:

What are the advantages and/or disadvantages of monitoring and troubleshooting a Festo CP-Factory by means of a digital twin-driven visualization?

We will discuss and dissect this in the upcoming sections to prove that our software gives the answers needed. This will be done with the help from the discussion in 4.1 and gave data towards in Chapter 7 and Chapter 8.

9.1. The strength that derives from digital twins

This section will focus on the targeted definition of the digital twin and why this is very important for future work, evolution of Industry 4.0 and upcoming discussions.

As seen by the extended definition in 4.2, we allow the solution to be altered in some specific ways if there are strong arguments for it. The system should be able to mimic the real world in a stricter manner such as those definitions mapped in 3.1, especially Bacchiega 2017, but the real strength of a Digital Twin as we see it should focus on improvements upon reality, not reality itself. By shifting the focus of DT-systems in this manner we can create better solutions than reality could hardly ever aim to achieve and the transition between VR-enabled DT-solutions towards Augmented Reality ones along with Desktop support will be easier. As we define below in 9.4 in regards to AR/VR usability, the AR system approach can and will benefit more from a loose definition since it will allow for easier portability between VR/AR/Desktop. In other words multiple solutions would not need to be developed if linking these two definitions under one umbrella and it would be beneficial because it allows one software to be used in more ways. The feeling of VR is increased by a lot by a strict Digital Twin definition but it is not needed in an AR application at all, the physical twin (or a QR-code) will act as base for the UI positions in AR. In VR the DT system will act as this base instead and because of this the strict possibilities are needed if we want portability between VR/AR. Another goal to allow the DT system to alter reality comes with our goal of overview and understandability. By removing parts, in a VR tied DT solution, we can see the inner workings and abstract away unnecessary parts of a

physical system. The user does not need security parts such as safety doors, panels or even move out of the way of moving equipment, this is a real strength. By removing the upper sections as seen in [Figure 44] compared to [Figure 45] we can actually improve upon how much information we get from the environment since unnecessary parts are removed as they won't add any information. This is one of the greatest possibilities and strengths with a definition like ours and the reason we think this is necessary to improve upon DT-solutions. Related work in the digital twin domain [17] identifies commissioning of virtual solutions to be costly both in time and education, which we find to be true as the data mapping of undocumented parts of the machinery can be very time consuming. Especially if the machinery is of aging character and not as rewarding to work with as the Festo CP-F.

9.2. Discussion around the OPC solution and its logging

As presented in detail in 7.1 we chose to use OPC-UA (Open Platform Communications - Unified Architecture), mainly because of the subscription logic that were easy to implement in the C# source code [32] and that the Festo system already had support for OPC which made the choice logical. Previous work in comparing OPC-UA frameworks [19] found out to be informative in our search for a solution compatible with Unity3d. By implementing the protocol along with the settings file (see 7.2.5) we could even let the operator change logic outside of the source code via some simple lines. By choosing OPC we also allow for expandability with the same argumentation and, as discussed in 11.5, we have ideas to utilize this even to a greater degree and impact upon logic, all these positive aspects are possible because of the subscriptive nature of the protocol. OPC is also one of the most standardized solutions for industrial data communication on the market as discussed in 7.1, which means our work becomes more generic.

One of the greatest problems with OPC as implemented here is that the datastream is not instant which would be the case with a direct link to sensors, this means we have a delay between data arriving to the DT and the event of the mapped data. For clarification this means the data-stream won't be constant since OPC has delays between the events that fetch data. We ended with a setting of 400ms on each of these fetches but it is probably possible to bring down these numbers if needed. We chose this delay so we won't impact the OPC-server too much and we felt it was possible to still map this system well with that delay. Also note that all data that is changed between these delays will be sent in bigger packages containing multiple events which brings down network traffic and congestion. One should also note that while this might impact real-time logic when connected to a PT, the CSV save file, see 7.2.4 for more information, won't be limited but this fact. The data that is saved uses the real time of the event as it occured in real life in the PT, not when the software received the data. This is also one of the upsides of the OPC protocol because in combination with how we implemented back-bone logic we can recreate events from these saved files just as an OPC-stream would work (but without the delay) and by doing this we immediately receive some bonuses to the software. Some of these are, but not limited to: Replayability of a previous run (as long or short as needed), training and experimental scenarios by sending modified data (perfect for testing security) and deep diving into specific problems that might have occurred previously such as to bugs or errors. The usage of CSV files to find bugs and to overview afterwards greatly impacts the success of our solution in a positive manner.

The biggest problem we had with the protocol tied to our workflow was when we mapped the data. Since OPC as a protocol works like a complex file-hierarchy just like folders and files, and the solution can have more than 2000 nodes on each module, this took longer than expected. We had no extensive documentation of what the nodes represented and where they could be found so we had to manually map and filter them in most cases. Festo hosts a web page [34] dedicated to this type of documentation but the information was lacking some of the most important nodes and where they could be found. Some were not even used in the hardware we mapped, but as a start and mapping comments on what the pure data-nodes meant, it proved to be sufficient.

9.3. Discussions with regards to debugging and overview

9.3.1. Discussing ease of use and overview

With support from 8.1.1 we argue that our solution that is built for overview will handle these parts in a better fashion due to the arguments made. But we should remember that this is not the goal of the MES4 solution. The MES4 will be more complex and harder to use because of the capabilities of controlling the software, it is just a matter of what the solutions focus upon. We could develop our software to also control the physical twin, but we are not allowed to do this according to our definition and there are good reasons for this approach (see 4.2). Security is foremost and it could even affect the properties of our software in negative ways when we focus on debugging and overview. We believe our software is quite easy to use and modify, especially when the settings are properly used and we also gain positive aspects tied to overview.

Tied to overview from 8.1.1.2 the complete picture and health of a live system is important and we focused to improve upon this important factor. One of our implemented aspects helps by showing historical data which gives a greater understandability of what the system has done previously, mainly via stations showing which carriers have visited previously and menus on the ASRS32 that show incoming and outgoing orders towards its storage [Figure 48]. Also the RASS will show what previous orders were produced before connecting the DT [Figure 46]. As we implemented these functions one might argue the usage is limited as is now and we recommend expanded functionality in future work, also note that the CSV logging is a strong addition towards historical understanding. We also implemented timestamps in most locations to allow the user to know when events occurred in relation to each other. The Branch also shows the complete date in its feed so we know when the run occured, especially important in CSV-mode when we look at saved data. We can't really compare this towards MES4 since it does not include stored data and runs in this fashion, this is a pure extension in functionality on our end.

A complete picture also needs data overview as it is live, the MES4 solution had obvious problems here as discussed above. Our software instead tried to improve upon these aspects by allowing event feeds and this is discussed more in detail below in 9.3.3 that focuses more on how we display information.

9.3.2. Discussing Error handling

This section builds upon data gathered in 8.1.2 but before we review it we have more ideas on how to handle errors in our DT solution: ambient and point-lights along with directional sound, which is possible with frameworks from the Unity3d engine. These could improve upon error handling even further. Solutions like these in the environment could in other words strengthen our solution even more compared to original solutions. Even animations and text on the screen in front of the user could improve these steps, but as we aborted the DRM before these implementations, mainly because of the time-factor and we already felt proof of concept had been achieved, they were never added to the solution event though they could be important.

The MES4 and the physical factory does not focus on overview and specifically error handling. Our solution is instead built with these two aspects as main focus, according to arguments made in 8.1.2 we believe that our solution has reached its goal and are the better solution if we want clear errors. It gives some data on how these could be presented for improved workflow and understanding if errors occur. There are of course situations in which our solution allows the operator to miss important information and we believe this is one of the greatest problems any overview software stands before. There might be errors occuring (such as pallets getting stuck, a belt breaks or power cut offs) that the OPC-server does not monitor via sensors, if that occurs there is only one way of finding the problem: overview of

the real world. This could actually be mitigated in a DT with some simple steps, mainly with the help of a real live feed of a camera inside the 3d engine.

The user could also miss information if his/her focus was elsewhere and as discussed in the beginning of this section we have some solutions. We strongly believe error handling is extremely important in any manufacturing process. Shutdown caused by errors could be expensive or even dangerous towards health and safety for personnel and they need to be found, fixed and reset as soon as possible to optimize output. A DT system could even allow for predictability against these problems, see 2.3.1.2, our solution won't focus on this other than the positive side effects of the OPC-logging. With these arguments we believe work towards DT-solutions can and will be beneficial to most organizations in the manufacturing business.

Our solution uses multiple sources to showcase errors as clear as possible. First off is the panel on top of each module, this will also show a comment as to what the current error is. The implemented feed will also show which errors were triggered, this could be expanded even more (see 11.5). The solution also has animated cards that float in the environment with lines pointing towards each problem [Figure 49] that greatly improves understanding of where the problem comes from, they are also easy to see from further away to direct an operator to the proper location. As seen in [Figure 50] we also implemented a map and this enables the user to see problems and which modules that are affected no matter where they are in the environment. All these aspects were important in conjunction to each other with colour coding to build a software that really focuses on how to detect and solve errors. We believe we created a system that by far improves error-handling, especially compared to the original solution that was lacking in this aspect.

9.3.3. Discussing Information Representation

As the MES4-software is created with a smaller solution in mind, see 8.1.3, where the user could easily walk around the different modules the desktop solution of MES4 is not made for overview (or debugging), this is meant to be achieved with the HMIs (Human-Machine-Interface) on each module. This approach might work for the Festo CP-F but if we want to scale the solution this might not always be true. Festo also has some apparent problems when giving information about the complete picture of a single module, much less the whole system. We have instead tried to improve upon the overview aspect but we also suffer from some problems, such as information overload in the feeds but we have discussed some solutions to this above in 8.1.3.1 to mitigate. As a quick reminder the feed can receive too much data which means overview impact in a negative manner, but by using different feeds and modifiability we can decrease the impact of this problem.

We believe that a DT-solution has the ability, with our definition as clarified in 9.1, to be developed with other focuses in mind than pure reflection upon the real world. We argue that the system we developed is better at overview, despite the feed problems, and imagine what could be done in an ever greater timespan and resources. Another aspect of this is what an organization's goal could be with a DT-solution and as we aimed for improved understandability of a live system, it might as well be developed for: showcasing a production facility, specific order tracking, historical aspects, tutorial runs, training, predictions and improvements to name a few. Our solution has some, if not most, of these aspects as well via the CSV logging as discussed previously, but because this was not in focus we believe improvements could be made in each field, it all depends on what we need the software to improve upon.

9.3.4. Discussing the solution design (visuals)

Festos approach, as discussed in 8.1.4, regarding the smaller screens with high contrasts on each module (HMI's), we argue is one of the best approaches that could be made with the technology used. The solution is thematic and the pictures make it easy to see the elements and recognize how to use the system between different UI's. A big difference here is the desktop version of MES4, seen in [Figure

8], which has none of these positive aspects, it is created as a standard looking software and we believe this is to create a more generic feeling and allow portability between systems. Our solution has completely different technical possibilities, mainly space and disregard of physical laws (objects can float for an example) which enables us to expand upon these aspects in ways that the physical world never could adhere to, even if we do take inspiration from the physical world. With that argument we think the proper design of a DT compared to a real world PT solution depends on the technology used and in what areas the DT-system aims to improve upon, even its definition will allow/disallow different approaches.

We used colours that were easy to see in AR/VR along with panels to stand out in a physical world. We also tied these colours to the organization enabling for customized branding of the product [Figure 48]. This enables our system to be used with other solutions that already might be tied to the organization, as long as other software are implemented by the same thought-process regarding visuals. Animations were used in the Error Panels that float and the conveyor belt-movements are also animated to show how they move. This became important for a polished feeling and to replicate that we saw a live, moving and real system in action inside the virtual world. Further work should expand upon these aspects with animations since we only had time for smaller parts of this technology, see 11.4 in future work for expanded thoughts.

9.4. VR and AR as technology

9.4.1. Pros/Cons of VR as improvement upon workflow

The absolute greatest problem VR technology stands before is that the advanced hardware often is bulky and introduces a hassle to start and use. Especially if there are alot of putting on/off on the users head. Thus a VR solution would work best where the user is meant to work in the simulated-environment for longer periods of time. For that to be true a more complete software has to be developed, maybe even possibilities to solve problems and use the equipment mapped in the DT system. As technology moves forward VR will see improvements in these fields. We already see HMD's (Head Mounted Displays) that are lighter and have batteries and wireless solutions. As time improves upon these features the ease of use will also be improved. We also argue that VR in many aspects might be obsolete in a couple of years where AR and holographic solutions will be even better, but our work is perfectly ported to AR as mentioned below in 9.4.2. The positive effects we get via VR are presented above in Chapter 8 and we believe these outshine the periodic problems of putting the hardware on.

9.4.2. Pros/Cons of AR as improvement upon workflow

As mentioned in 7.5 AR is really interesting and our technology and software produced here are portable to AR as well. AR is harder to start due to the scanning of reference points for orientation, but when done it does not hinder current workflow if implemented in a discrete manner. In other words we do not talk about hand-held hardware (phones and tablets) but rather HMD technology that acts as glasses where elements are a constant part of the overview and workflow. The problem with AR is, compared to that of the positive side of Desktop and VR, that the tech won't be allowed to be used in a secure location, for AR to really work well the operator has to be in the real environment of the PT-system. We believe this generally isn't a problem but in hazardous environments the solution of a VR-application tied to a DT is better than AR, this is also true if the operator works in another location than the physical asset or provides support from far away.

Compared to Festo's own solution for AR or many other examples of technology that uses a hand-held device we want to clarify that we instead talk about real holographic technology which would improve upon the existing solution, as an example: [Figure 55][13]. The reason this is important and repeated here is that we see that the technology used like this generally hinder the operator by needing a phone or similar as medium, in a real solution the operator needs his/her hands and using AR technology in

this manner only limits the amazing possibilities. Sure can it be used as a quick detailed look in some specific parts of a real system, this is already proved by Festos AR solution [34], but with the factors we focus on here tied to overview and debugging it might not be enough. A phone certainly is powerful enough to render complete simulations but as long as AR is developed to be used in this manner, we don't think the real impact upon current industry solutions won't be as great as it can be.

9.4.3. VR/AR, should it be used?

With support from arguments above we find that depending on the solution we know if a DT is beneficial in given situations. In other words the purpose of software will along with targeted technology define if it should be used or not. We believe that the DT approach should be used in VR since the operator comes as close as possible to the real system and its usage is logic to utilize if we, by any reason, want the user to be away from the PT. An AR solution should not be used, from our viewpoint, with a phone or similar and it should instead be used as a way to complement the reality in a manner where we can show data via new visualizations and this could only be achieved with more advanced holographic technology [13]. The AR technology can only be used in real life if it is discrete and allows a wider field of view, we argue that any hand-held device will hold the operator back and ordinary solutions would be better, such as real interactive displays. The work we have done aims to be proof of concept of this discussion and we believe we have done so and AR is the true goal of most DT-systems because that is the best approach for improving Industry 4.0 tied to overview, understandability and debugging if implemented in the right manner, see 11.8 for future work tied to this.

9.5. Reconnect to the overarching question

What are the advantages and/or disadvantages of monitoring and troubleshooting a Festo CP-Factory by means of a digital twin-driven visualization?

To answer the question we can see that solutions like ours could substantially improve upon, even revolutionize, how we overview, use and handle production systems today. Organizations that move into industry 4.0 already has the backbone needed for this evolution and we can see that our implementation is clearly proving this with the help of a I4.0 solution. VR can use a Digital Twin to improve upon the Physical Twin in manners the real world won't allow, such as testing, abstracting information, making reruns with the help of saved data and evolve how we can visualize data in correlation to the system we overview. AR could even introduce important elements such as screens and text that won't/can't even exist in the real world. If we tie this to holographic technology, already as available today, we can actually improve upon the connection between human and machine: We see this is the way forward for Industry 4.0 and with that said, we implemented some clear improvements upon the Festo-CP-F system. The impact of our work can improve upon other solutions as well because of the generic nature of the software.

We feel that we have found impacting factors to the question we set out to answer and the solution not only feels natural, it is a step we need to take in Industry 4.0, or should this evolution already allow the internal I4.0 step to be called: “Augmented Industry”?

We see fewer disadvantages in our work but one might argue the gain compared to the cost of these evolutions have to be defined, and we agree. This is however outside of this thesis viewpoint and we argue that new solutions that is developed today, and in the future, in a generic fashion should not cost much more than standard generic solutions that has been created previously, as long as the backend of hardware supports I4.0 definitions.

10. Conclusions

In this section we aim to sum the report and present our results tied to our overarching question in a conclusive manner:

What are the advantages and/or disadvantages of monitoring and troubleshooting a Festo CP-Factory by means of a digital twin-driven visualization?

10.1. Purpose of the question

The question is important as industry steps into its next phase, Industry 4.0, see 2.1, and to develop the technology that comes with I4.0 we need new definitions and areas to expand into, Digital Twin-solutions are on the rise ever since NASA presented the expression in 2010. Industry is looking into new ways to represent and overview factories all over the world to give an edge in production and surveillance. The traditional operator with hands-on tasks is now moving towards monitoring and overviewing assignments instead. This requires better visual feedback for the operator to be able to detect and solve anomalies in the production flow, which otherwise could result in costly stops. The reason it is time to move towards digital representation is that systems now have the underlying technology with sensors that allow us to replicate physical assets and what they do into new digital solutions. AR/VR is also on the rise when it comes to technological progress, it's also cheap enough that it is available to the general public which makes the invention more logical to use in new ways. We targeted the Festo Cyber Physical Factory since the platform is made for research and learning. It is also a smaller system that is logical to map in these ways under our timespan and it proved it could be used as a springboard for greater, real production, solutions and by working in this way we could expand our work towards other solutions as well.

The focus of the question lies in how we could improve upon existing solutions for overview and debugging and how we could visualize data in a fashion that enables a better solution than that of MES4, which is the software solution that comes with Festo CP-F. Thus would our report and improvements only focus upon how we can see data live and problems that arise as production halts and how to recover from these.

10.2. Overview of the thesis and its work

We defined the question as seen above in 4.1, and discussed upon what needed to be achieved. For this overview we instead break down the report in lesser overarching questions here regarding our work and the process for direct understandability and answers in a streamlined manner:

1. Which definition of digital twin would be applicable to our work, taking Festos existing definitions presented from Chapter 3 in mind?

The definitions compared by Festo towards their own softwares (such as CIROS), see 3.1, were not enough for what we wanted to achieve, thus we extended the previous definition in 4.2 and there are well founded reasons for this approach. We could via the definition keep our solution within reasonable limitations, and since the purpose of this work focuses on overview, understandability and recovery from faults, we created the definition to have these aspects in mind. One of the greatest possibilities discovered is that the extended definition allows alterations of a DT-system if it would bring positive side effects. This was utilized as we allowed the user/operator to abstract away parts of the DT-system (something that would, technically, not be allowed by a stricter definition) to increase overview capabilities, see [Figure 44] compared to [Figure 45]. Our definition also allowed expanded portability between solutions such as AR/VR or Desktop from the same solution because of the generic approach.

2. Is OPC the best solution compared to other protocols for transferring data from an active system to a digital twin and why?
    a. Yes: Does OPC (Open Platform Communications, standardized communication protocol) produce extensive and relevant data we can reach and implement?
        i. Yes: How do we access this data and sort it in the most effective way?
        ii. No: How do we capture the data in an effective way to achieve our goal?
    b. No: What is the alternative and why is it better?

As apparent in 7.1 and 9.2 we went with OPC since the CP-F-system already had support for this approach and in the end it allowed for some smart solutions such as saved runs, see 7.2.4 and expandability/maintainability by using a settings-file, see 7.2.5. The OPC protocol is moving towards being the standard communications protocol of Industry 4.0 and many equipment vendors support it, such as in this case does Siemens have the OPC server built straight into the PLC. By making this exclusive protocol choice we never explored answers to the no-questions above and their theoretical answers will not be found in this report.

