2020•2021

Faculteit Industriële Ingenieurswetenschappen

master in de industriële wetenschappen: elektronica-ICT

Masterthesis

Automated recuperation and stock refilling in an industry 4.0 training production line

PROMOTOR :

Prof. dr. ir. Eric DEMEESTER

PROMOTOR :

ing. Peter EVENS

COPROMOTOR :

ir. Danny JOOSTEN

BEGELEIDER :

Dhr. Dario Batalha Par CLEMENTE

Eran Kaelen, Charles Vancoppenolle

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

Gezamenlijke opleiding UHasselt en KU Leuven

2020•2021

Faculteit Industriële Ingenieurswetenschappen

master in de industriële wetenschappen: elektronica-ICT

Masterthesis

Automated recuperation and stock refilling in an industry 4.0 training production line

PROMOTOR :

Prof. dr. ir. Eric DEMEESTER

PROMOTOR :

ing. Peter EVENS

COPROMOTOR :

ir. Danny JOOSTEN

BEGELEIDER :

Dhr. Dario Batalha Par CLEMENTE

Eran Kaelen, Charles Vancoppenolle

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

UHASSELT KU LEUVEN

Acknowledgement

We would like to express our gratitude towards everyone who helped us in the realization of this Master's thesis. Firstly, we would like to thank our external promotors Ing. Peter Evens, Ir. Danny Joosten and Dirk Vrancken from the T2-campus for providing us the opportunity to work on this thesis. Their guidance and feedback played an important role in producing this work. Furthermore, we would like to thank our internal promotor Prof. Dr. Ir. Eric Demeester for his suggestions throughout this project. Additionally, we would like to thank Mariano Carreras, Iñaki Fagoaga and Jon Gonzalez from SMC International Training, and Dario Clemente from VDAB, for their technical support regarding the existing hardware and software.

Contents

List of tables

List of figures

List of symbols

AGV: Automated guided vehicle

AMQP: Advanced message queuing protocol

API: Application programming interface

CoAP: Constrained application protocol

COBOT: Collaborative robot

DDS: Data distribution service

FIFO: First in, first out

GIF: Graphic interchange format

HMI: Human-machine interface

HTTP: Hypertext transfer protocol

ICN: Information-centric networking

ID: Identifier

IDE: Integrated development environment

IoT: Internet of things

JSON: JavaScript object notation

LAN: Local area network

LED: Light-emitting diode

LIDAR: Light detection and ranging or laser imaging detection and ranging

M2M: Machine-to-machine

MES: Manufacturing execution system

ML: Machine learning

MPS: Modular production system

MQTT: Message queuing telemetry transport

MTO: Make to order

MTS: Make to stock

MVC: Model-view-controller

NFC: Near-field communication

NUC: Next unit of computing

OPC: Open platform communication

OPCUA: Open platform communication unified architecture

PC: Personal computer

PLC: Programmable logic controller

POI: Point of interest

QOS: Quality of service

QR: Quick response

REST: Representation state transfer

ROS: Robot operating system

SDK: Software development kit

SIF: Smart innovative factory

SLAM: Simultaneous localization and mapping

SMC: Sintered metal corporation

UI: User interface

URL: Uniform resource locator

USB: Universal serial bus

WLAN: Wireless local area network

Abstract

Industry 4.0 is the current automation and data exchange trend in manufacturing. It focuses on automation, efficiency, connectivity and data gathering. The smart innovative factory (SIF) 400 by SMC International Training simulates a smart production line to inspire and teach the key concepts of Industry 4.0. It integrates the order management, production, storage and shipment of goods.

The SIF-400 is a modular system, consisting of 13 stations, from which 12 form a connected production line. The last, physically separated station, is used to disassemble the end products back to the raw materials, in order to achieve reusability. Currently, the recuperation of end products and the refilling of raw materials have to be done manually by an operator, which is time and labour consuming.

This Master's thesis deals with this problem by integrating an automated guided vehicle (AGV) in combination with a mobile application within the SIF-400's environment. The AGV is used to transport end products and raw materials between stations when critical stock levels are reached. Besides, the mobile application is used to notify operators when a transfer between the AGV and a station, or the inverse operation, is required. Furthermore, the resources are controlled by a centralized control service, which manages the communications and takes logical decisions. Hence, this thesis results in the integration of an AGV in an existing Industry 4.0 production system, which currently offers an efficient and expandable transport solution.

Abstract in Dutch

Industrie 4.0 is de huidige trend van automatisatie en data-uitwisseling in de maakindustrie. Hierbij ligt de focus op automatisatie, efficiëntie, connectiviteit en dataverwerving. De *smart innovative factory* (SIF) 400 van SMC International Training simuleert een slimme productielijn om de kernbegrippen van Industrie 4.0 aan te leren.

De SIF-400 is een modulair systeem, opgebouwd uit 13 stations, waarvan 12 een aansluitende productielijn vormen. Het laatste, gescheiden station, wordt gebruikt om de eindproducten terug te demonteren tot de grondstoffen, om zo herbruikbaarheid te bereiken. Momenteel wordt de recuperatie van eindproducten en de aanvulling van grondstoffen manueel uitgevoerd door een operator, hetgeen een tijdrovende en arbeidsintensieve opdracht is.

Deze masterproef behandelt dit probleem door een automatisch geleid voertuig (AGV) in combinatie met een mobiele applicatie te integreren in de SIF-400 omgeving. De AGV wordt gebruikt om de eindproducten en de grondstoffen tussen stations te vervoeren bij kritieke voorraadniveaus. Daarnaast wordt de mobiele applicatie gebruikt om de operator te verwittigen wanneer een overdracht tussen de AGV en een station, of omgekeerd, vereist is. Verder worden de *resources* bestuurd aan de hand van een centrale controle service die de communicatie beheert en logische beslissingen neemt. Bijgevolg resulteert deze thesis in de integratie van een AGV in een bestaande Industrie 4.0 productielijn, waarmee momenteel een efficiënte en uitbreidbare transportoplossing werd bereikt.

1 Introduction

Industry 4.0, referring to the fourth industrial revolution, is the current trend of automation and data exchange in the manufacturing context. Sensors, actuators, devices and operators within an industrial environment are constantly interconnected by means of modern smart communication technologies, including machine-to-machine (M2M) and internet of things (IoT). The exchanged and processed data permit to obtain an increased efficiency of the production process.

Several training systems exist to teach operators to work with these Industry 4.0 principles. The smart innovative factory (SIF) 400 by Sintered Metal Corporation (SMC) International Training is one of those training systems. It simulates a highly automated smart factory in which a trainee can master all aspects of the factory's processes like producing, storing, shipping and recycling products. The system principally consists of 14 modules: 13 stations and one automated guided vehicle (AGV). The first 12 stations form the actual production line, whereas the 13th station, called the recycling station, enables operators to decompose the end product into its constituting components. Hence, it enables to close the loop and as such to obtain a circular production system. A more detailed overview of the architecture and workflow of the SIF-400 is given in chapter *2 Materials*. The following sections discuss the problem statement and the related objectives.

1.1 Problem statement

The T2-campus, located in Genk, Belgium, possesses such a SIF-400 system. As aforementioned, the end products are manually decomposed by means of the 13th station. This task is completed by an operator, which at first carries the end products from the 12th station, the dispatching station, towards the recycling station. Thereafter he or she decomposes the end product into raw materials. Finally, he or she is in charge of refilling the stations with their respective raw material(s). Concerning this, limitations are induced by the current system conception as explained in the following paragraphs.

Firstly, the lack of details from the implemented alarms forces a manual check of the operator. Currently, the alarms, which are station-specific, warn the operator about the need for a specific action, often corresponding to either the recuperation of end products or the refilling of raw materials. However, no further information is provided concerning the actual required intervention at a particular station. Consequently, the operator is compelled to go to the station in question to check the cause of the alarm. Only from then on, he or she is able to prepare the necessary goods. This results in much wasted time, more specifically manhours as well as production time. Manhours waste refers to the time that could have been spent by the operator on more value-added tasks. Regarding this situation, the operator is forced to move back and forth to check manually the content of the alarm, instead of achieving the required action in a single step. The production time waste on the other hand refers to the waiting time of the stations while the operator is moving around. Therefore, the material availability for production will reduce.