3. What are the exact goals that the visualization needs to achieve to aid overview and debugging?

The thesis used DRM (see 5.2 for explanations and 7.3 for the iterations themself), as a research methodology because of the interactive design-tag that comes with "visualization" along with VR/AR in the main question. We felt this was the way to go as implementation and experience from these implementations could drive our work forward if it was grounded on discussion and previous work. The focus upon our research was overview and debugging so the visualization tried to focus upon this as we implemented and corrected ideas throughout the DRM-iterations. We managed 5 iterations in total, if we had more time it would probably have continued towards the double. We did feel we had enough information gained and that our software was even beyond proof of concept so this never hindered the thesis in any fashion.

The goals we found that the software had to focus upon were: Clear errors and how they should be recovered from. Overview should be made in an intuitive and modifiable manner where grouping of data feels natural. The software should be easy to use and be as intuitive as possible to allow less misunderstanding and lengthy education to both how our software worked, but also the physical twin.

4. How would an effective solution look like and function?
    a. We needed to implement different solutions as is procedure with DRM-research
    b. Test them in a controlled manner each iteration and analyze success and failure parameters
    c. Then present our solution compared to the same success parameters.

This is where the DRM helped by allowing us to implement and with the data produced in the Prescriptive I and discussion that followed in Prescriptive II in each step we could define successful and unsuccessful parameters in our solution. The testing was done after each DRM implementation and the implementation itself was done in the Descriptive step of the DRM phases. The presentation of the work is done here in this section and presented in detail in Chapter 8 and discussed upon in Chapter 9.

5. How does the Digital Twin-solution compare to the traditional system for monitoring the factory?
    a. Improvement: In what way would our solution be better than the existing solutions?
    b. Downsides: What are the negative aspects with a VR/AR solution compared to the existing solutions?

We also needed to define how the Festo solutions were handling the areas of interest compared to our Digital Twin and by doing this we actually found substantial problems that we tried to make improvements upon. We mapped these as can be seen above in question 3 and 4 and it allowed a comparison between our solution and the original as presented in Chapter 8 and discussed upon in Chapter 9. We have uncovered that if we focus upon overview and debugging our system has a better approach towards representation of data than the current MES4 solution. The reasoning behind the solution MES4 went with is that its a desktop application also meant to steer and control the physical twin, that is its focus, not overview and logical choices tied to overview of the live system. We also reinvented how we can showcase data, Festo has an amazing approach where they use pictures [Figure 16] with nodes attached to HMIs to improve upon explicit data. With this knowledge a DT-solution is the absolute best approach and we mimicked this by animating UI-elements (arrows), [Figure 47], on top of the conveyor belts as they were in the real world. We also made similar stations as those in [Figure 16] to translate carriers in the digital twin, this enabled historical data and orientations of carriers and created cards at each storage Station in the ASRS32 module. A great opportunity came with this approach as we could now follow production orders as it passes through the flow. This means placement of elements becomes important for grouping data in digital twins and keeping explanations to a minimum. To the extent we did this in our solution would not be possible without VR/Desktop and a digital twin or AR-solutions attached to the physical asset. When using the watch-tables in the original solution [Figure 9] we can not see what each data-node means (this is the underlying OPC-server) and our implementation instead focused on a feed that shows what the machine is doing as events occur (the feed have a string tied to events that explains what the machine does instead of raw data, see [Figure 48] as an example). This is not possible in a clear manner in the original solution.

10.3. Summation of the results and its impact

Since we achieved our goals we answered the overarching question with: The impact could be substantial by linking a Digital Twin-solution with a Physical Twin. Since we use a extended definition that allows us to alter reality, if there are good reasons behind the choice, we can see great improvements in: portability among different software solutions (one software can do it all, AR/VR and Desktop) and overview becomes easier and clearer due to the increased possibilities within a digital solution compared to reality.

The most important results produced are the proof of concept we made towards the usefulness of digital twins in a short timeplan. We produced a software that actually works as intended in a team of two in around two months and the work produced data that we aim to use in next phases, presented below in Chapter 11, Future Work. Since I4.0 is in full bloom the industry is rushing towards theoretical solutions on how we could use the technology to further improve upon manufacturing and production. Our work uses CPF (Cyber Physical Factory) solutions along with Unity 3D, a game engine, to create a simple solution that in very few steps can increase understandability and overview of real systems in simple manners. As seen in our implementation in 7.4 our solution takes a physical system into VR which allows us to redefine laws of how we present data that won't be allowed by a standard desktop-solution. The real impact upon our work comes from the availability to implement the UI changes into the real world via AR-solutions, not mainly via handheld devices, but real holographic technology [13]. As seen in 7.5 we can easily port our work to AR solutions which in turn allows for the implementation to be used in the real world as holographic extensions to real hardware, the connection between human and machine have never been greater.

If we instead focus on the VR tied Digital twin solution we created, it allows saving live data to CSV-files and this proved to have a massive impact upon the value of our work. Since we had saved previous runs of production flow and we could open a stream from these files, not only live connections via OPC, we gained some amazing side effects. Our solution could be used to find faults in previous runs which are important from a debugging, understandability and future prevention-perspective after the fault has occurred. We also gained the possibility to test, alter and modify previous runs for increased possibilities, imagine the security-testing aspects alone. We know that systems often do these kinds of logs, they are imperative in any production flow but now we can

visualize those logs via a digital twin, which means the operator can see what has happened in a simulation as close to the real world as possible, not only view tables of data as were standard previously. Another positive aspect of DT-solutions are support and hazardous environments, imagine that an expert is sitting far from the physical unit, by connecting a digital twin like ours to the physical counterpart we allow the expert to be as close to reality as possible, just with a connection online in real time and here does the overview and debugging focus of our software really shine.

We feel that intuitive and informative visualizations of anomalies in a productional flow can reduce downtimes and in the long run keep costs down. Operators that monitor machinery can benefit from the extended information our system can provide in terms of guidance and directives in the work environment. Inexperienced operators will have a higher probability of recovering from errors in the workflow and might not even need to leave the workstation to find help if necessary. Help could be found virtually nearby.

11. Future Work

There is still work to be done when it comes to the pure software to make it as complete as possible, this has been the second most important part since we wanted to focus on the research first and proof of concept, which we argue that we have achieved as discussed in Chapter 8 and Chapter 9. But below follow some important and interesting steps to further the research and develop the software and by that achieve an even better and more complete representation compared to our extended definition of a digital twin.

11.1. Complete mapping of the factory

It's imperative that we continue to map the factory so all modules can be present on the network and in the software at the same time. This was also one of our main goals but we did not have time to map each unit since it takes a lot of time to understand each unit's underlying systems.

To map every relevant node in a Festo OPC server and create extensive documentation over its structure for future use is also an important next step. It would make our software more extensive if it could use all relevant data. When it comes to documentation of the Festo assets this could create an easier workflow for other researches or extensions made to this application. It would also mean that mapping of new assets could be done in a fraction of the time due to the naming conventions.

11.2. Playback tools

To create a tool for pausing, playing, loading, rewinding and moving forward in the timeline when using the log files (CSV-mode) inside of the software would be really handy for the user and strengthen the possibilities with replayability. This would make specific moments of a previous run easier to get to, and also allow replayability without restarting and waiting for the overviewed moment to pass by again. To clarify we would like to implement ordinary functions like any video/music-player has for movement in timelines.

11.3. Audio

The Festo CPS produces audible alarms to make operators aware of states that deviate from normal operation (mainly the RASS unit). Alarms provide a good complement to the visual ones displayed on the HMI panels. Although in a scenario with a production grade factory, the environment would most certainly be a lot more noisy than our test facility. This as well as the use of hearing protection could suppress the alarms, making it go unnoticed. Our solution could benefit from the use of audible alarms but could do so in a more controlled way. Surrounding noise would be filtered out due to headsets, making it a better working environment for the operator and the sound frequencies can be more user friendly selected. Also as a guidance for solving problems, speech directives can be applied to aid the user.

11.4. Animation

According to our description the digital twin should mimic the physical counterpart to the greatest degree possible, this would also mean that it should move and behave in the same way. This can be done via animation and object translation work. This would also bring meaning to the objects (modules) that exist in the virtual space. We argue that it would enhance the overview of the system in an improved way and it was first one of our main goals, but due to the complex mapping section of our work we did not have time to implement these animations. We mainly wanted to represent where the carriers were and how belts and robots moved in world space.

11.5. Expand the settings functionality

We want to even increase the impact of the settings file for improved functionality and expandability, see 7.2.5. For example each OPC-node section that has a bool or string value should have fields in the file for comments output to the software. In other words the operator could change what text comes up in the feed when the node changes value. This would enable the operator to also customize completely new nodes and produce the logic straight from the settings file. We could also modify sensor output in the environment depending on the implementation. Let's say for an example that a sensor handles some logic today, the same might not be true if the hardware gets updated in some fashion. We could also reduce the size of the source code because as it is implemented now it has to listen for all types of nodes. By doing this we could shrink this behaviour and only listen for the nodes that change some logic in the simulation (such as belt movement, carrier positions and station logic).

We also want to produce a simpler software that handles the settings via a GUI with buttons and so on to make the solution feel more worked, we could also secure the integrity of the file by doing this by having a predefined structure be produced by the software in a consistent manner. As is now with the setting file the operator needs to be careful not to break the structure, this could render the software unusable. A GUI would also enable us to save data in different files and group the data in a more logical fashion with tabs and tables. This would also enable security in a completely different manner where login and encryption of the file could be enabled.

11.6. Build Tools

Expand the tool in Unity 3D that enables the user to modify factory positions and iterations inside the software itself, this would greatly extend its usage to all possible Festo-solutions out there. In other words this would make our software generic and all Festo CP-F solutions could use the software without changing the source code. This is expanded upon in 11.7 since we want to take this thought process even further and beyond only Festo-solutions.

11.7. Complete tool-suite for creating generic digital twins

We have proved that it is possible to create DT's with software that is not primarily made for it via Unity 3D. Imagine a tool that's similar to the game engine but instead purely made for the construction of industrial twins in a generic and object oriented fashion. Some of these tools already exist, but most if not all focus on specific systems or other fields of twins. We want to create a system that enables overview and creation of DT's in AR/VR and traditional Desktop at the same time no matter what systems are used in the background. A solution that has support for plugins for expandability and enables older machines to be imported to the digital era via simple and supported sensors which would be beneficial to most industrial organizations since it is the cheapest option. Just the possibilities to replace hundreds of different software with one complete solution would bring overview (and if supported: control) of different systems to a breeze. We argue that the impact of our research gave us knowledge that would be beneficial in the creation of a complete solution and that next-generation would benefit from a do-it-all software tied to DT´s. We strongly believe this is possible as we use the OPC protocol here, but other solutions connected might not only use one solution and thus interfaces

would have to be created. Nothing hinders us from bringing different systems together and building logic depending on what sensor and data we get from the PT-system, this could be the future of how we bring equipment together from different manufacturers in a cheap, simple and intuitive manner in Industry 4.0 or generations to come.

11.8. Holographic work, the real power of the technology

As we have discussed in 7.5 and 9.4.2 the impact of Augmented Reality solutions could be revolutionary. To develop enhanced overlays to physical assets via HoloLens [13] or similar technology would connect humans to machines in a completely new and intuitive manner. Nothing hinders this technology to be used with hand held devices as well, but we believe for real impact to be made in real industry we need some sort of head mounted screen (like similarities to glasses) to let the outfit and gear of any operator to be as close to as is today [Figure 55].

Unity 3D has libraries for HoloLens and it allows us to port all of our work in a similar manner as the current AR conversion has done. Tool-suite's, see 11.7, would need similar support for the hardware to have impact upon industry in the manner we aim upon were Desktop and AR/VR is built-in seamless.

11.9. Create a duplex connection between the twins

While the expanded definition in 4.2 won't need two way connections we strongly believe that a third mode is valuable: Control Mode which enables a connection that controls the physical twin from the digital solution, it would be good for usability in real industry. Most of our discussion here could be expanded upon with the possibility to enable control but implementations need to be aware of and update security to defend their systems (this is also true with any software solution tied to I4.0 systems). For example could the operator click on images representing an order in the digital environment to allow the real system to produce said order. Imagine that the operator even connects different pieces in HoloLens and then pushes the product to the digital twin and the physical solution creates that order, all this is possible due to the proof we made in this report along with MES4 SQL query logic.

12. References

[1] E. Negri, L. Fumagalli and M. Macchi, "A Review of the Roles of Digital Twin in CPS-based Production Systems", Procedia Manufacturing, vol. 11, pp. 939-948, 2017. Available: 10.1016/j.promfg.2017.07.198 [Accessed 15 April 2020].

[2] J. Jasperneite, Was hinter Begriffen wie Industrie 4.0 steckt, Internet Und Autom. 12 (2012) 12.

[3] M. Eckhart, A. Ekelhart and E. Weippl, "Enhancing Cyber Situational Awareness for Cyber-Physical Systems through Digital Twins", *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2019. Available: 10.1109/etfa.2019.8869197.

[4] "CP Factory | Festo USA", Festo.com, 2020. [Online]. Available: https://www.festo.com/us/en/e/technical-education/learning-systems/factory-automation-and-industry-4-0/learning-systems-industry-4-0/cp-factory-id_36140/. [Used: 24- Feb- 2020].

[5] J. Ríos, J.C. Hernandez, M. Oliva, F. Mas, Product Avatar as Digital Counterpart of a Physical Individual Product : Literature Review and Implications in an Aircraft, in: ISPE CE, 2015: pp. 657–666.

[6] M. Schluse, J. Rossmann, From Simulation to Experimentable Digital Twins, in: Syst. Eng. (ISSE), 2016 IEEE Int. Symp., 2016: pp. 1–6.

[7] A. Canedo, Industrial IoT Lifecycle via Digital Twins, in: Proc. Elev. IEEE/ACM/IFIP Int. Conf. Hardware/Software Codesign Syst. Synth., 2016: p. 29.

[8] G.N. Schroeder, C. Steinmetz, C.E. Pereira, E.D. B., Digital Twin Data Modeling with AutomationML and a Communication Methodology for Data Exchange, in: IFAC-PapersOnLine, Elsevier B.V., 2016: pp. 12–17. doi:10.1016/j.ifacol.2016.11.115.

[9] D. Gorecky, M. Schmitt, M. Loskyll and D. Zuhlke, "Human-machine-interaction in the industry 4.0 era", 2014 12th IEEE International Conference on Industrial Informatics (INDIN), 2014. Available: 10.1109/indin.2014.6945523 [Accessed 21 May 2020].

[10] "Home Page - OPC Foundation", *OPC Foundation*, 2020. [Online]. Available: https://opcfoundation.org/. [Accessed: 15- Apr- 2020].

[11] "Ciros VR", Festo.com, 2020. [Online]. Available: https://www.festo.com/net/SupportPortal/Files/492986/DID1165en_CIROS_VR_A4_rev1.pdf. [Used: 24- Feb- 2020].

[12] "Festo Didactic InfoPortal", Ip.festo-didactic.com, 2020. [Online]. Available: https://ip.festo-didactic.com/InfoPortal/AR/EN/index.html. [Accessed: 23- May- 2020].

[13] "Microsoft HoloLens | Mixed Reality Technology for Business", Microsoft.com, 2020. [Online]. Available: https://www.microsoft.com/hololens/. [Accessed: 25- May- 2020].

[14] R. Bolton et al., "Customer experience challenges: bringing together digital, physical and social realms", Journal of Service Management, vol. 29, no. 5, pp. 776-808, 2018. Available: 10.1108/josm-04-2018-0113 [Accessed 15 April 2020].

[15] R. Söderberg, K. Wärmefjord, J. Carlson and L. Lindkvist, "Toward a Digital Twin for real-time geometry assurance in individualized production", CIRP Annals, vol. 66, no. 1, pp. 137-140, 2017. Available: 10.1016/j.cirp.2017.04.038 [Accessed 15 April 2020].

[16] G.Bacchiega, "Embedded digital twin", Slideshare.net, 2017. [Online]. Available: https://www.slideshare.net/gbacchiega/embedded-digital-twin-76567196. [Accessed: 15- Apr- 2020].

[17] J. Persson, J. Norrman, “Virtual Production Line -Virtual Commissioning,” M.S. Thesis, Division of Production and Materials Engineering, Lund Univ., Lund, 2018. Accessed on: May., 18, 2020. [Online]. Available: http://lup.lub.lu.se/student-papers/record/8949117/file/8949129.pdf

[18] J. Polcar, P. Horejsi, P. Kopecek and M. Latif, "Using Unity3D as an Elevator Simulation Tool", DAAAM Proceedings, pp. 0517-0522, 2017. Available: 10.2507/28th.daaam.proceedings.073 [Accessed 19 May 2020].

[19] H. Haskamp, M. Meyer, R. Mollmann, F. Orth and A. Colombo, "Benchmarking of existing OPC UA implementations for Industrie 4.0-compliant digitalization solutions", 2017 IEEE 15th International Conference on Industrial Informatics (INDIN), 2017. Available: 10.1109/indin.2017.8104838 [Accessed 19 May 2020].

[20] V. Kuts, T. Otto, T. Tähemaa and Y. Bondarenko, "Digital Twin Based Synchronised Control And Simulation Of The Industrial Robotic Cell Using Virtual Reality", Journal of Machine Engineering, vol. 19, no. 1, pp. 128-144, 2019. Available: 10.5604/01.3001.0013.0464 [Accessed 21 May 2020].

[21] C. Turner, W. Hutabarat, J. Oyekan and A. Tiwari, "Discrete Event Simulation and Virtual Reality Use in Industry: New Opportunities and Future Trends", *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 6, pp. 882-894, 2016. Available: 10.1109/thms.2016.2596099.

[22] M. Kostolani, J. Murin and S. Kozak, "Intelligent predictive maintenance control using augmented reality", *2019 22nd International Conference on Process Control (PC19)*, 2019. Available: 10.1109/pc.2019.8815042.

[23] B. Rüttimann and M. Stöckli, "Lean and Industry 4.0—Twins, Partners, or Contenders? A Due Clarification Regarding the Supposed Clash of Two Production Systems", *Journal of Service Science and Management*, vol. 09, no. 06, pp. 485-500, 2016. Available: 10.4236/jssm.2016.96051.

[24] Säfsten, K. och Gustavsson, M. (n.d.). *Forskningsmetodik för ingenjörer och andra problemlösare*. 1st ed. Studentlitteratur AB, Lund.

[25] L. Blessing, A. Chakrabarti and L. Blessing, *DRM, a design research methodology*. Dordrecht: Springer, 2009.

[26] M. Arvola, *Interaktionsdesign och UX*, 1st ed. Lund: Studentlitteratur, 2014.

[27] P. Pavliscak, *Emotionally intelligent design*. Sebastopol, CA: O'Reilly Media, 2019.

[28] D. Shen and Y. Zhou, "Effect of VR Interactive Design System on Visual Feedback and Operational Experience", 2017 10th International Symposium on Computational Intelligence and Design (ISCID), 2017. Available: 10.1109/iscid.2017.190.

[29] C. Linn, S. Bender, J. Prosser, K. Schmitt and D. Werth, "Virtual remote inspection — A new concept for virtual reality enhanced real-time maintenance", *2017 23rd International Conference on Virtual System & Multimedia (VSMM)*, 2017. Available: 10.1109/vsmm.2017.8346304.

[30] S. Profanter, A. Tekat, K. Dorofeev, M. Rickert and A. Knoll, "OPC UA versus ROS, DDS, and MQTT: Performance Evaluation of Industry 4.0 Protocols", 2019 IEEE International Conference on Industrial Technology (ICIT), 2019. Available: 10.1109/icit.2019.8755050 [Accessed 18 April 2020].

[31] P. Marcon et al., "Communication technology for industry 4.0", 2017 Progress In Electromagnetics Research Symposium - Spring (PIERS), 2017. Available: 10.1109/piers.2017.8262021 [Accessed 18 April 2020].

[32] "OPC UA .NET! The official UA .NET Stack and Sample Applications from the OPC Foundation", Opcfoundation.github.io, 2020. [Online]. Available: http://opcfoundation.github.io/UA-.NETStandard/. [Accessed: 26- May- 2020].

[33] "Prosys OPC UA Browser - Prosys OPC", Prosysopc.com, 2020. [Online]. Available: https://www.prosysopc.com/products/opc-ua-browser/. [Accessed: 24- May- 2020].

[34] "Festo Didactic InfoPortal - Hardware Overview", Ip.festo-didactic.com, 2020. [Online]. Available: https://ip.festo-didactic.com/InfoPortal/CPFactoryLab/hardware/index.php?lang=en [Accessed: 23- May- 2020].

[35] Festo-didactic.com, 2020. [Online]. Available: https://www.festo-didactic.com/ov3/media/customers/1100/8039313_deen_v0.9_lp8040796_mps_d_robot_station_complete_manual.pdf. [Accessed: 26- May- 2020].

[36] "MES4 - MPS® The Modular Production System - Learning Systems - Festo Didactic", Festo-didactic.com, 2020. [Online]. Available: https://www.festo-didactic.com/int-en/learning-systems/mps-the-modular- production-system/mes4.htm?fbid= aW50LmVuLjU1Ny4xNy4xOC41ODUuNTM3NjA. [Accessed: 24- May- 2020].

[37] V. Zammitto, "Visualization Techniques In Video Games," Electronic Visualisation and the Arts (EVA 2008), pp. 267–276, Jul. 2008.

Appendices

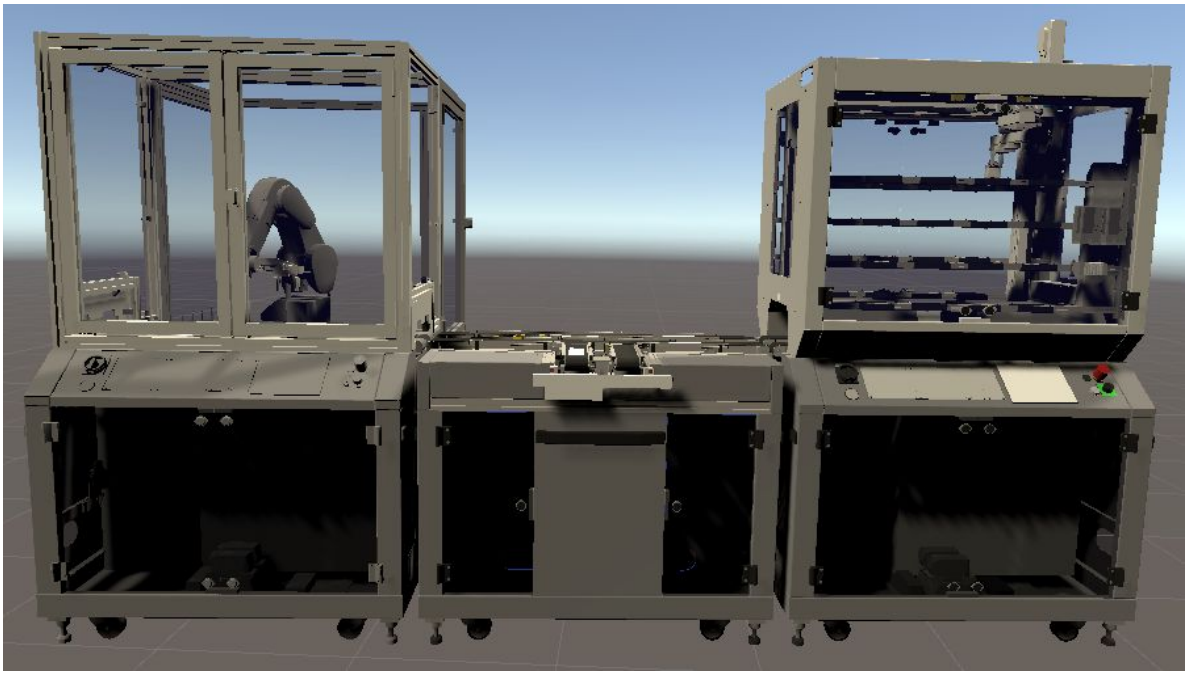

Figure 4, Festo pure DT:
To the right we can see the ASRS32 unit with mounting rows where all pallets, see [Figure 59], normally lay in storage. The arm is also visible in the top right inside the unit. In the middle we have the BRANCH without its top mounted measuring unit. We can also see the two belts that act as input/output to the system facing the camera. To the left we have the RASS with its Mitsubishi robotic arm, this mesh is missing its back behind the arm here (plexiglass as cover, PCB-storage box along with tubes with fuses that feed the arm and its different head-tools).

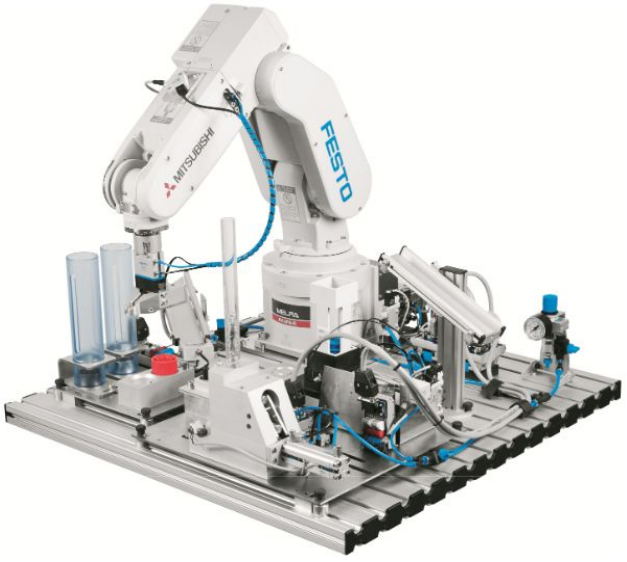

Figure 5, the Mitsubishi arm used by RASS
An overview of the arm used in the RASS unit. The picture is owned by and taken from Festo-didactic [35].

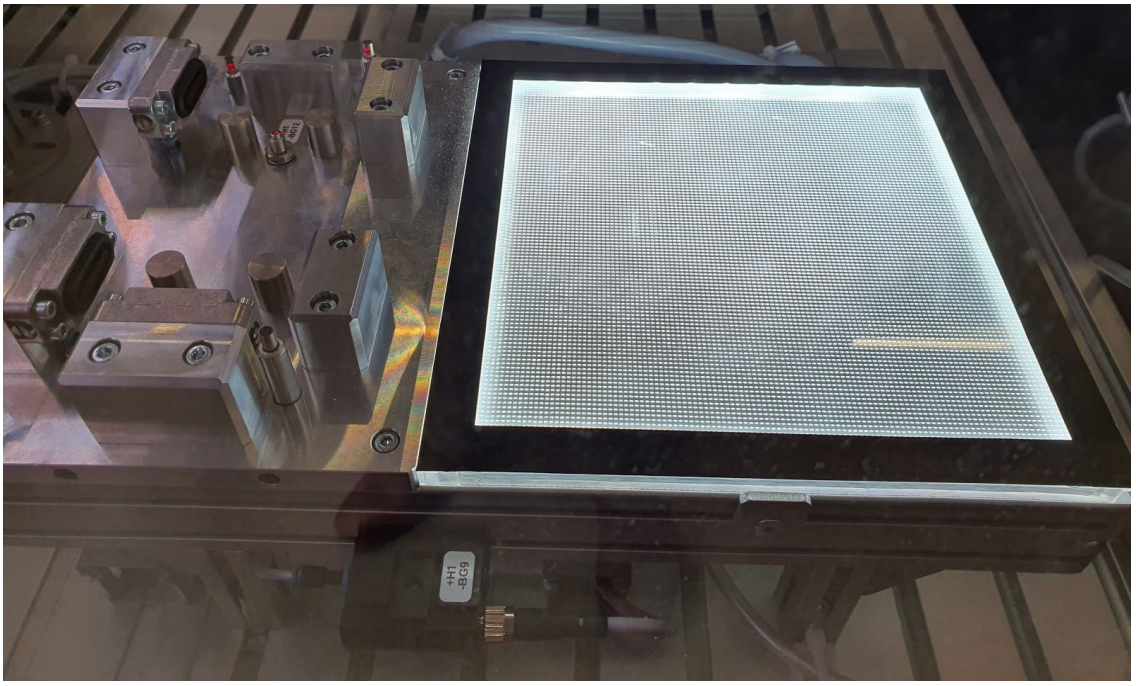

Figure 6, The Lightboard:
The physical asset (RASS) has a mounting station where a camera can read its positioning via a camera. To the left is the assembly-station where it will mount the PCB and fuses. We can also see one of the white labels in the bottom of the screen, these were important to understand the underlying sensors.

NodeID	DisplayName	Value	Datatype	Timestamp	StatusCode
ns=3;s="dbRfidData"."ID2"."iCarrierID"	iCarrierID	0	Opc.Ua.DataValue	2020-05-08 12:53:47:864	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xK1_MB30"	xK1_MB30	True	Opc.Ua.DataValue	2020-05-08 12:53:47:864	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xG1_BG36"	xG1_BG36	False	Opc.Ua.DataValue	2020-05-08 12:53:47:865	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xG1_BG37"	xG1_BG37	False	Opc.Ua.DataValue	2020-05-08 12:53:47:865	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xG1_BG20"	xG1_BG20	False	Opc.Ua.DataValue	2020-05-08 12:53:47:865	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xG1_BG21"	xG1_BG21	False	Opc.Ua.DataValue	2020-05-08 12:53:47:865	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xG1_BG22"	xG1_BG22	False	Opc.Ua.DataValue	2020-05-08 12:53:47:866	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xG1_BG23"	xG1_BG23	False	Opc.Ua.DataValue	2020-05-08 12:53:47:866	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xG1_BG24"	xG1_BG24	False	Opc.Ua.DataValue	2020-05-08 12:53:47:866	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xG1_BG26"	xG1_BG26	False	Opc.Ua.DataValue	2020-05-08 12:53:47:866	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xG1_BG27"	xG1_BG27	False	Opc.Ua.DataValue	2020-05-08 12:53:47:867	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xG1_BG30"	xG1_BG30	True	Opc.Ua.DataValue	2020-05-08 12:53:47:867	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xG1_BG31"	xG1_BG31	False	Opc.Ua.DataValue	2020-05-08 12:53:47:867	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xG1_BG32"	xG1_BG32	False	Opc.Ua.DataValue	2020-05-08 12:53:47:868	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xG1_BG33"	xG1_BG33	False	Opc.Ua.DataValue	2020-05-08 12:53:47:868	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xG1_BG34"	xG1_BG34	False	Opc.Ua.DataValue	2020-05-08 12:53:47:868	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xU1_BG46"	xU1_BG46	True	Opc.Ua.DataValue	2020-05-08 12:53:47:869	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xU1_BG47"	xU1_BG47	False	Opc.Ua.DataValue	2020-05-08 12:53:47:869	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xU1_BG50"	xU1_BG50	False	Opc.Ua.DataValue	2020-05-08 12:53:47:869	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xU1_BG51"	xU1_BG51	True	Opc.Ua.DataValue	2020-05-08 12:53:47:869	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xU1_BG52"	xU1_BG52	False	Opc.Ua.DataValue	2020-05-08 12:53:47:870	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xU1_BG53"	xU1_BG53	False	Opc.Ua.DataValue	2020-05-08 12:53:47:870	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xU1_BG54"	xU1_BG54	False	Opc.Ua.DataValue	2020-05-08 12:53:47:871	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xU1_BG55"	xU1_BG55	False	Opc.Ua.DataValue	2020-05-08 12:53:47:871	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xU1_BG56"	xU1_BG56	False	Opc.Ua.DataValue	2020-05-08 12:53:47:871	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xU1_BG57"	xU1_BG57	False	Opc.Ua.DataValue	2020-05-08 12:53:47:871	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xU1_MB50"	xU1_MB50	True	Opc.Ua.DataValue	2020-05-08 12:53:47:872	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xU1_MB51"	xU1_MB51	False	Opc.Ua.DataValue	2020-05-08 12:53:47:872	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xU1_MB52"	xU1_MB52	False	Opc.Ua.DataValue	2020-05-08 12:53:47:872	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xU1_MB53"	xU1_MB53	True	Opc.Ua.DataValue	2020-05-08 12:53:47:872	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xU1_MB54"	xU1_MB54	False	Opc.Ua.DataValue	2020-05-08 12:53:47:873	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="xU1_MB55"	xU1_MB55	True	Opc.Ua.DataValue	2020-05-08 12:53:47:873	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="dbRfidCntr"."ID1"."xDone"	xDone	True	Opc.Ua.DataValue	2020-05-08 12:53:47:873	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="dbRfidCntr"."ID1"."xBusy"	xBusy	False	Opc.Ua.DataValue	2020-05-08 12:53:47:874	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="dbRfidCntr"."ID1"."iLen"	iLen	26	Opc.Ua.DataValue	2020-05-08 12:53:47:874	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="dbRfidData"."ID1"."iCarrierID"	iCarrierID	3	Opc.Ua.DataValue	2020-05-08 12:53:47:874	Opc.Ua.Client.MonitoredItemStatus
ns=3;s="dbRfidData"."ID1"."iCode"	iCode	2	Opc.Ua.DataValue	2020-05-08 12:53:47:875	Opc.Ua.Client.MonitoredItemStatus

Figure 7, CSV file:
This figure represents how the data is logged in tables, this is data from the ASRS32 module. Here can we also see the speed in which data can be parsed into the machine if we focus on the Timestamp column.

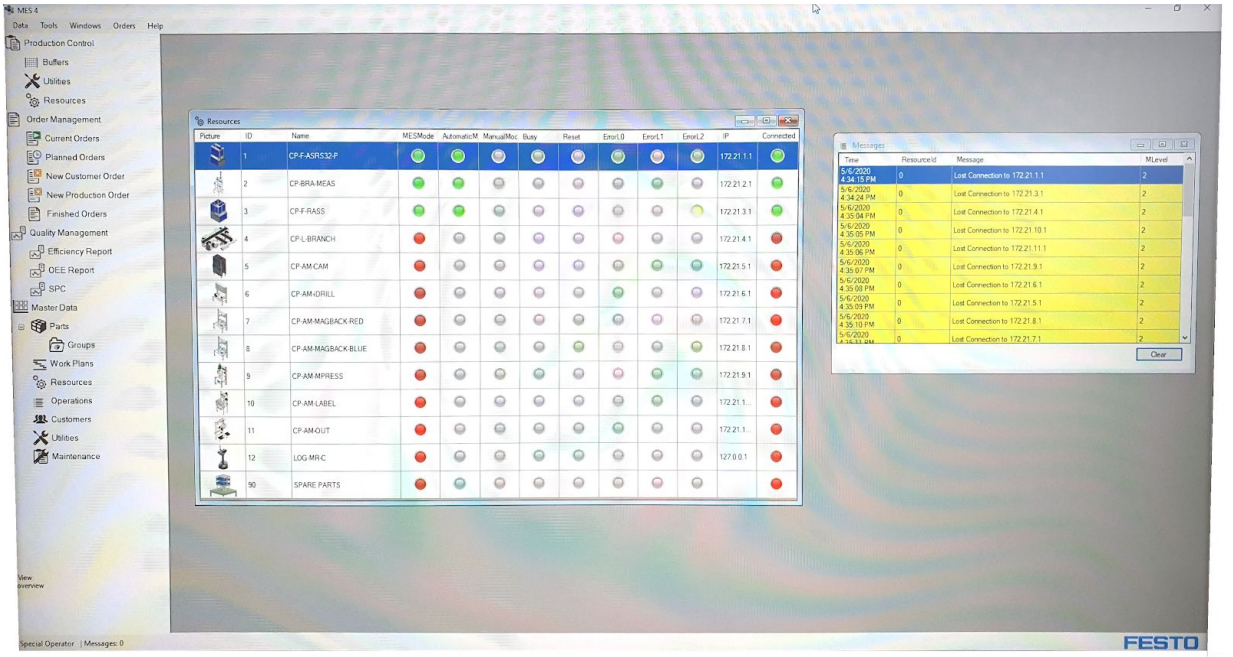

Figure 8, MES4 - Main window of the MES4 software:

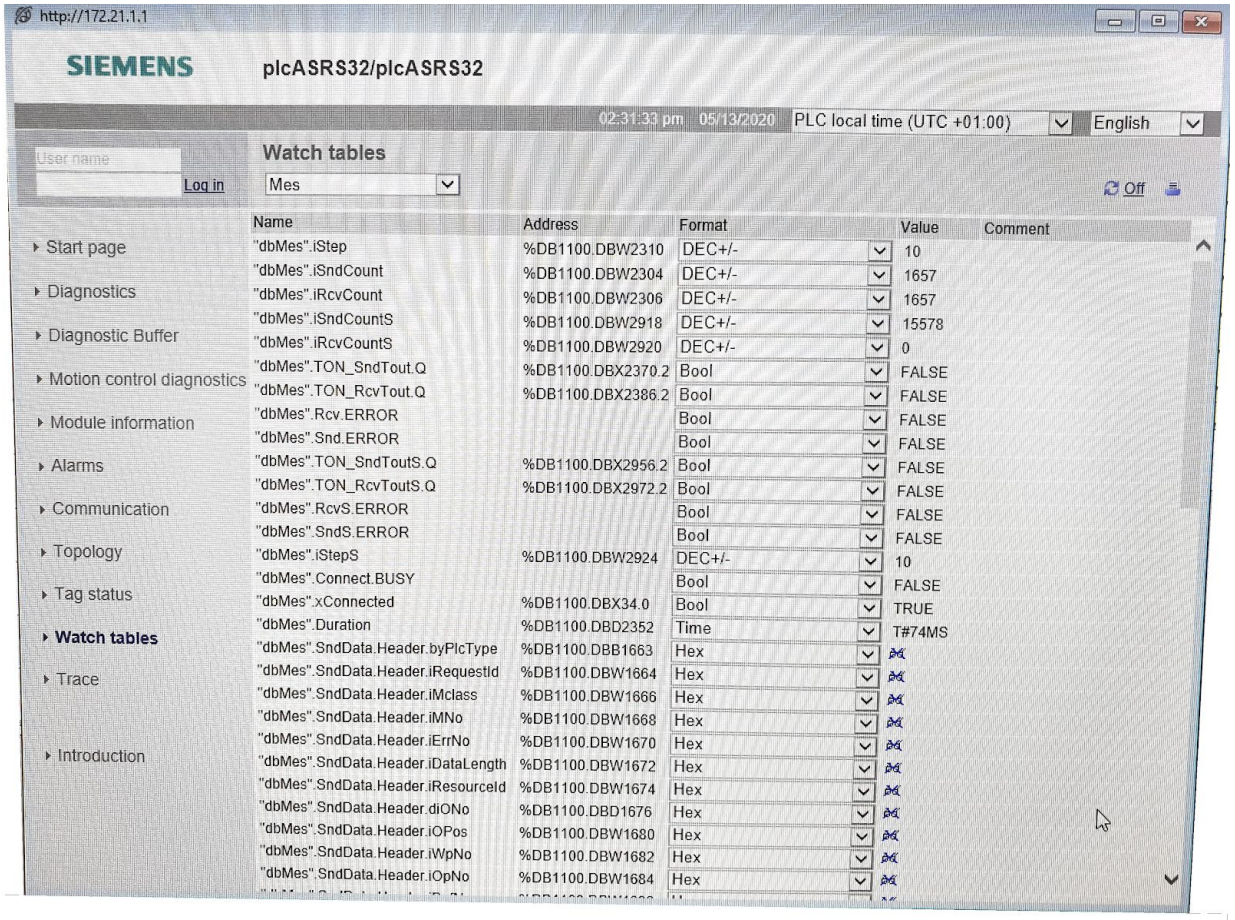

Figure 9, MES4 - Watch tables:

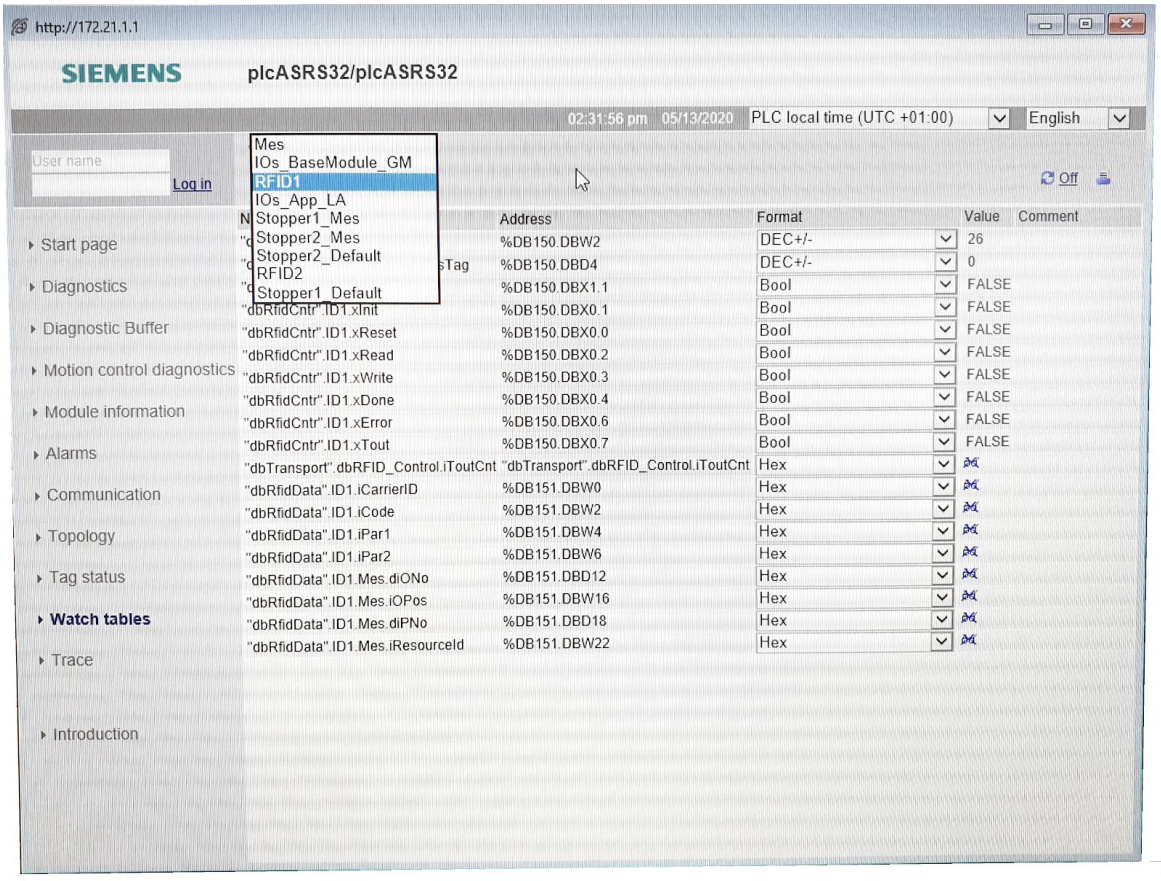

Figure 10, MES4 - The watch table dropdown menu:

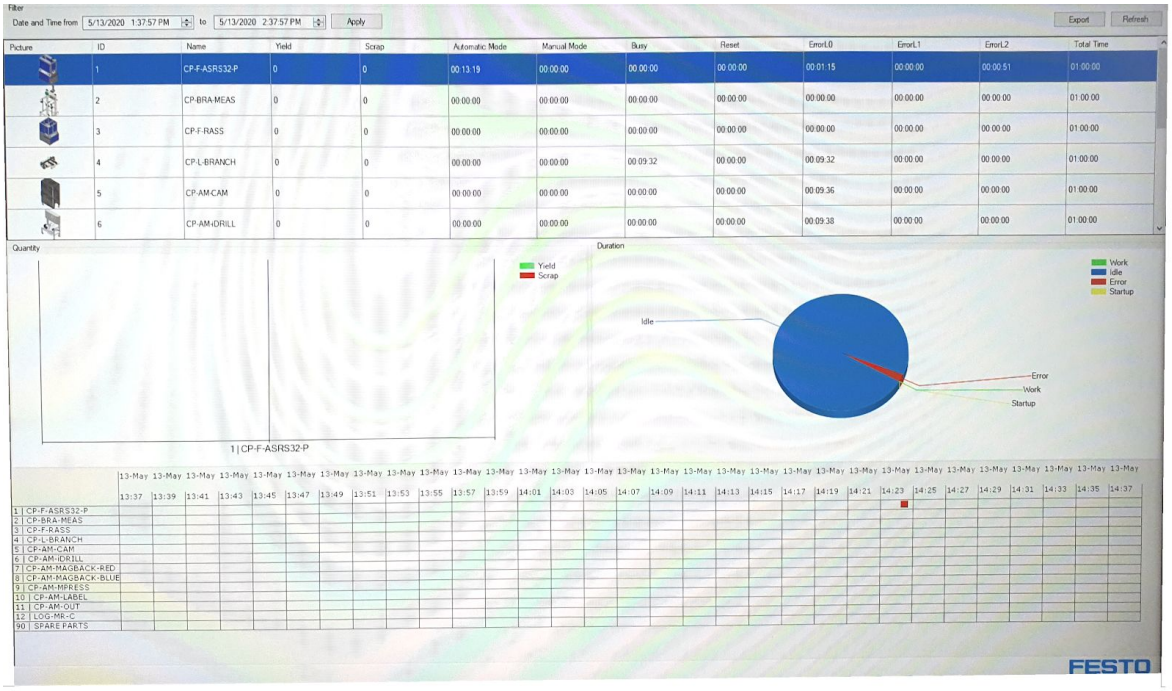

Figure 11, MES4 - Historical data:

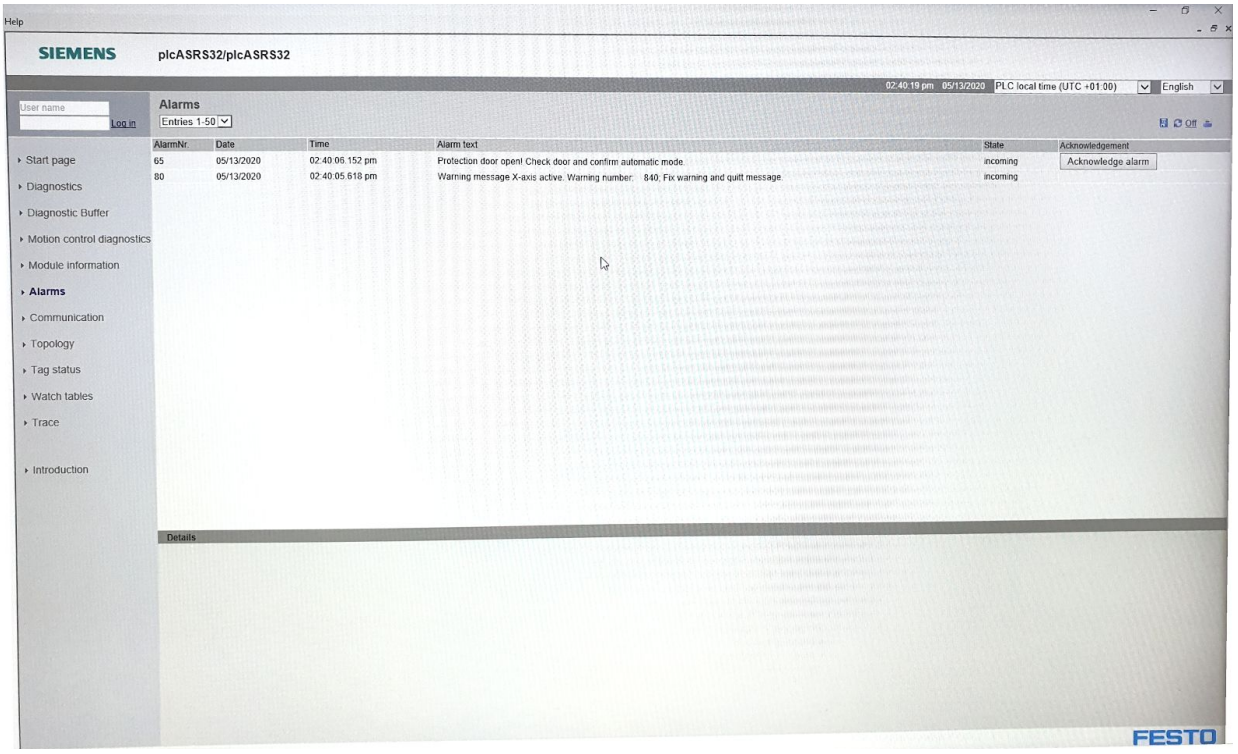

Figure 12, MES4 - Error screen:

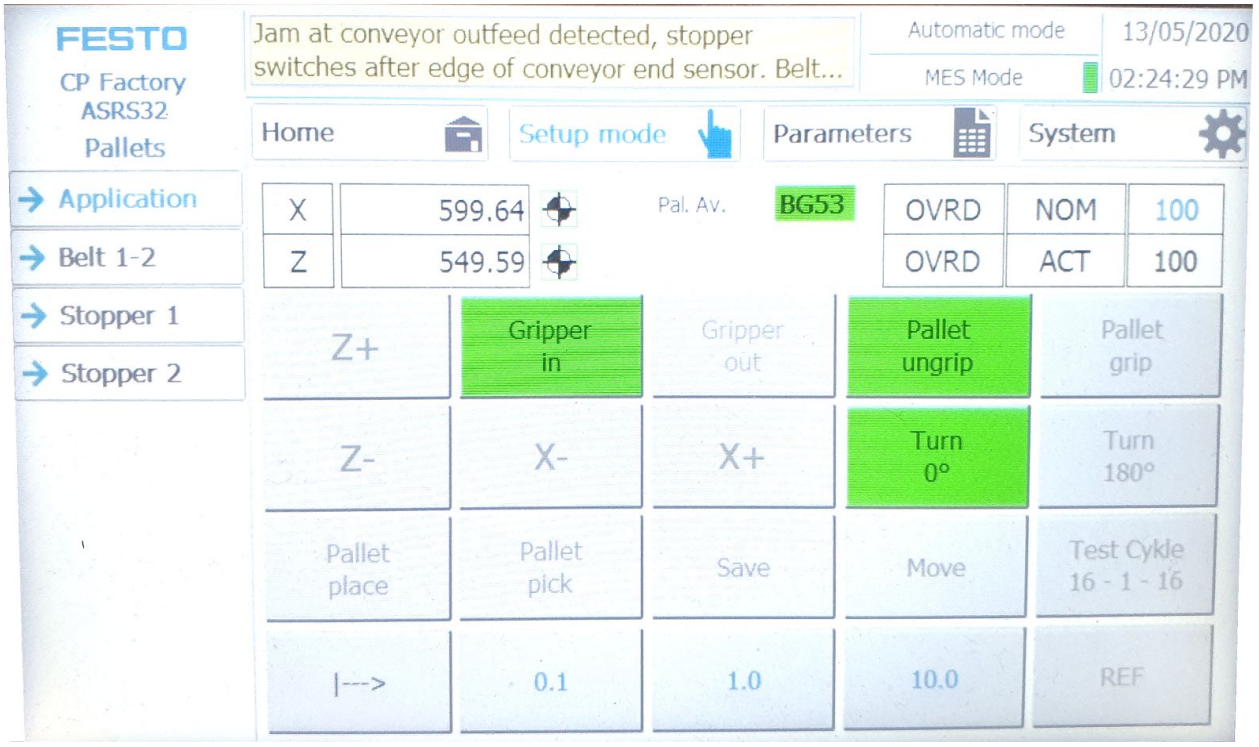

Figure 13: ASRS32 - Settings of the gripper arm

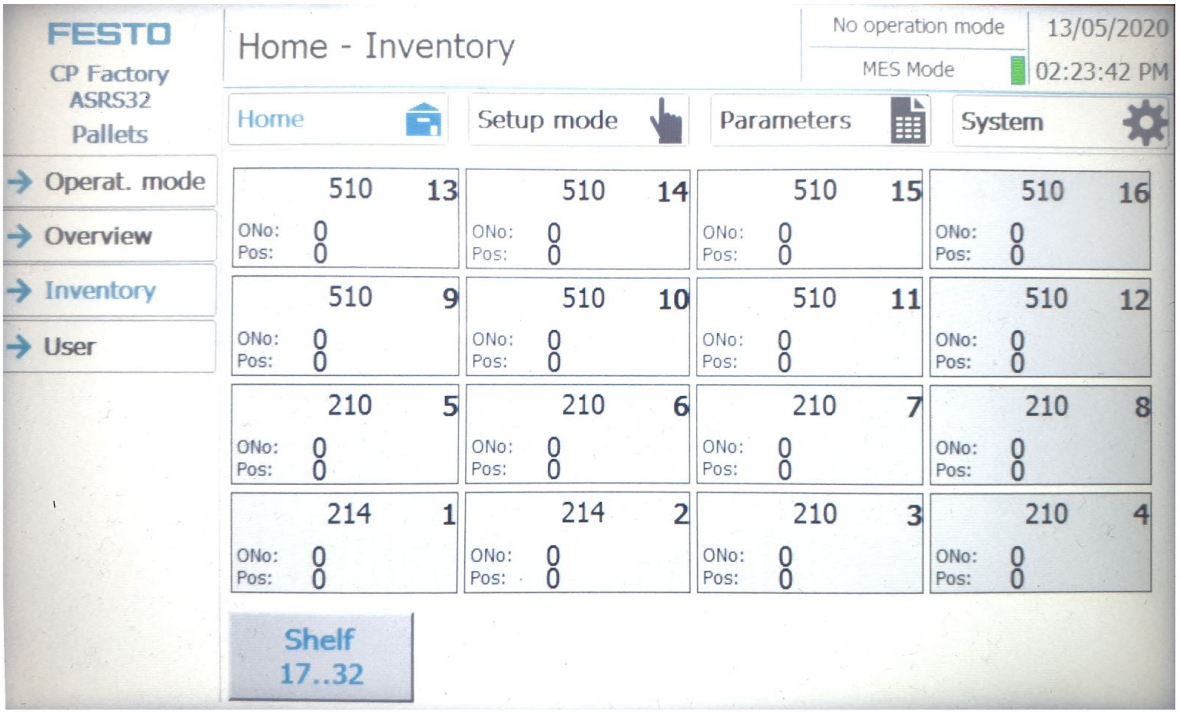

Figure 14, ASRS32 - Storage:
This is the menu that is showing what is stored and where it is stored in the pallet buffer.

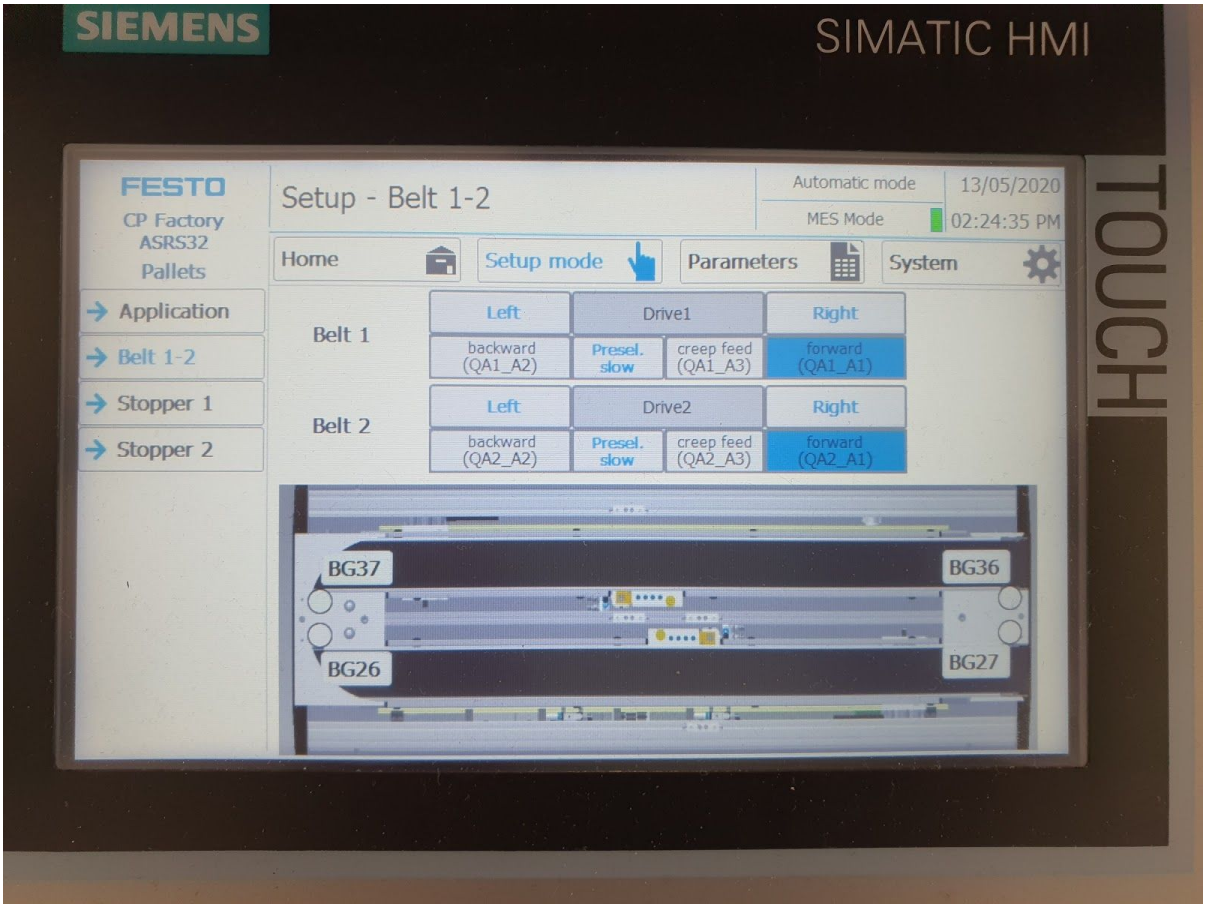

Figure 15, ASRS32 - Belt overview:
Picture shows the complete screen from Siemens unedited to get how the system looks to the naked eye.