Secondly, and in relation to the first limitation, the current integration approach of the alarm in the human-machine interface (HMI) of each station introduces an additional delay. The integrated alarms are displayed as blinking digital light-emitting diodes (LED) on the HMIs of the respective stations. Hence, a certain delay occurs between the moment when the alarm is generated and the moment when the alarm is noticed by the operator. As with the first limitation, a secondary downtime is added.

Alongside the manufacturing system, an AGV was provided by the Spanish constructor, SMC Internation Training. This movable robot was delivered with an integrated web interface, enabling the monitoring and the control of its actions. However, the AGV is, to date, not integrated in the existing production system. The lack of integration of the AGV within the SIF-400 system misses an opportunity for it to act as an inspiring Industry 4.0 technology demonstrator and teaser.

The T2-campus suggests the implementation of the AGV as a transport solution within the existing manufacturing system. The AGV is hereby in charge of the transport of the end products and raw materials between the multiple stations. It thereby offers a concrete solution to the current inefficient material flow. Although the proposed solution reduces the operator's movements, gaining a considerable efficiency, the operator is still necessary regarding the transfer of the objects from the stations to the AGV or inversely.

The focus of this thesis is on the implementation of a separated system into an already established Industry 4.0 manufacturing line. This requires careful consideration of the possible implementation methods as well as the communications protocols used. The latter is all the more significant as the main requirement is described as the prohibition of modifying the existing resources. The following section discusses the objectives which are associated to this problem statement.

1.2 Objectives

The main task consists of integrating the AGV into the existing SIF-400. This task comprises two objectives, namely the automated recuperation of finished goods and the automated stock refilling for raw materials. Both objectives and the main ideas of the method employed to realize these objectives are discussed below.

1.2.1 Objective 1: Automated recuperation of finished goods

The first objective is the automated recuperation of finished goods. Those finished goods are dispatched by the SIF-412 station from one of three possible docks. Each dock is able to contain at most three finished goods. The automated recuperation objective consists in the automatic transfer of the finished goods from the dispatching station to the recycling station.

Figure 1 shows a schematic overview of the SIF-400 production line and highlights the stations where the AGV should navigate between.

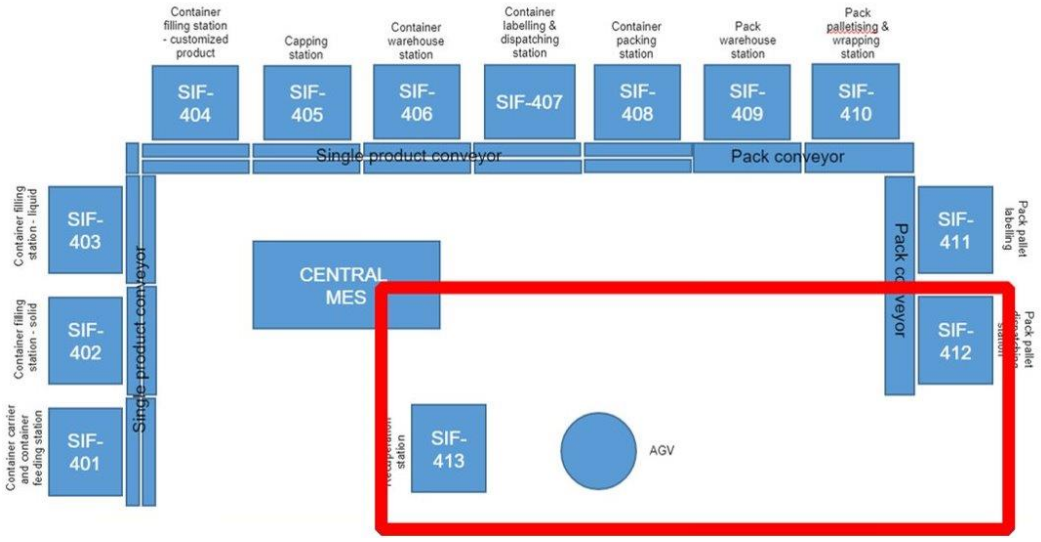

Figure 1 Schematic overview of the SIF-400 with the highlighted automated recuperation objective