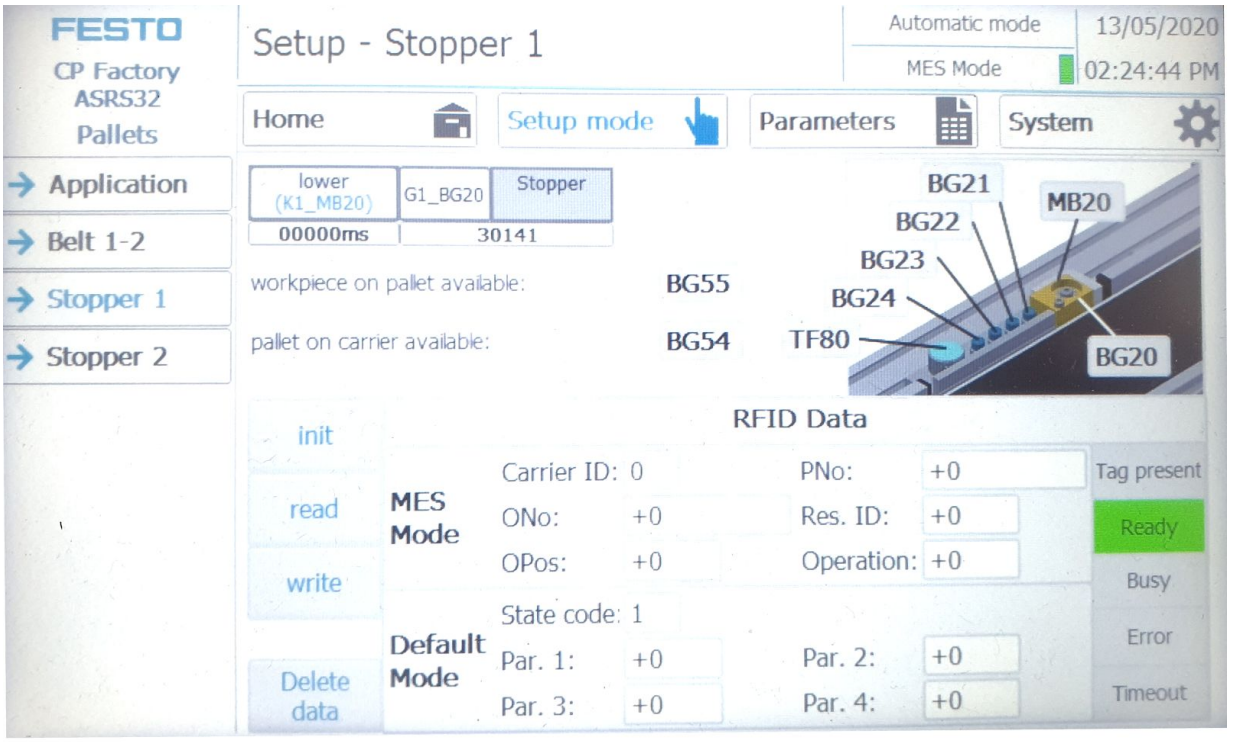

Figure 16, ASRS32 - Stopper 1 and Stopper 2:
Showing corresponding carrier and its information on each station as they arrive

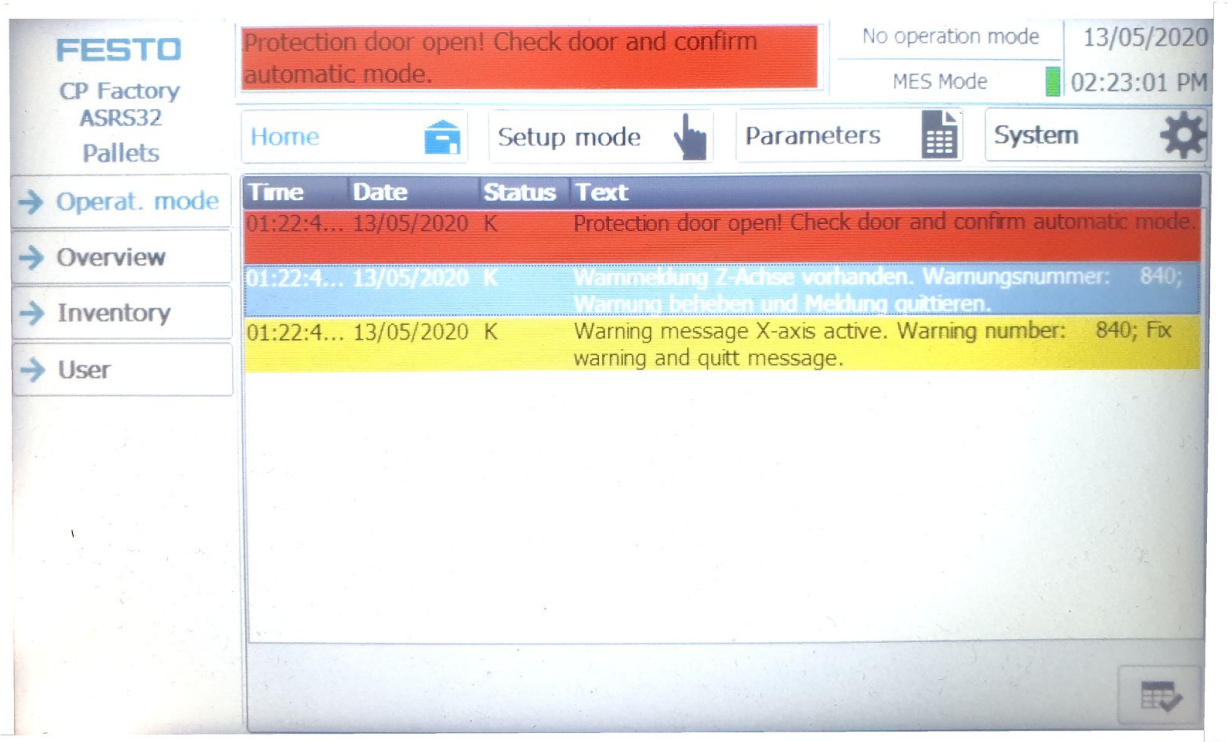

Figure 17, ASRS32 - Error screen.
The most important menu for handling and overviewing errors, this is basically the same text as shown in [Figure 12]. The feedback is lacking, we still don't get orientation and scope of problem.

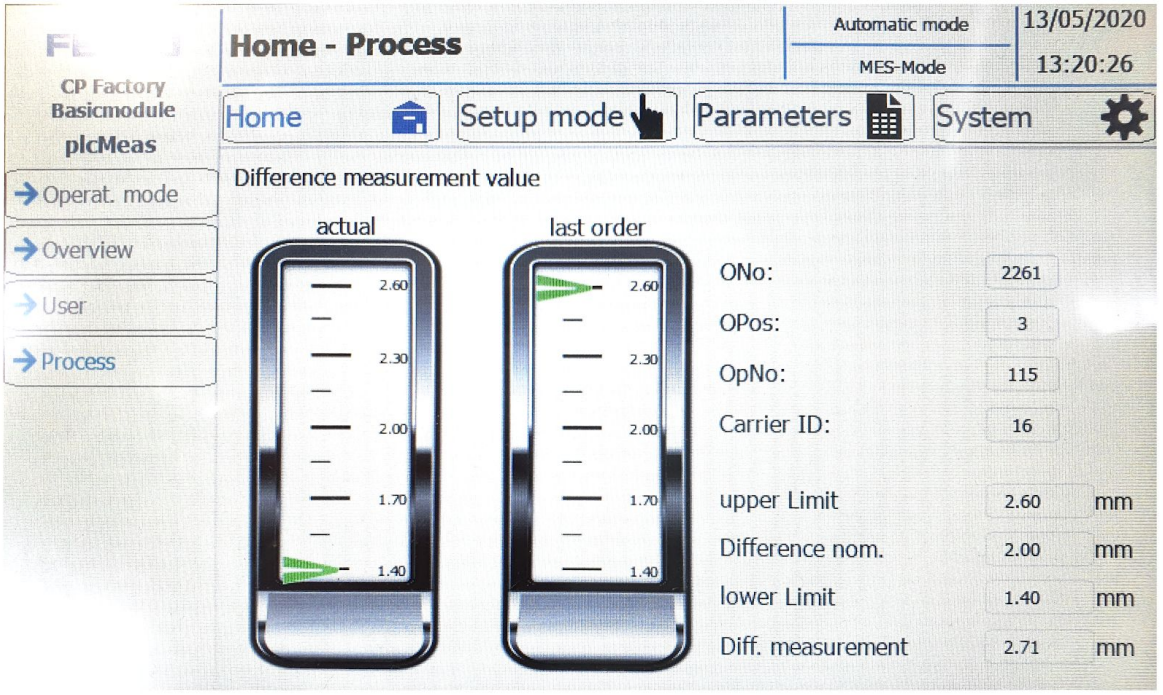

Figure 18, CP-F-Branch - Measure attachment laser-view

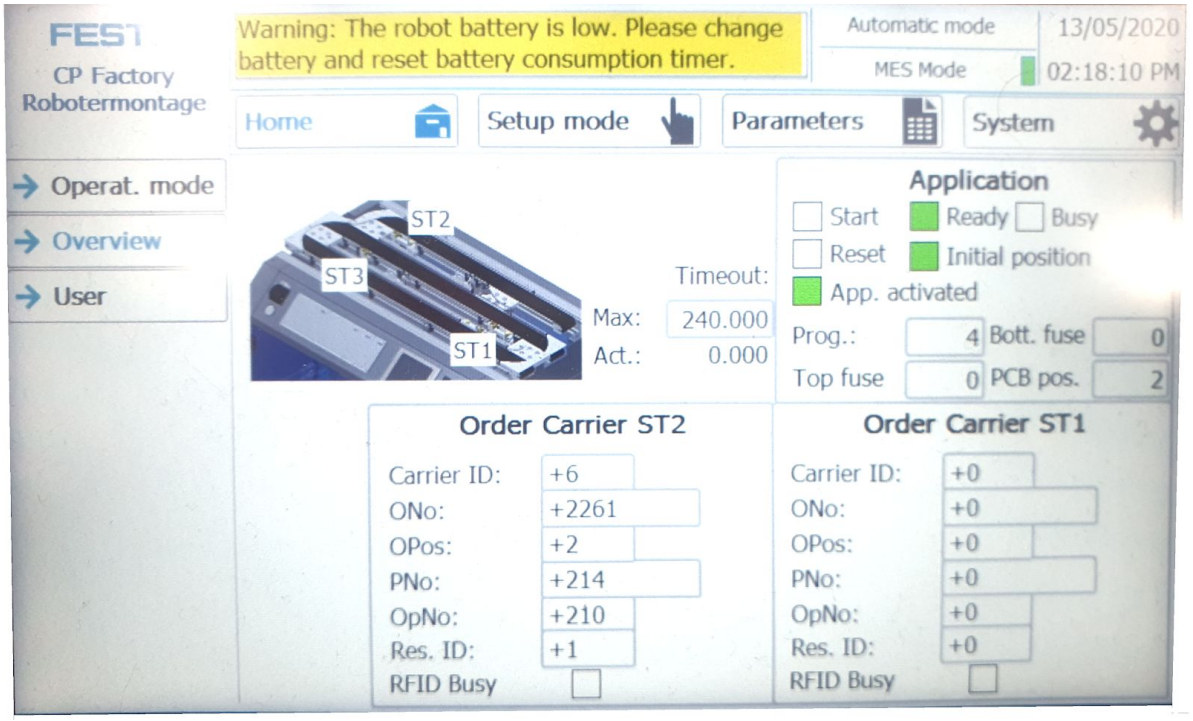

Figure 19, RASS - Main overview.

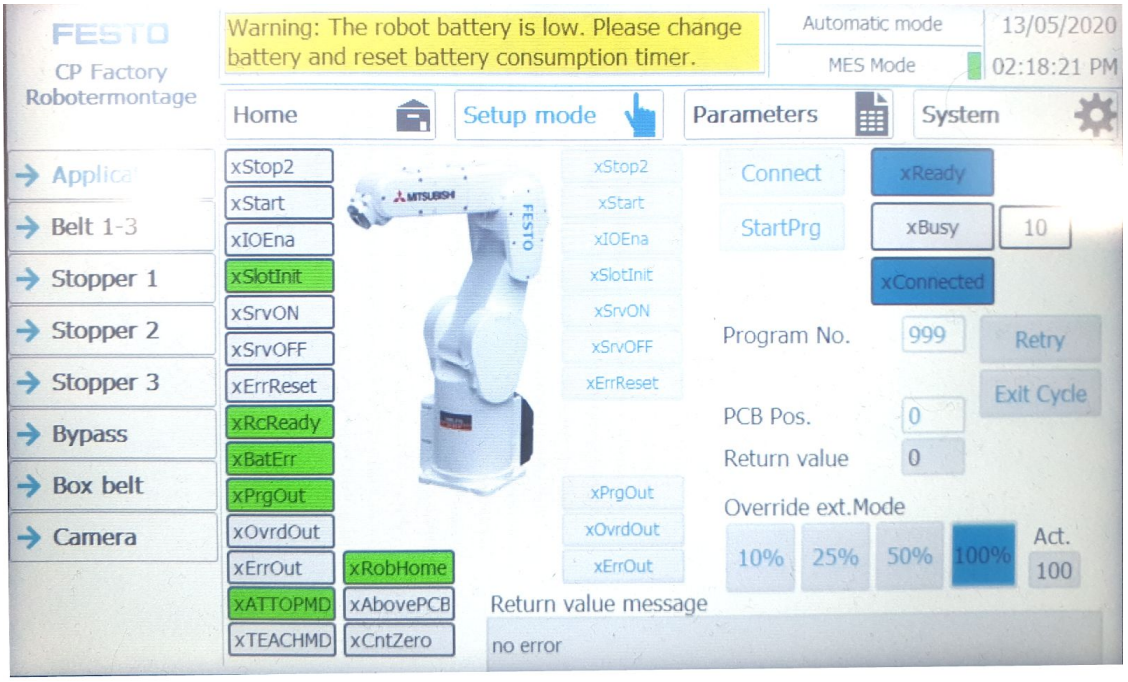

Figure 20, RASS - Robot Arm.

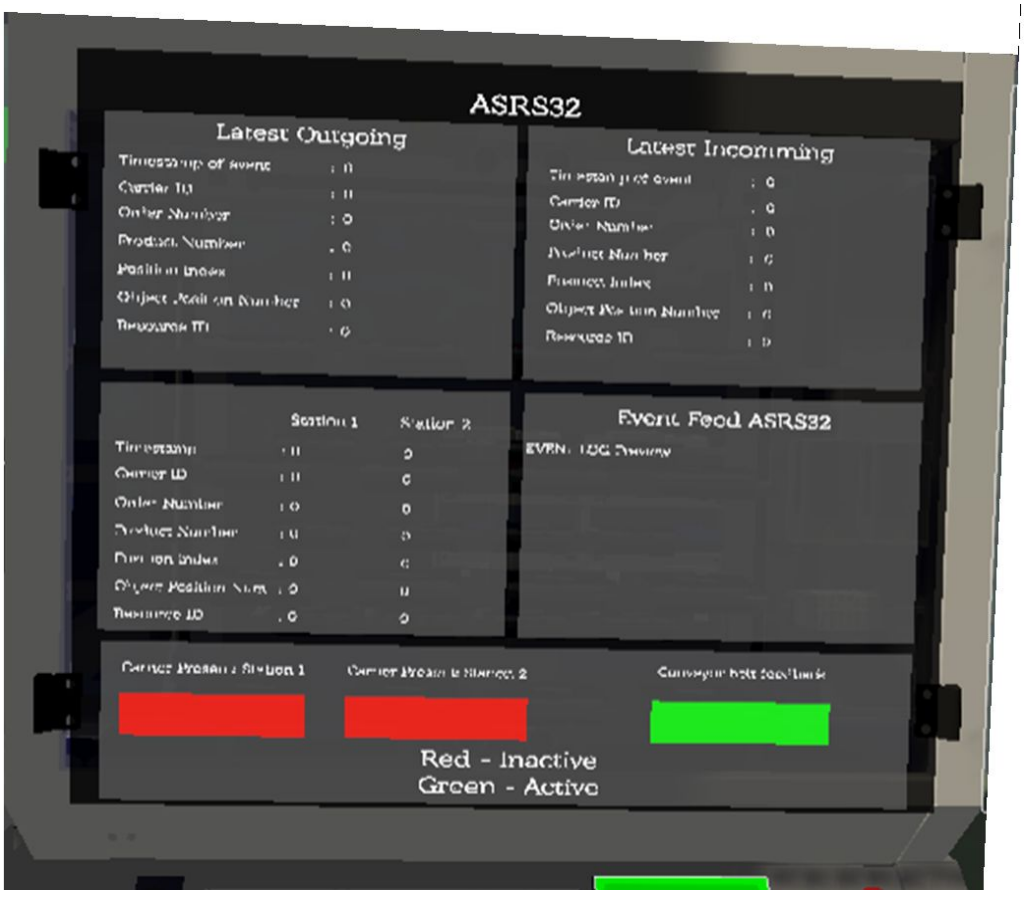

Figure 21, DRM 1- Menu:
First menu iteration in front of the ASRS32 front window

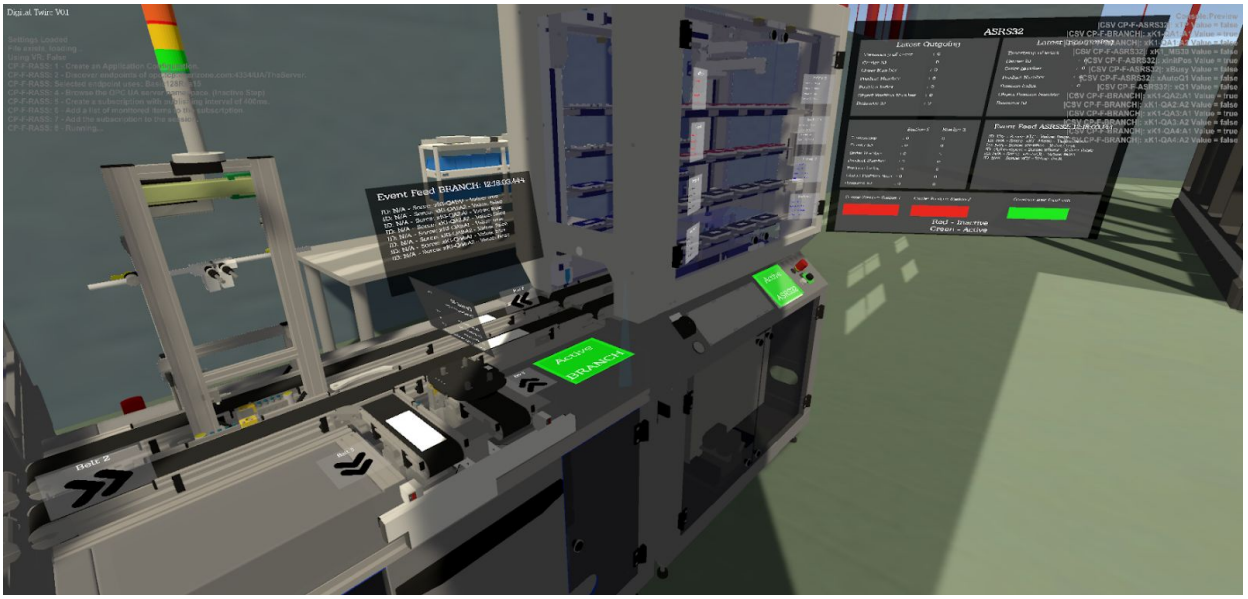

Figure 22, DRM 2 - Implementation showcase

Figure 23, DRM 3 - Error and readability improvements:

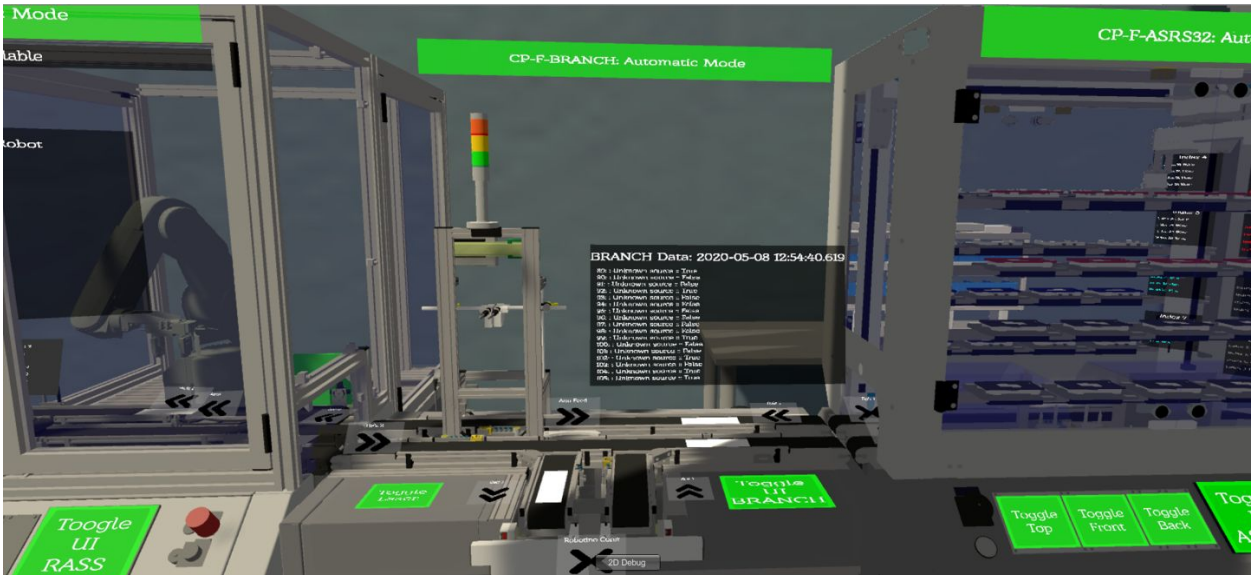

Figure 24, DRM 3 - BRANCH Mixed-Mode:

Figure 25, DRM 3 - BRANCH Overview-Mode:
The user has toggled the Measure (Laser) attachment off compared to [Figure 24], which improved visibility on the unit.

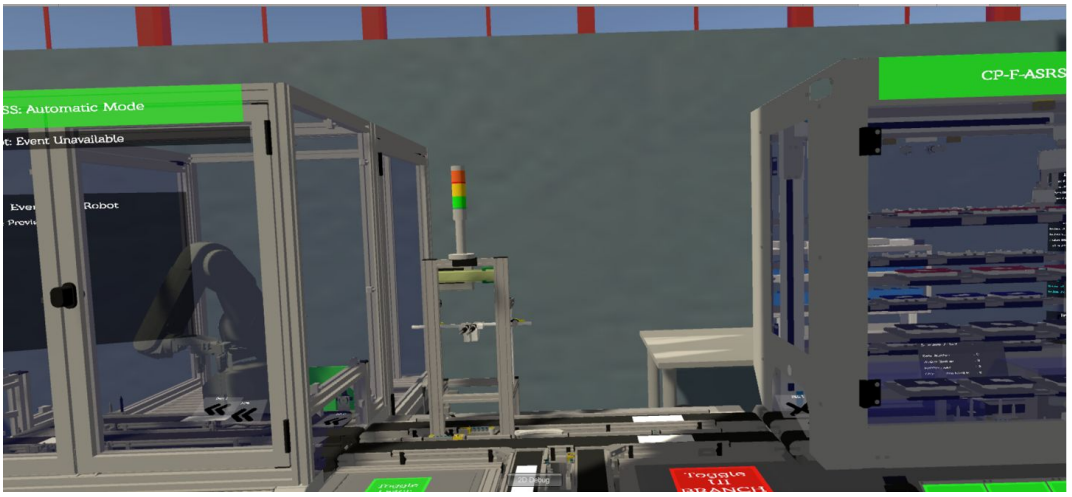

Figure 26, DRM 3 - BRANCH Digital Twin-Mode:
This picture shows the software when the complete UI is turned off which allows for a stricter DT definition (Digital Twin-Mode).

Figure 27, DRM 3 - ASRS32 Mixed-Mode:
This is how the ASRS32 looks like when we enter the software, we can see that UIs exist inside the machine now and that the event feed (far right) is giving a proper name when it receives data. We can also see how updated values on the menu to the right got another colour to stand out in the environment.

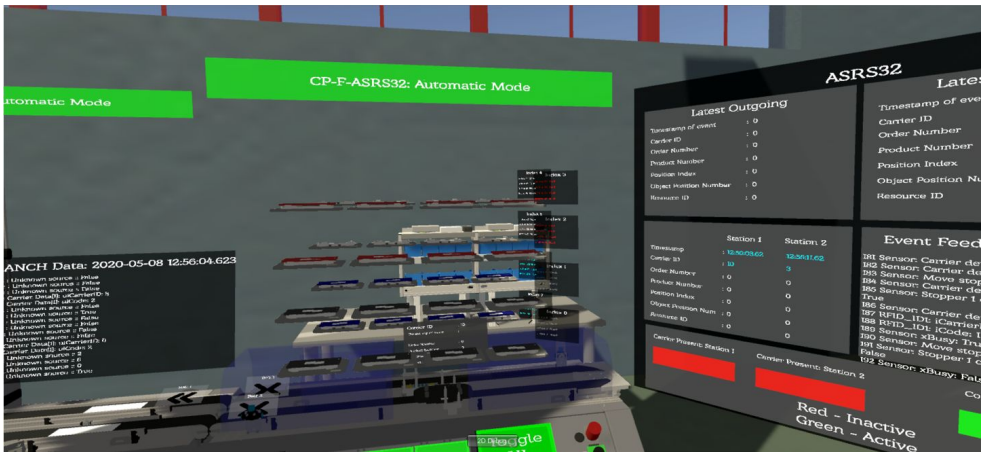

Figure 28, DRM 3 - ASRS-32 Top Abstract:

The security measures of the machine are abstracted away with a button push, we can more clearly see the data inside of the machine compared to [Picture 27]. Also notice the pallets inside of the storage now are present to easily understand what and where resources are stored inside the machine. The UI-elements got moved to the corresponding row as well for improved understandability.

Figure 29, DRM 3 - ASRS-32 Overview-Mode:

We removed all storage positions, the UI-elements are now much easier to read, this is how the overviewing is meant to work when using our software in Overview-Mode.

Figure 30, DRM 3 - ASRS-32 Off:

As we combine Overview- and Digital Twin-mode we can see how easy it is to follow the carrier and its positions in the workflow.

Figure 31, DRM 3 - RASS Mixed-Mode:
We can see a bigger feed to the left and it starts to print out what the machine is doing, not only its raw data. It also has its own feed for robot work, and a small header on top showing current work (none of these have been doing work in the picture)

Figure 32, DRM 3 - RASS Overview-Mode:
Here the user has toggled the top cage off, which improved visibility.

Figure 33, DRM 3 - RASS Off:
Here the user turned off the UI and the top cage, as [Figure 30].

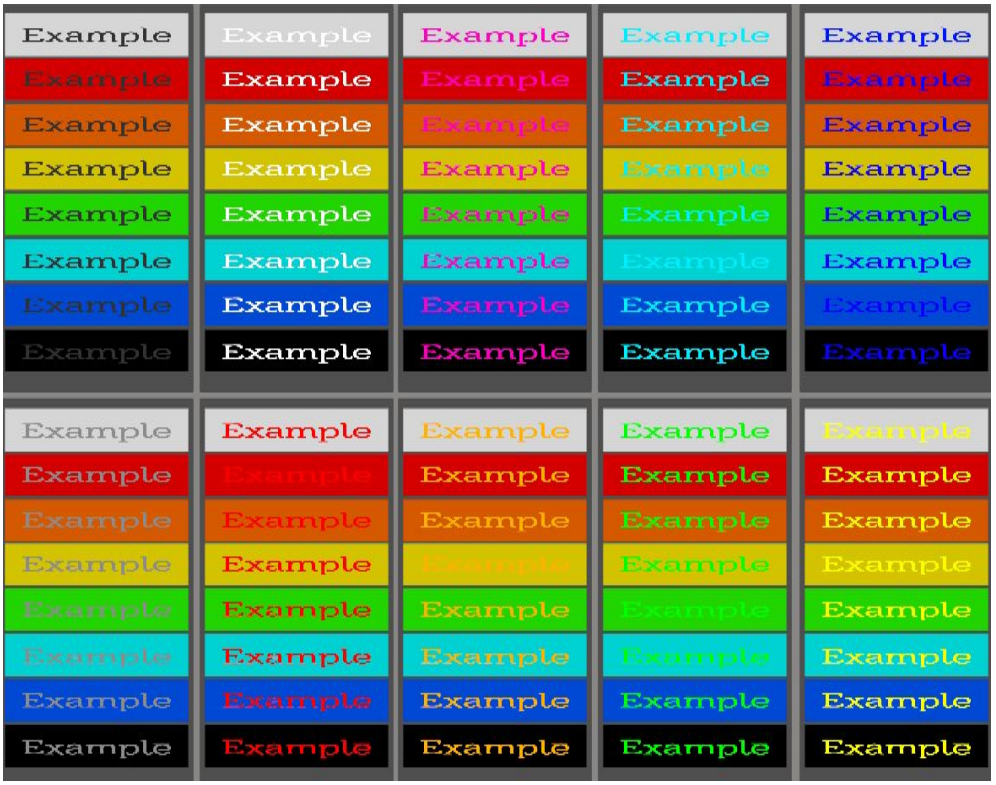

Figure 34, DRM 4 - Color-schemes:

In the third iteration we noticed that it was hard to see some elements in the UI, especially from further away. Thus we implemented this UI to see what colour schemes worked best in the real solution and made comparisons to select the best ones.

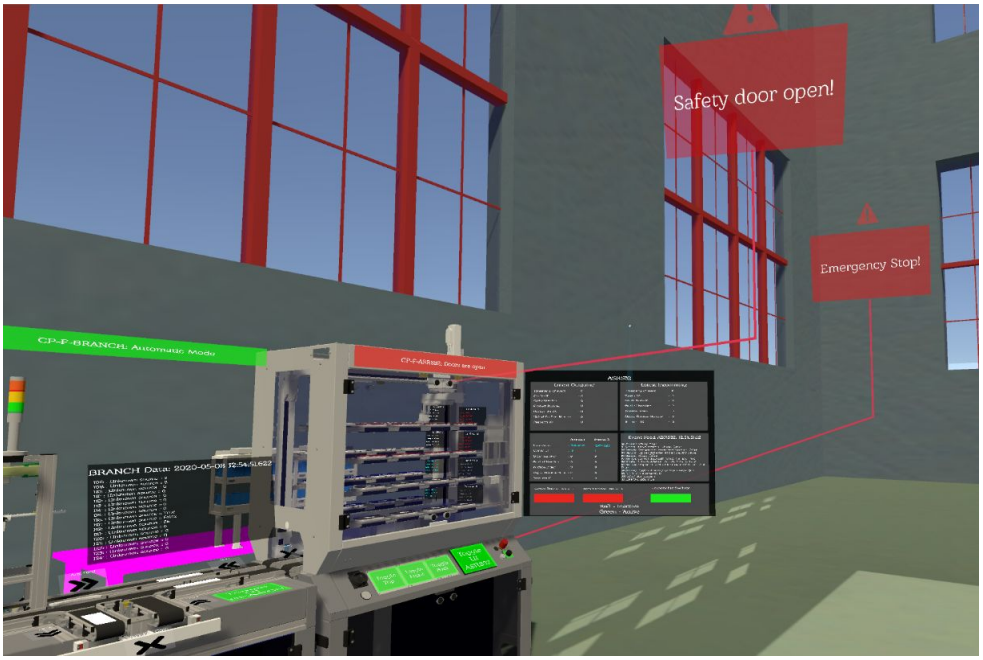

Figure 35, DRM 4 - Error Overview:

We wanted to focus on the error handling, because of this the solution now displayed warnings with lines attached to the point of origin of the fault, this improved understandability and readability, as it not only makes it even clearer which module has a problem but also where on that particular module.

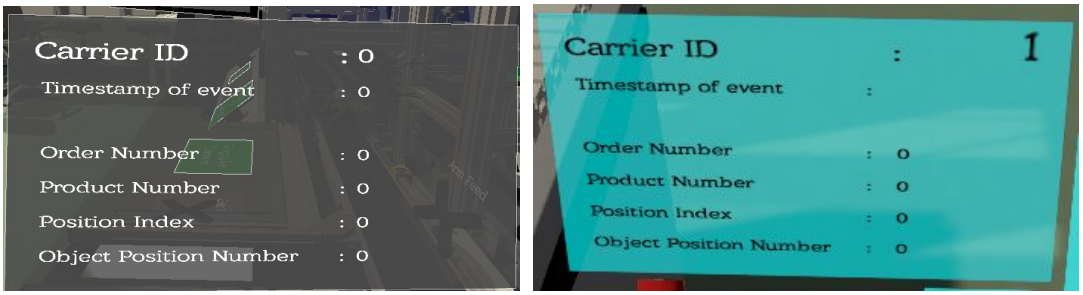

Figure 36, DRM 4 - Carrier Card Comparison:
We can see the before (grey) vs the after implementation (cyan), we wanted the carriers, especially their cards, to be more focusable in the environment and easy to spot as they moved positions. Up close the difference seems to be negative with the new colours, but further away we saw an increase in readability due to the fact that we played with pixel-count.

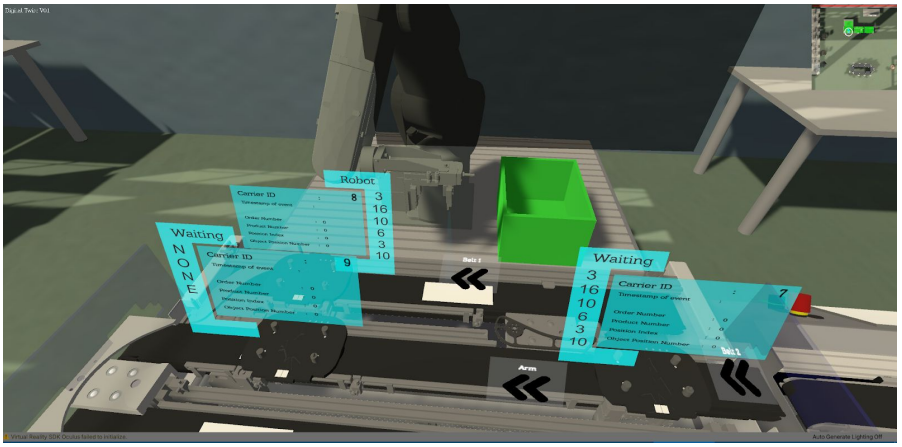

Figure 37, DRM 4 - Stations:
The card-holders for each carrier station on all the modules which allowed historical data.

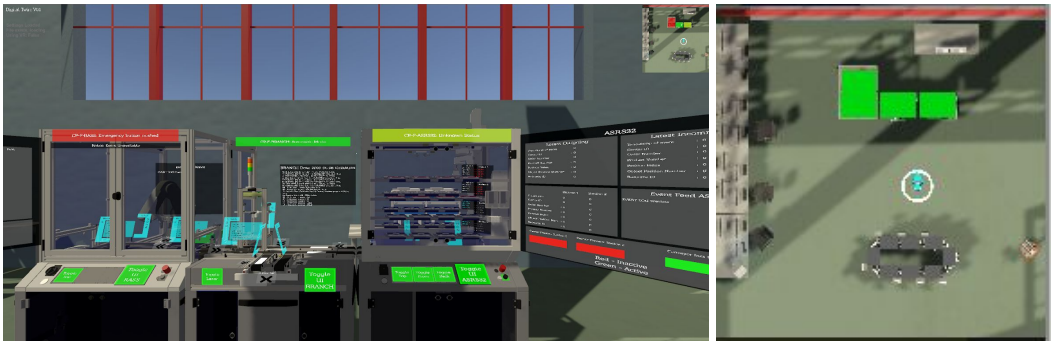

Figure 38, DRM 4 - Minimap overview:
We also chose to implement a small window (top right) that shows the factory from a top-down view and the user orientation, a zoomed in picture can be seen to the right where all units are working. We can also see how each carrier and the stations light up in the environment.

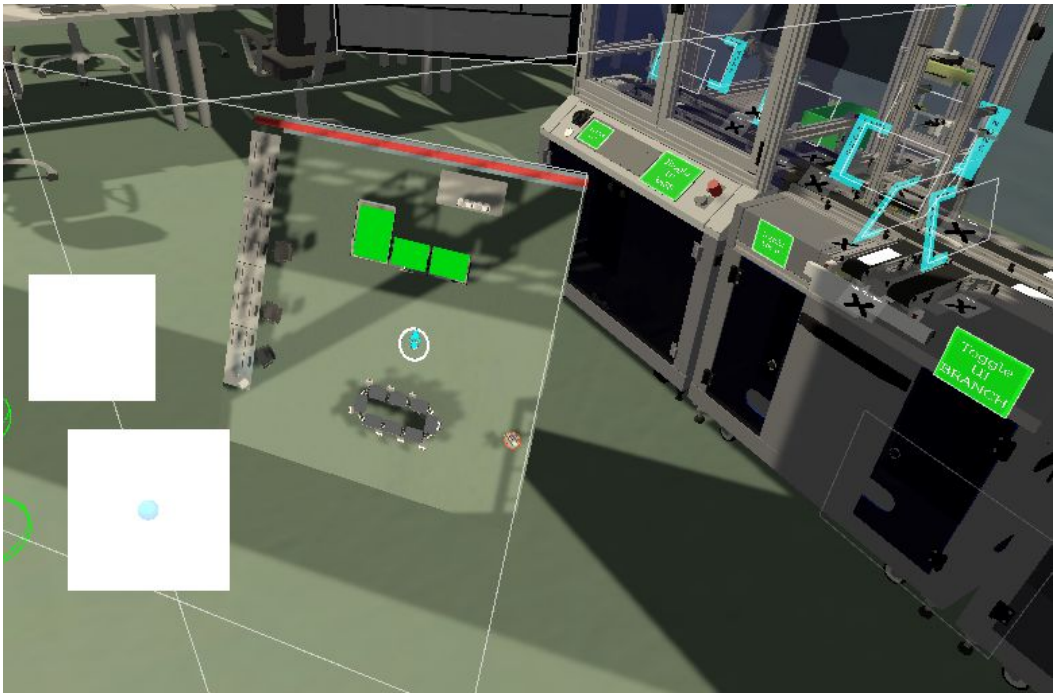

Figure 39, DRM 4 - VR Minimap:

Notice this is from the VR view inside the editor, the VR user has these elements attached to the left hand instead which means the VR user won't have it on the screen unless he/she wants to look at it.

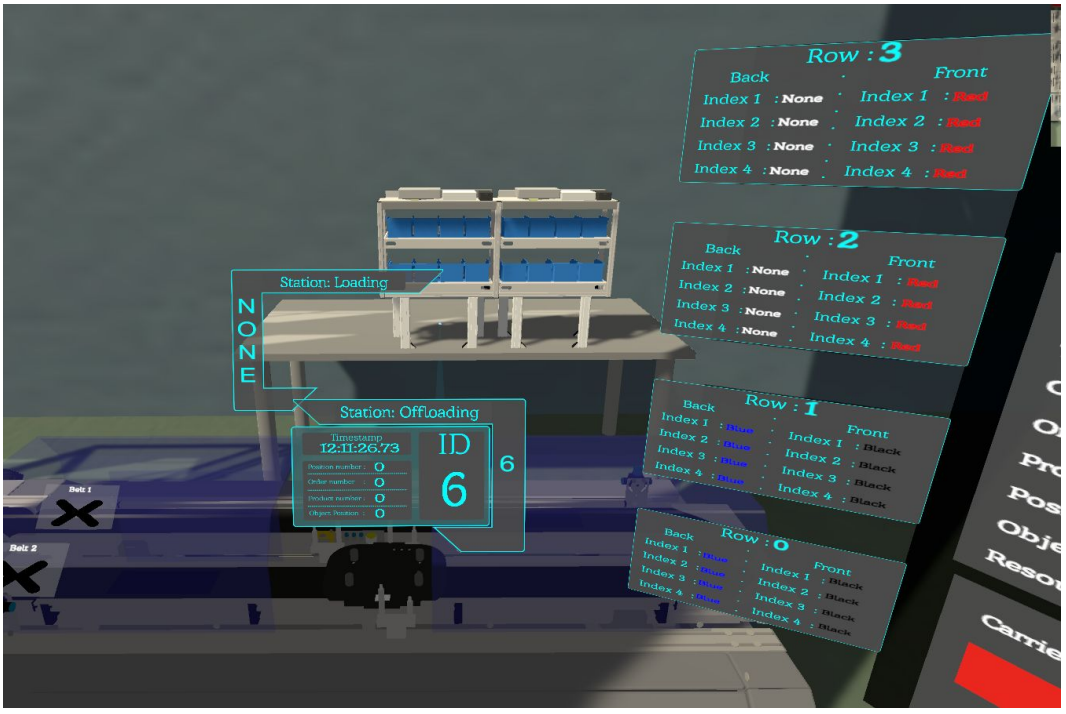

Figure 40, DRM 5 - Station and Carrier updates:

Figure 41, DRM 5 - Thematic implementations:

Figure 42, DRM 5 - Minimap expanded:

We can see the error logic on the newer system, mainly the minimap (top right) now also has smaller sections showing red or yellow smaller sections, these are error specific elements and its position tells the user what error that has flagged (only the common ones are shown in this manner).