To realize this objective, it was divided into different steps:

1. the continuous monitoring of full docks at the SIF-412: the resulting implementation must constantly monitor the stock levels of the dispatching station;
2. the service request to the AGV for the SIF-412: when the maximum capacity of a dock is reached, the automatic movement of the AGV to the SIF-412 station should be initiated by the resulting implementation;
3. the service request to the operator for the SIF-412: once the AGV arrives at the SIF-412, the resulting implementation should trigger the operator's mobile signalling application to request his assistance and move to station SIF-412 that needs his service;
4. the confirmation of completion of the service request by the operator: after all items have been successfully transferred to the AGV, the operator communicates to the resulting implementation that the task is completed;
5. the initiation of the automatic movement of the AGV towards SIF-413: after the transfer of the goods to the AGV, the automatic movement of the AGV towards the recycling station, namely the SIF-413, must be initiated;
6. the service request to an operator for SIF-413: once the AGV arrives at the SIF-413, an operator must also be notified in order to transfer the finished goods from the AGV to the recycling station;
7. the confirmation of the completion of the service request by the operator: finally, once the operator has transferred the finished goods from the AGV to the SIF-413, the AGV can be released for new service requests.

1.2.2 Objective 2: Automated stock refilling of raw materials

After having decomposed the finished goods back to their raw materials at the SIF-413, it is necessary to bring these raw materials back into production. Therefore, these materials will have to be redistributed from the recycling station SIF-413 to the respective production stations. The automation of this material flow forms the second objective.

A schematic overview of the SIF-400, with the involved stations highlighted, is shown in figure 2.

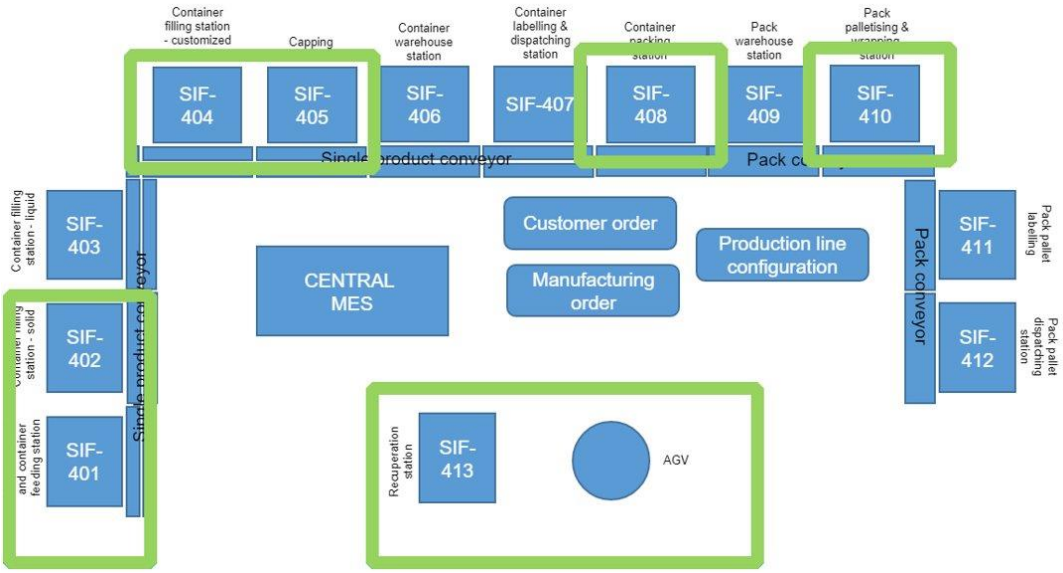

Figure 2 Schematic overview of the SIF-400 with the highlighted automated stock refilling objective

The workflow of this objective is in line with the workflow of the first objective on recuperation. However, the difference consists in the managed stations. Firstly, contrary to the recuperation objective, the critical stock level detection is applied onto six different stations, more specifically the SIF-401, SIF-402, SIF-404, SIF-405, SIF-408 and SIF-410. All those stations require the refilling of one or more specific raw material(s).

Table 1 shows an overview of the stations with their respective raw material(s). Secondly, the recycling station is, regarding this objective, defined as the source station, whereas, for the workflow of the automated recuperation objective, it was defined as the destination station. Although both objectives differ, the workflow is divided in similar steps, whereby the AGV automatically navigates to the appropriate locations and an operator is notified when a service request is required.

Table 1 The stored raw material(s) per SIF-400 station

Station	Inventory
SIF-401	▪ Blue pallets (holding containers) ▪ Square containers ▪ Round containers
SIF-402	▪ Blue pebbles ▪ Red pebbles ▪ Yellow pebbles
SIF-404	▪ Customized goods in a square container ▪ Customized goods in a round container
SIF-405	▪ Square caps ▪ Round caps
SIF-408	▪ Black packs
SIF-410	▪ White pallets (holding packs)

1.2.3 Combined objective

The resulting implementation needs to combine both the automatic recuperation of finished goods and the automatic refilling of raw materials. Regarding this combined implementation, the focus is on priority management, in order to avoid an empty stock in any of the seven stations involved, which would cause a reduced efficiency of the production process.

In addition, the resulting implementation must monitor the battery level of the AGV. An automatic docking process is necessary when a critical battery level is reached. Hereby, it is also required to take into account the current low stock levels.

1.3 Methods

In order to achieve the defined objectives, different methodologies were adopted. The following section describes the multiple steps taken to make progress through this project.

Initially, based on the described objectives, two research questions were formulated:

- Which methodologies and technologies exist to add new functionalities to an existing production process?
- Which industrial communication protocols exist? Which ones can be applied between the AGV and the SIF-400 system?

Each of the research questions focusses on an essential aspect of the project. The first question relates to the implementation methodology, since both resources, namely the existing production system and the AGV, are two independent systems that have their own control unit and user interface (UI). Besides, the main requirement is to avoid making modifications to those resources. The second question is related to the first, as the use of appropriate communication protocols could be an important aspect in the resulting implementation method.

To start, research about the current state of the art was carried out. Articles associated to the above-described research questions were searched and analysed. Those articles were mainly from the KULeuven Libis and UHasselt online libraries.

Following the research, and based on the obtained results, an initial control system architecture was proposed to the stakeholders. At the same time, a first acquaintance with the actual technologies was organised. This also included the development of a test implementation to evaluate the resulting communication protocols in practice. Concerning this, the programming language Python, which is used within the integrated development environment (IDE) PyCharm, was chosen. This programming language was preferred mainly because of the large collection of publicly available libraries, from which a significant part facilitates the application of various communication protocols.

The following period was devoted to further discussion of the final system architecture, after which the development of the first objective was started. As it turned out during the test phase, and based on the additional results regarding the second research question, Python was defined as the programming language for the development of both objectives. Moreover, a centralised control service was determined as the system architecture for the solution to be developed. The research and the results obtained are further discussed in chapter *3 Research*.

Simultaneously with the development of the main Python program, progress was made on the operator's mobile signalling application. Initially the focus lied on the design of the UI after which the communication between the service and the application was investigated.

Concerning this, different frameworks, associated with a specific IDE, were identified as possible candidates for the application development. Finally, the Flutter software development kit (SDK), used by means of the Android Studio IDE, was found to be the appropriate technology for the development of the operator's mobile signalling application due to its cross-platform feature. Section *3.3.2 Mobile Signalling Application* is devoted to the Dart programming language, which is used within the Flutter SDK.

Once the requirements of automated recuperation of finished products were met, the focus was shifted to the automated stock refilling. The centralised control service and the operator's mobile signalling application were expanded to meet the requirements of this second objective. The same methods and technologies were used to develop this objective as for the first objective.

Finally, optimisations were made to overcome some of the challenges encountered. Furthermore, the entire implementation was thoroughly tested, problems were solved and improvements were made. At last, everything was documented and the service was implemented in a Docker container on the server.