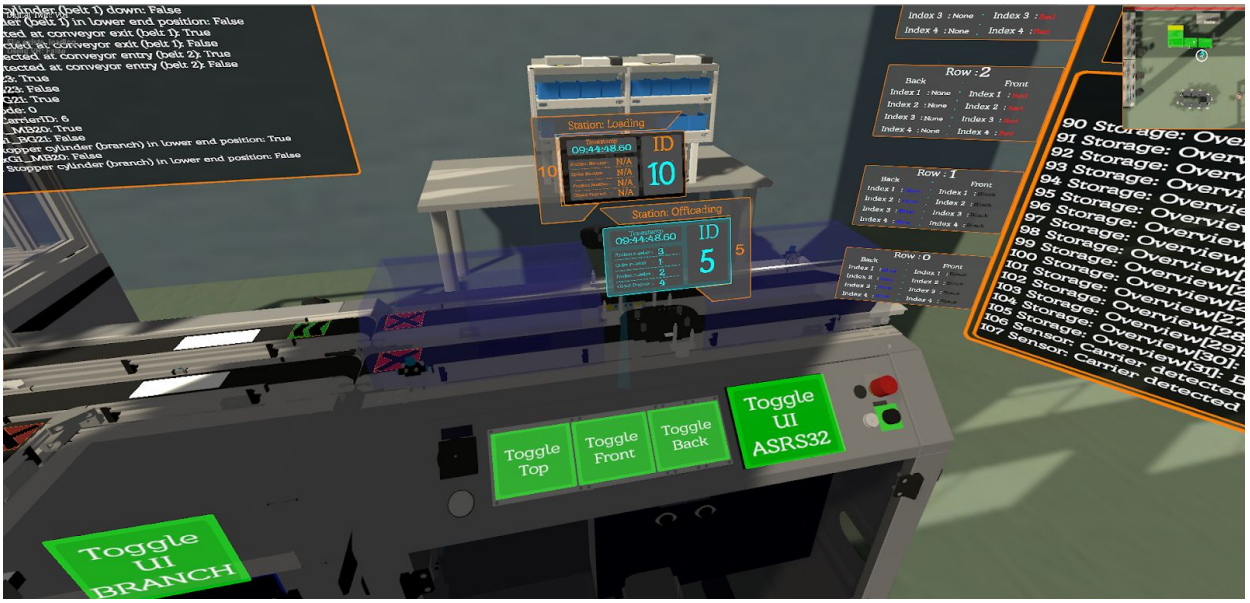

Figure 43, DRM 5 - Colour updates:

We can see the final colors as orange and cyan in combination. The card in front (ID 5) has an active payload on top of it, thus it still has the cyan colors to really stand out and be easy to find in the environment. The card in the back (10) does not have a payload, and its values and color-scheme are less prominent.

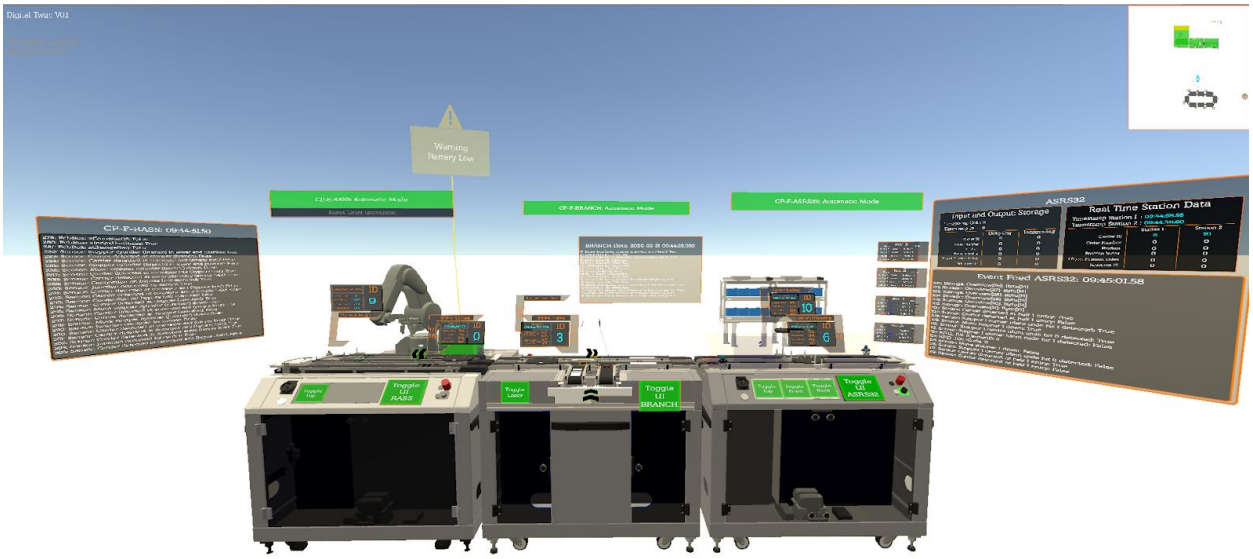

Figure 44, Final - Factory in Overview-Mode

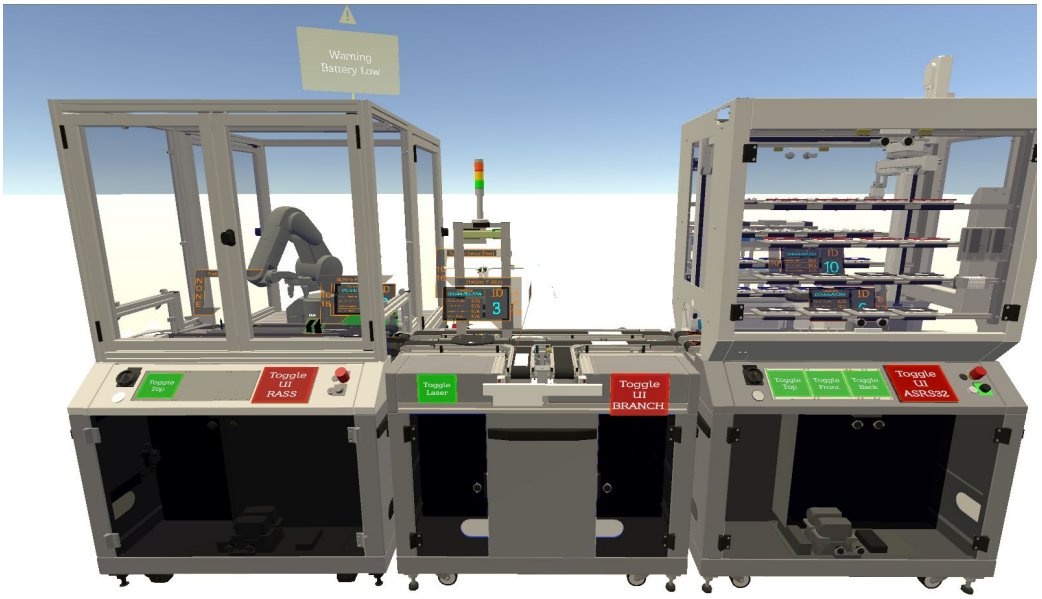

Figure 45, Final - Factory in Digital Twin-Mode

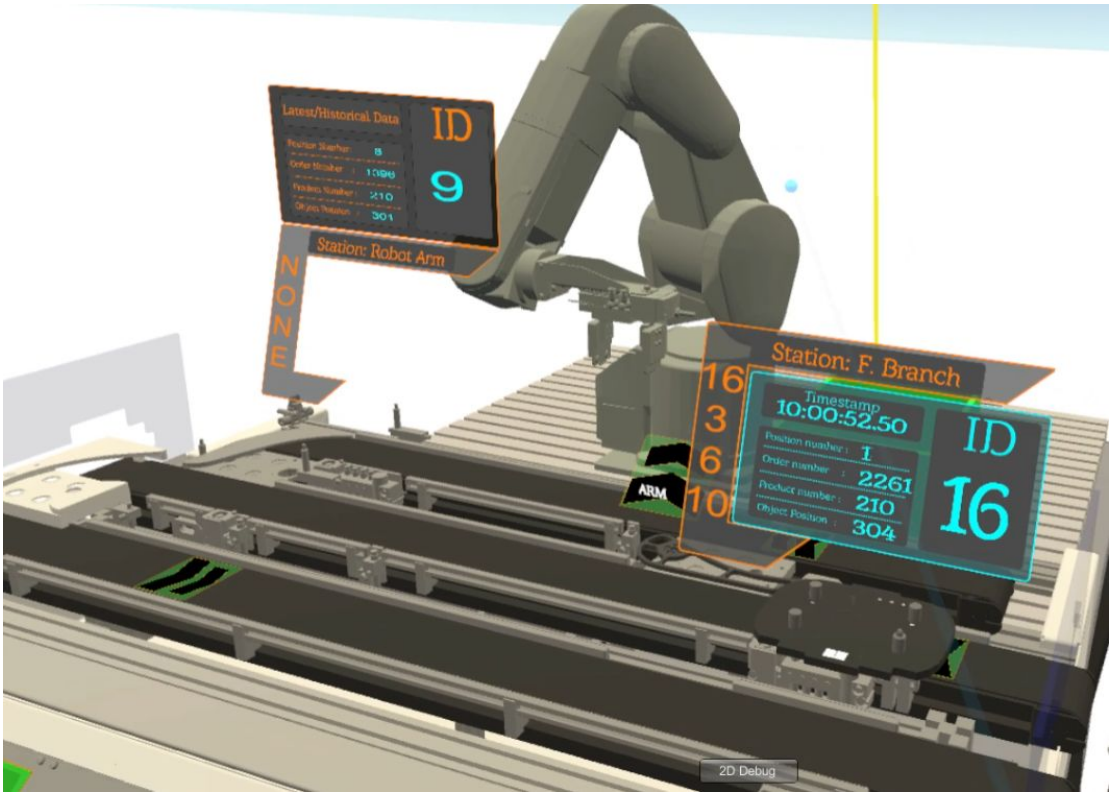

Figure 46, Final - RASS Order overview

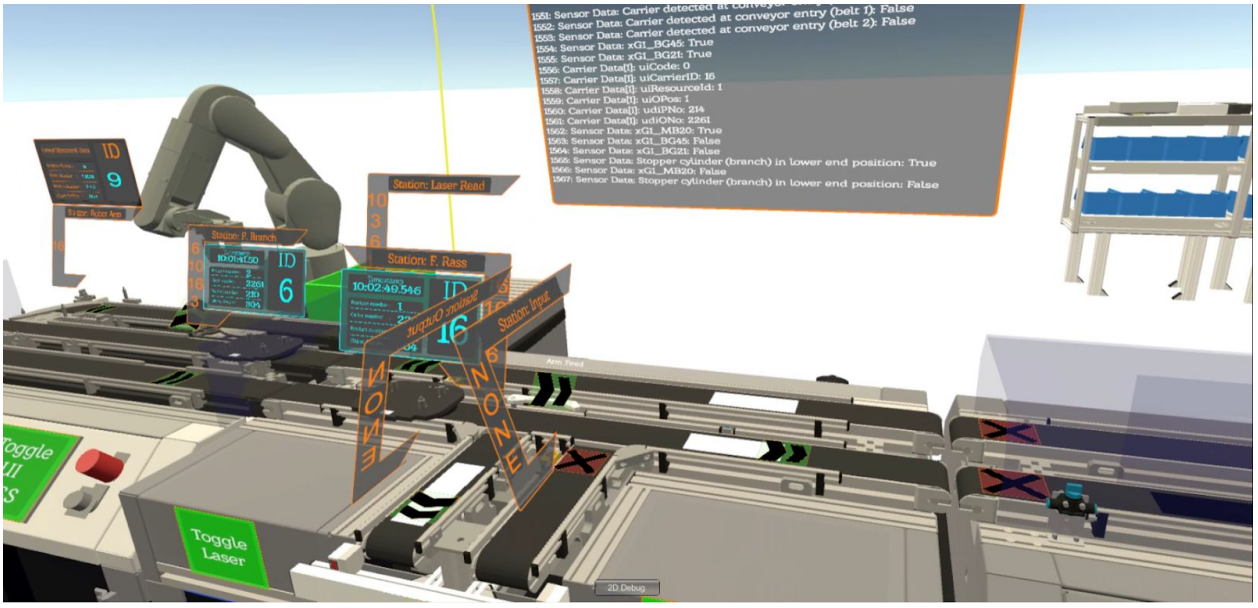

Figure 47, Final - Belt movements

Figure 48, Final - Feed example:

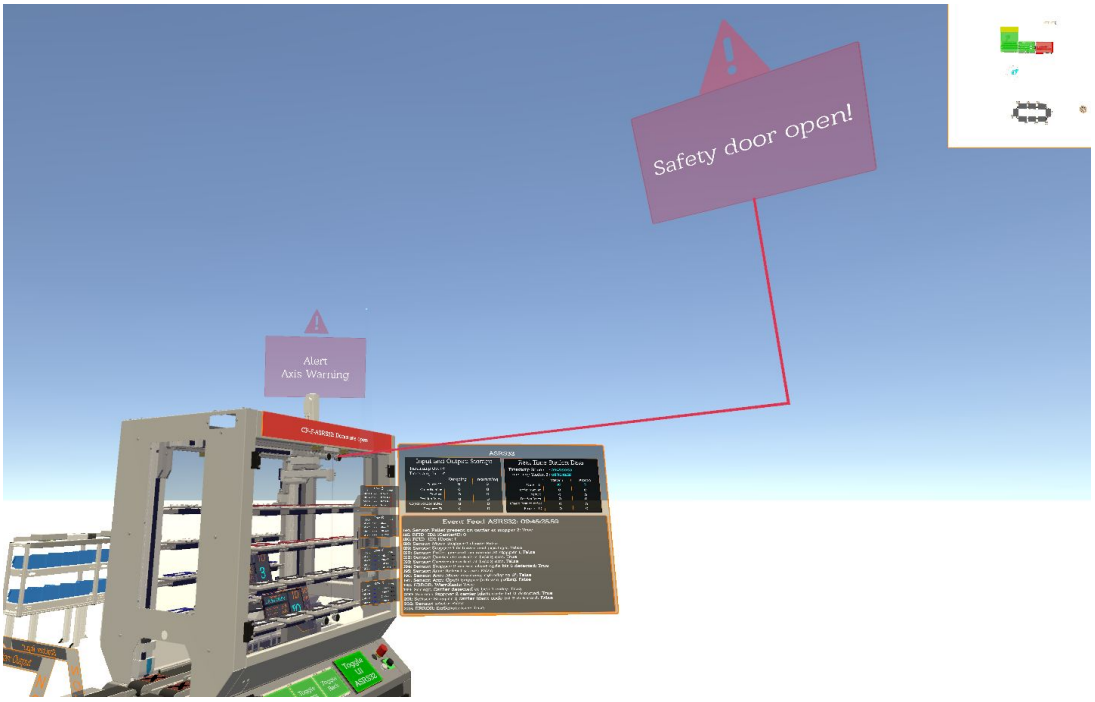

Figure 49, Final - Safety door open:

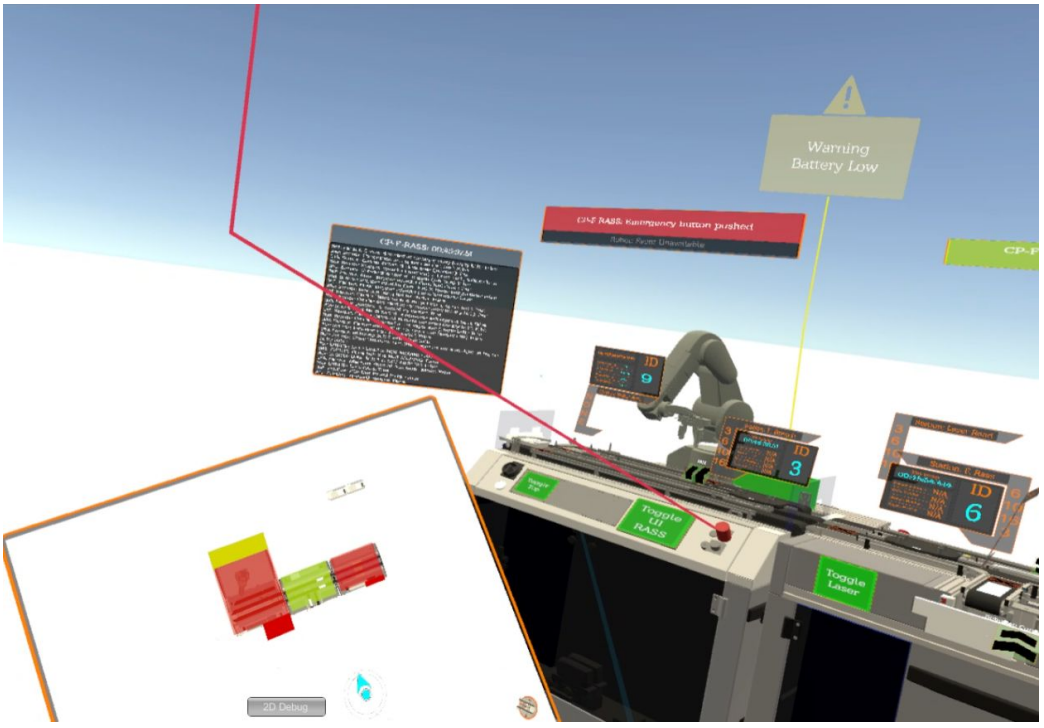

Figure 50, Final - Minimap function in VR:

Figure 51, Final - Environment interactions:
The controllers and VR functionality used in our solution are powered by the Steam VR engine.

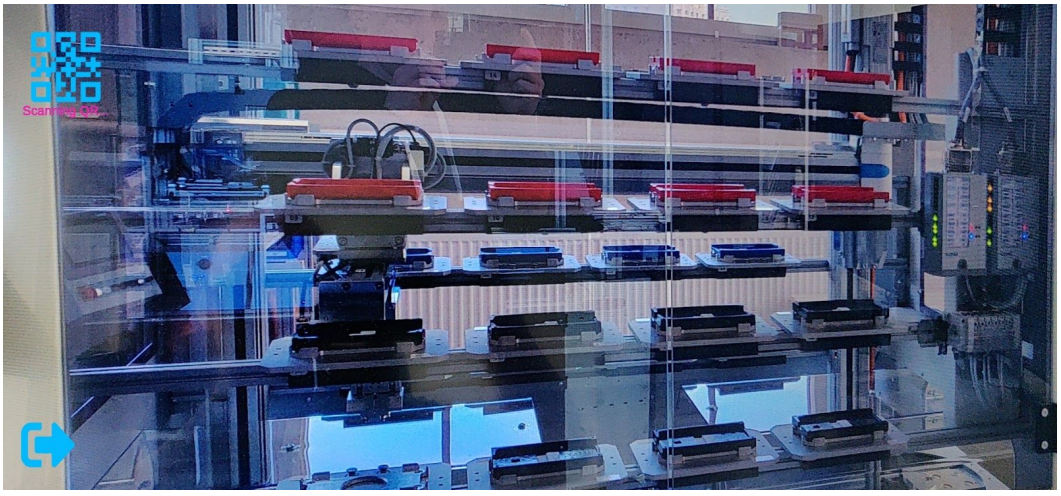

Figure 52, AR - Android Connecting to OPC

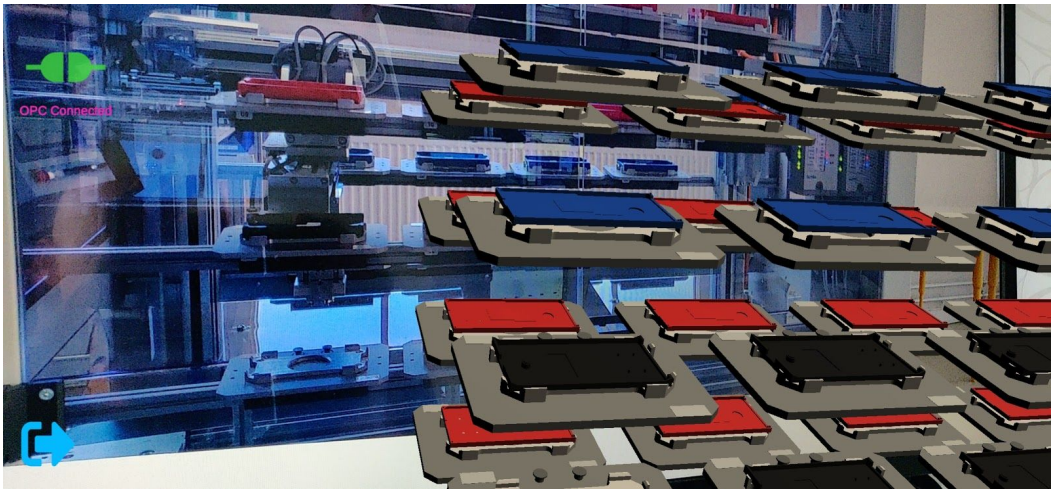

Figure 53, AR - Android Connected to OPC

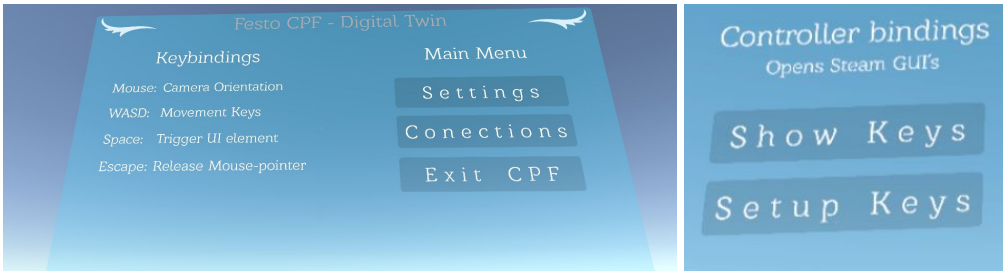

Figure 54, AR - In-simulation menu:

This is the default menu as it looks in the environment, the user just has to click on the two buttons on the right picture to bring up explanations or change how the controllers work. The left picture shows keyboard bindings and depending on mode (VR/Desktop) these menus change for different functionalities.