1.4 Outline

The following chapters will discuss all details of the followed method. To start, chapter *2 Materials* gives a detailed explanation about the existing hardware and software. Furthermore, chapter *3 Research* answers the formulated research questions. Chapter *4 Implementation* goes into detail about the chosen system architecture and implemented software to meet the objectives. In chapter *5 Results* the comparison is made between the manual recuperation and stock refilling, and the recuperation and stock refilling with an AGV. Finally, chapter *6 Conclusion* concludes this Master's thesis and describes options for future work.

2 Materials

The aim of this Master's thesis is the integration of an AGV into the SIF-400. Hence, both the mobile robot and the smart manufacturing system form the key materials considering this project. Both resources were provided as independent systems, already embedding their respective execution systems and other programs. This chapter describes both resources, more specifically their integrated hardware, combined with the standard provided software platform, and the capabilities they offer.

2.1 SIF-400

The SIF-400 production system, built by SMC International Training simulates a highly automated smart factory. This manufacturing system was built for education purposes, offering trainees the opportunities to learn the key concepts of Industry 4.0. It consists of 13 independent stations, from which 12, identified as SIF-401 until SIF-412, form the actual production system. The last separated station, labelled as SIF-413, was designed for recycling purposes. It is used by an operator in order to disassemble the finished goods, dispatched by the SIF-412 station, to raw materials, whereafter those materials are used to replenish the respective stations. Hence, reusability is achieved. Figure 3 shows a picture of a possible arrangement of some SIF-400 stations.

Figure 3 Example of arrangement of SIF-400 stations [1, p. 1]

The paragraph below explains briefly the workflow the manufacturing system, whereby the task of each station is discussed. A schematic overview of the SIF-400 can be seen in figures 1 and 2.

The SIF-401 initializes the process by providing a small blue pallet on which either a round or square container is placed. Thereafter, this container is filled with pebbles by the SIF-402 or with liquids in the SIF-403. The SIF-402 is responsible for the dispensing of solid goods, represented by three different coloured pebbles. On the contrary, the SIF-403 is in charge of filling the containers with two possible types of liquids, more specifically water and oil. Aside, containers with handmade goods can also be handled. Those are directly placed by the SIF-404 on a provided empty blue pallet. Following the filling, the SIF-405 closes the container with respectively a round or square cap. The SIF-406 implements the first warehouse station, designed for storing single containers. Furthermore, the SIF-407 is in charge of labelling those containers by means of a sticker with a quick response (QR) code. Afterwards, a collaborative robot (COBOT), as an integrated part of the SIF-408, regroups multiple containers into a large pack, whereafter this pack is sent to the remaining stations. Those remaining stations make part of the dispatching section. Firstly, a second storage station is implemented by means of the SIF-

409. However, contrary to warehouse station SIF-406, single packs are stored here instead of single containers. Secondly, the SIF-410 station is responsible for wrapping those packs with foil. The second last station, namely the SIF-411, labels the packs with a sticker including a barcode and a near-field communication (NFC) chip. Thereby, it forms the finished good. Finally, those finished goods are dispatched from one of three possible docks of the SIF-412.

2.1.1 Hardware

As mentioned above, all stations were designed to work independently. Each station has its own execution system, implemented into a B&R programmable logic controller (PLC). The system is responsible for the monitoring of the sensors and the control of the actuators on that station. Regarding this, it takes decisions based on the data communicated with the manufacturing execution system (MES) of the SIF-400, called the SIFMES. The SIFMES is further discussed in section *2.1.2 Software*. Furthermore, an HMI enables the control of the available processes and the supervision of the remaining stocks.

The stations communicate with the server, whereon the software was integrated, by means of a local area network (LAN). The communication is based on the open platform communication unified architecture protocol (OPC UA), a leading M2M protocol in the industrial context. Aside from this cabled network, a wireless access point, integrated in the SIF-401, enables the wireless connection with the software.

2.1.2 Software

The software of the SIF-400 systems is mainly implemented as a MES. A MES permits the management of simulated orders and the coordination of the different stations, by means of a service, an application programming interface (API) and a database.

As aforementioned, the MES of the SIF-400 is called SIFMES. It consists of three subsystems: the MovService, the database and the API. Firstly, the database stores the data used during the production processes, more specifically the states of the stations and the orders data. The stations retrieve this data by means of the MovService, which form a gateway between those endpoints. Hereby, the communication is based on the OPC UA protocol. Furthermore, the API enables other services to interact with the MovService, and hence with the data stored in the database. Finally, this API is used by the web interface, which provides a dashboard of the SIFMES. Figure 4 shows an overview of the structure of the SIFMES.

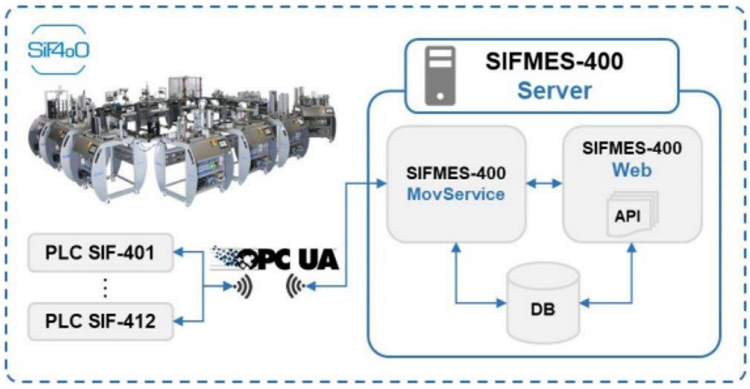

Figure 4 Architecture of the SIF-400 MES [2, p. 12]

2.2 AGV

Aside from the stationary system, an AGV was provided by SMC International Training as part of the complete SIF-400 configuration. This AGV is considered as the 14th station, hence labelled as SIF-414. The main purpose of the AGV is the transport of goods, whereby it automatically follows the most appropriate path using its sensors in combination with a path finding algorithm. Figure 5 shows a picture of the AGV provided by SMC International Training.

Figure 5 Picture of the provided AGV [1, p. 1]

2.2.1 Hardware

The AGV is equivalent to an upgraded version of the TurtleBot2. Firstly, a Kobuki mobile base provides the structure, battery and motion system of the AGV. Secondly, an Orbecc Astra 3-dimensional distance sensor and a Hokuyo light detection and ranging or laser imaging detection and ranging (LIDAR) device form the sensor inputs for the path finding algorithm. Furthermore, the execution system is implemented by means of a next unit of computing (NUC). Finally, an Asus network router provides the AGV of a personal wireless local area network (WLAN). In addition, a universal serial bus (USB) hub increases the number of available input ports of the NUC [3].

2.2.2 Software

Onto the NUC, the Ubuntu 16.04, a Linux based operating system, is installed. This operating system hosts an open-source robot operating system (ROS) framework, which manages the AGV. ROS is based on scripts or services which are called nodes. Besides, the principle of topics is used to facilitate the communication between the nodes. Regarding this, a node can subscribe to any available topic. From that moment, the node is triggered whenever a message is published to that topic. On the other hand, a node can publish a message to any existing topic, and hence trigger all other nodes which have subscribed to it [4].

The AGV presents different functionalities. Firstly, it has a simultaneous localization and mapping (SLAM) algorithm which is used to constitute a map of its surroundings based on the incoming sensors data. Besides, the algorithm also detects obstacles which the AGV should avoid, matches data from multiple sensors to ensure the location of these obstacles, predicts the relative position of the robot to these obstacles, updates the map and localizes itself in real-time [5]. In order to implement this algorithm, the ROS gmapping package is applied [6]. Finally, an automatic docking functionality is available as well.

In addition to this system architecture, SMC International Training added a web interface to facilitate the control of the AGV. This interface integrates a control panel on which the telemetry of the AGV is monitored. Besides, the control mode, for example the automatic movement to a predefined position, is selectable. Other functionalities are also implemented.