Figure 55, HoloLens

Tech example of HoloLens from microsoft [13] and an example of how the technology could be used in reality. The picture is owned by and taken from Microsoft

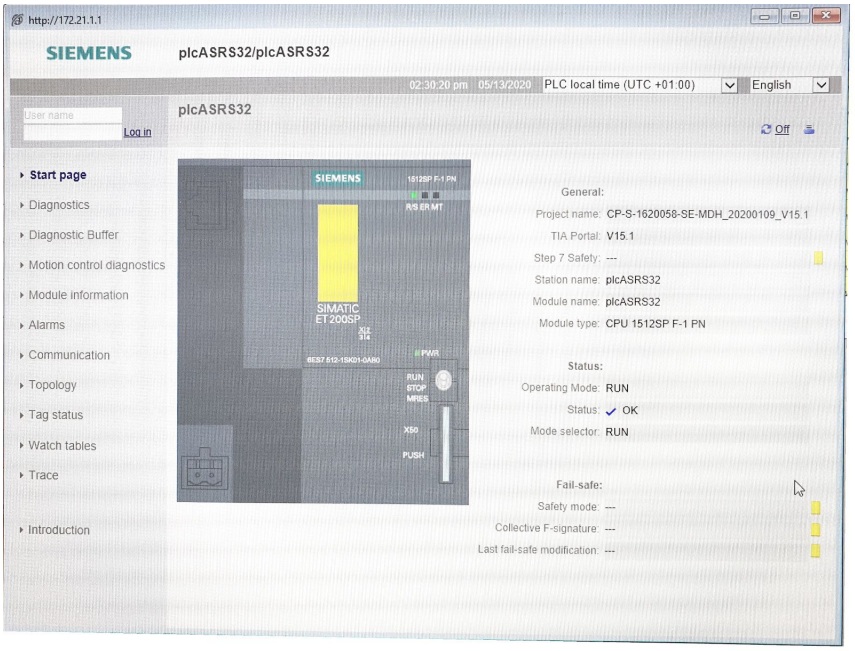

Figure 56, MES4 - ASRS32 PLC overview:

This is the start page of the ASRS32 unit, here can we find very simple overview of its status, but the main screen [Figure 8] does bring more information.

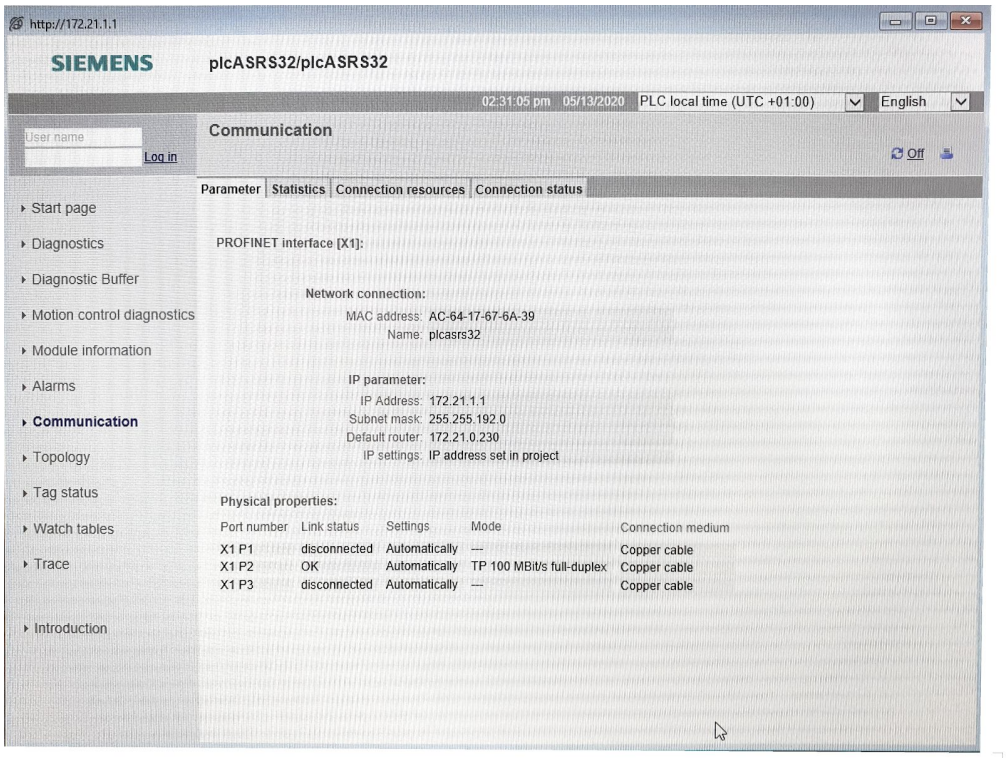

Figure 57, MES4 - Communication page:

Some simple data over the connections made towards the PLC. 100 MBits/s full duplex is well enough to let us receive (and send if we expand functionality) data to our software. Our solution will not bring this data into consideration when it comes to overview (accept that the address will be shown as endpoint when connecting).

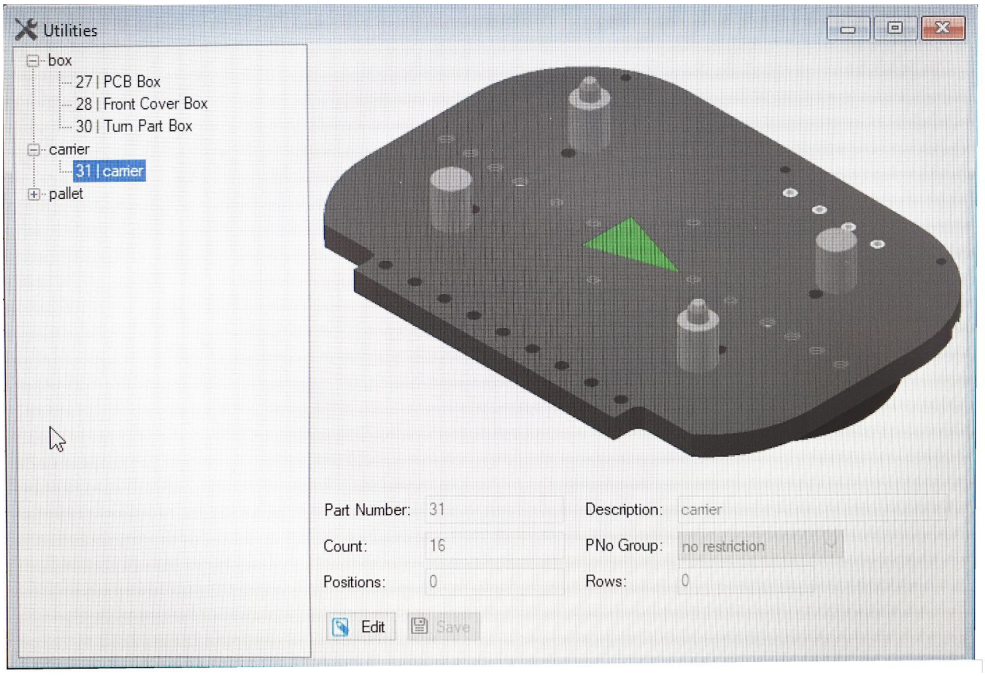

Figure 58, MES4 - The utility page:

From this page can we change values on specific elements when we refill the PCB box for example (The RASS storage for PCB-Boards to mount on top of each order). This picture also shows how each physical carrier looks. ASRS32 or RASS arm will mount the pallet on top of the spiked cylinders.

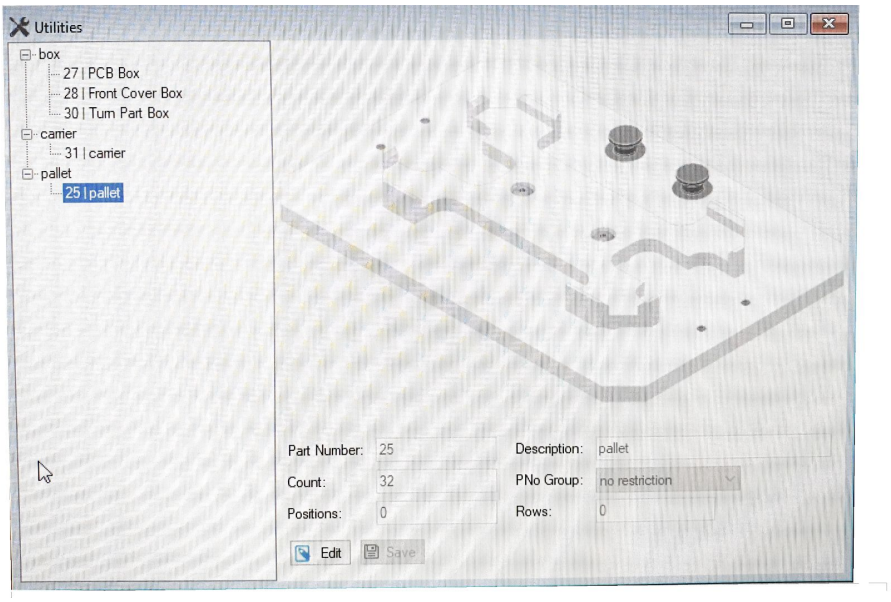

Figure 59, MES4 - The pallets view:
Overview of the pallet that has the payload (mobile) on top of it, also note the ID 25, this is an example of the data that will be presented by the OPC-nodes if we have an empty pallet (such as the ASRS32 storage). The two cylinder buttons (grey) in the back of the picture are where the arms grab the pallets for transportation between stations.

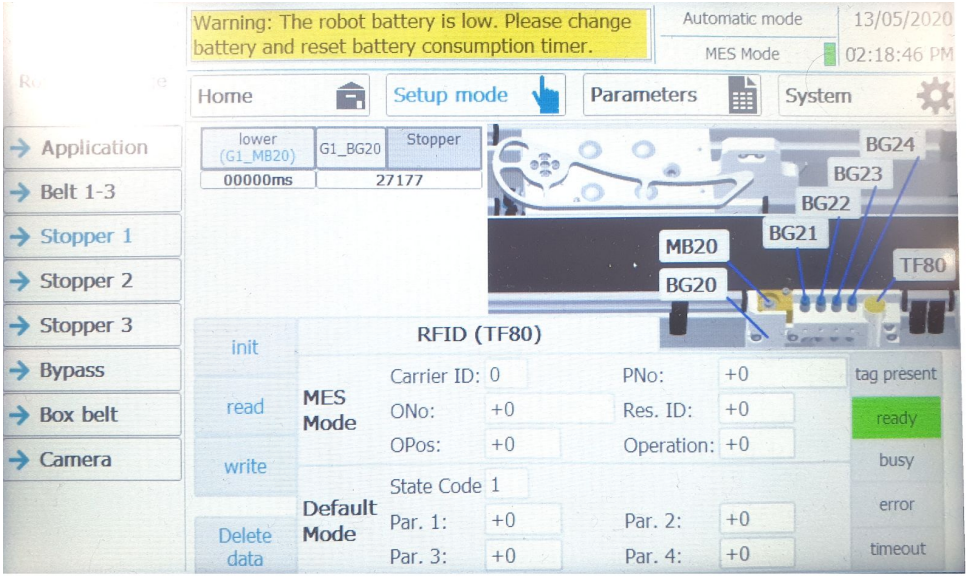

Figure 60, RASS - Stopper 1:
The stopper menus shows the same information as all other Stopper menus, it also shows the deflector, but not if it is redirecting or not.

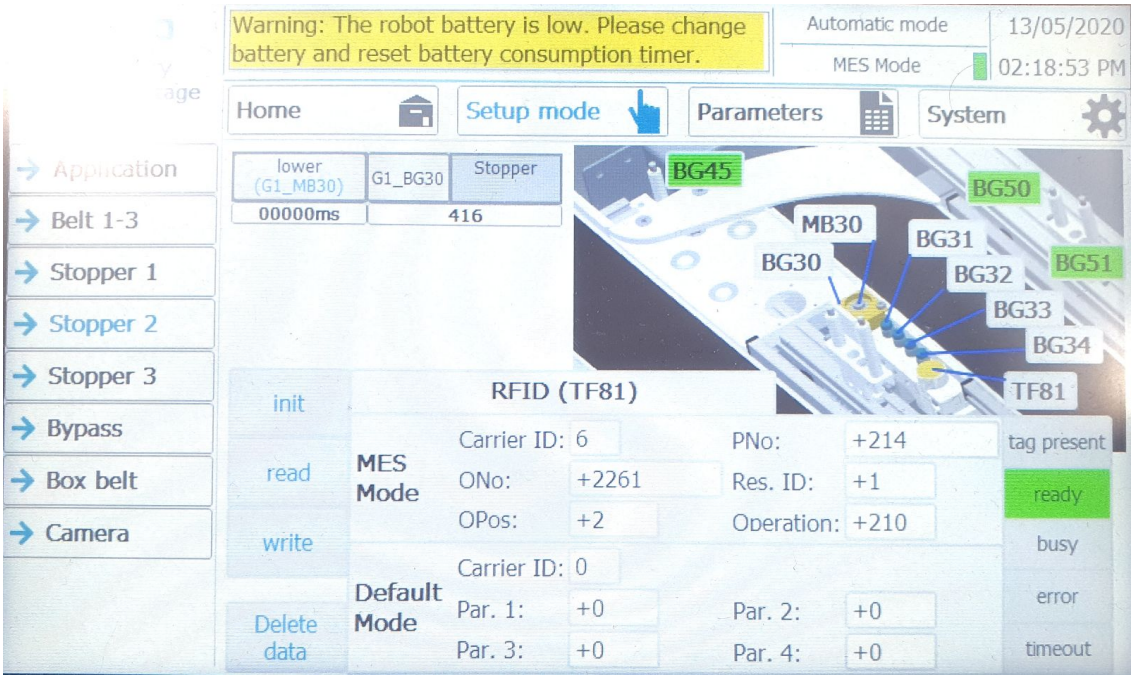

Figure 61, RASS - Stopper 2, Robot station:

Same as other Stoppers, accept that it will show the latest unit as well. We also see sensors without any other information as explanation.

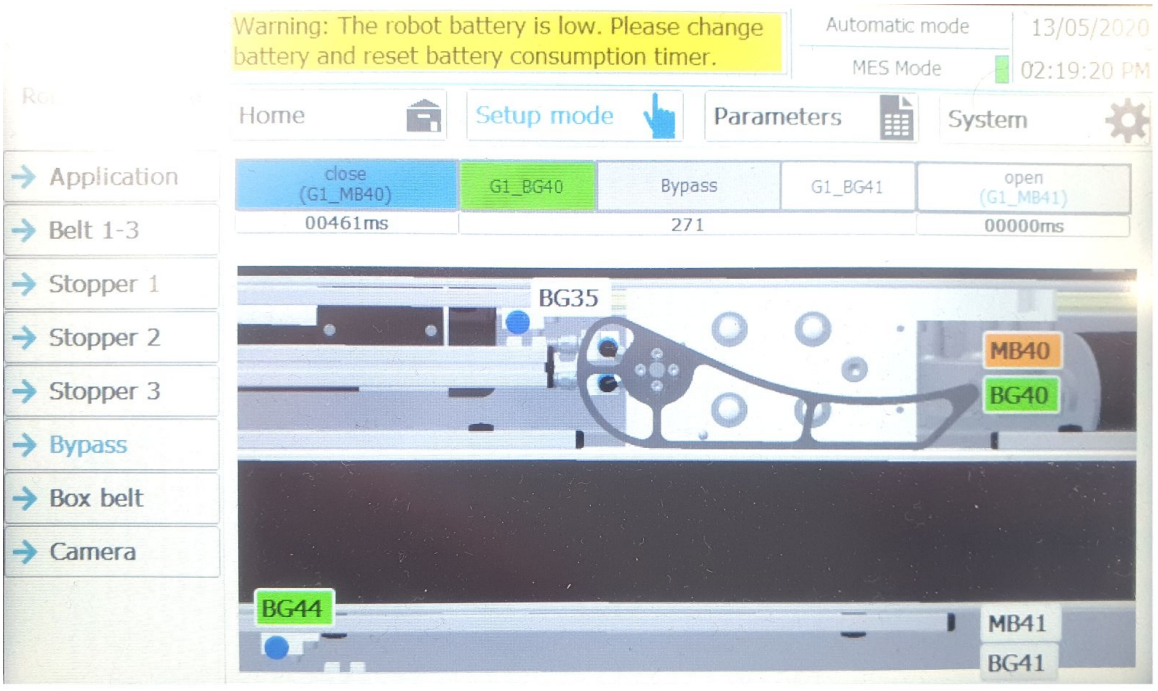

Figure 62, RASS - Divert arm:

We can see the orientation of the divert arm via colored sensors, the problem is there is no clear explanation of what the sensors mean (if BG40 is true (green here) it means that the deflector arm is closed, carrier continues straight on bypass belt, if BG41 is true the opposite is true and carriers is sent to robot arm).

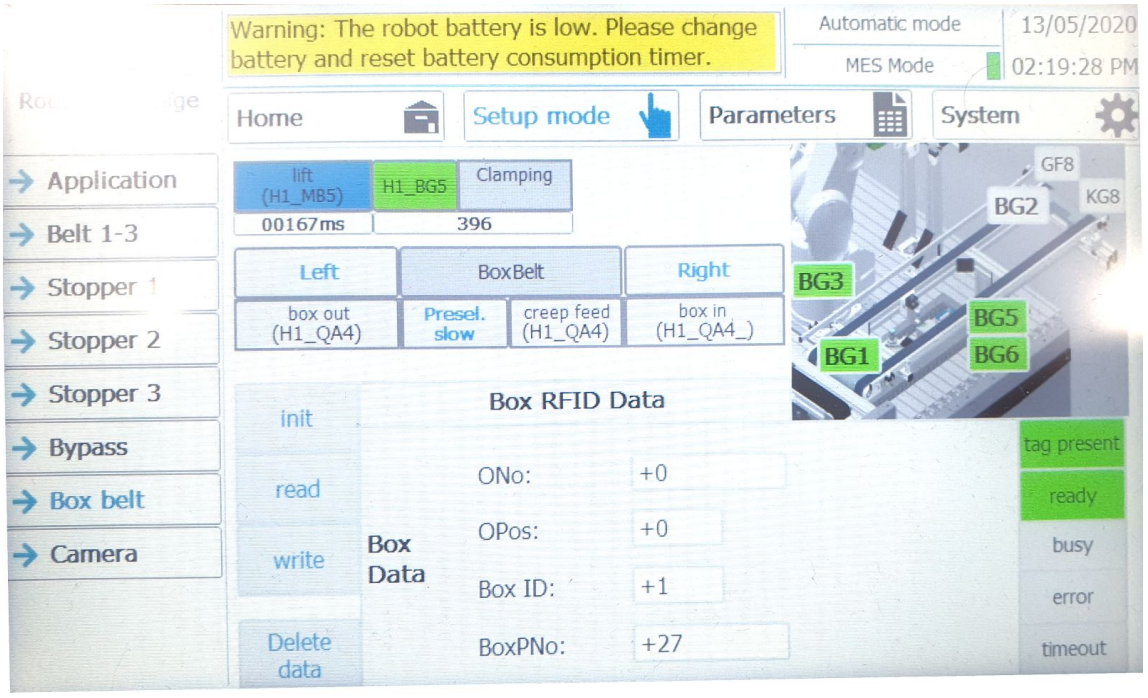

Figure 63, RASS - PCB BOX:

This menu shows the PCB box information, but all sensors are once again not explained. We can also see if the belts used to load/unload the pcb box are active or not.