3 Research

Prior to the actual implementation, research was conducted in order to collect the most convenient theoretical models. With regards to the predefined problem statement and objectives, two principals were identified as the focus of the research. Namely, the existing and appropriate implementation methods for integration of new functional modules within an existing automated environment together with the contemporary communication protocols for communication between the new and existing architecture. This chapter discusses both subjects, from the sketch of the context, to the actual conclusion, passing by the obtained results.

3.1 Implementation methods

In the first place, it is crucial to determine which integration approaches exist, and which are appropriate solutions for the given scenario. Therefore, the following research question was drawn up: 'How to integrate a new functional module into an existing production system?'. This section discusses the obtained results related to this research question.

Firstly, theoretical models which can form possible solutions to the problem are described. Thereafter, the system architecture of the existing SIF-400 manufacturing system is detailed in relation to the obtained theoretical models. Finally, a conclusion is drawn wherein the most appropriate solution is chosen.

3.1.1 Theoretical model

In the context of Industry 4.0, innovative implementation methods are proposed. These proposals are all the more important as manufacturers aim to remain competitive, in a fast-evolving industrial sector. This objective is even harder to attain as the provided goods should continually meet rapidly changing consumer demands. Consequently, manufactures are forced to adapt their production approach away from mass-production and towards mass-customization [7]. By realizing mass-customization, companies provide products, which have a lower lot size and a shorter product life cycle, but with the low unit costs and a high efficiency associated with mass production [7]. However, carrying out this manufacturing technique entails continuous organizational and operational improvements. As a consequence, tremendous costs are regularly generated.

The engendered charges raise even higher when conventional production systems are used. These production lines generally consist of a permanent sequence of dependent modules, managed as a whole by a centralized control unit. Therefore, even minor adjustments possibly become unconceivable. As part of the contemporary implementation principles, agent-based systems and plug-and-produce modules gain a considerable popularity. With regards to this research, a simplified implementation is achievable if both principles are considered.

A Agent-based systems

An agent-based system differs significantly from a standard manufacturing line in its architecture. As the name describes, self-contained entities, called agents, form the base of an agent-based system. Generally, each individual agent module achieves a precise operation,

corresponding to an established step in a production process. Based on the required use case, those agents are combined into a singular production line.

A modular production system (MPS) is a concrete illustration of an agent-based system applied in the industrial context. MPS's agents are standardized modular machines, each belonging to a specific module class. The class is part of a limited set, containing: process machine primitives, motion units, modular fixturing and configurable control systems [7]. The MPS principle claims that, by combining modules from different classes, a fully automated manufacturing system implementation is achievable [7]. Figure 6 shows an example of a module set, represented as the module pool.

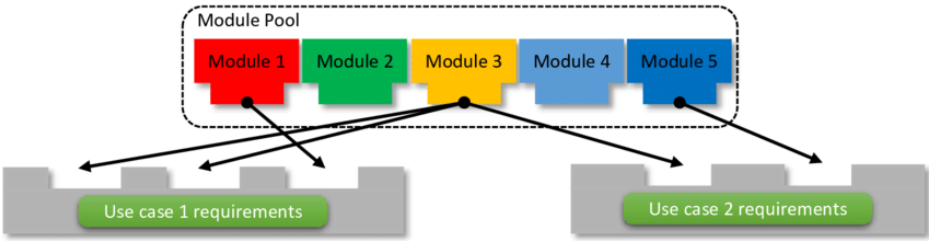

Figure 6 Example of a module pool of a MPS [7, p. 3]

B Plug-and-produce modules

The term 'plug-and-produce' is directly derived from 'plug-and-play', found in the computer jargon, defining the ease with which a controller can be plugged in and used directly [8]. The definition remains almost the same in the industrial context, however the name 'plug-and-produce' is used. In systems integrating such plug-and-produce modules, the focus lies on the communication between them to ensure the proper functioning. Based on the state of the other modules, a module is capable of deciding which action is required.

The term 'plug-and-produce' is directly related to the principle of the agent-based system, as the plug-and-produce module could be seen as an agent. However, the difference lies in the practicality of the system. As the name suggests, a plug-and-produce module works immediately after having been plugged in. Other advantages over the traditional systems are discussed in the following section.

C Advantages

Agent-based systems bring a first advantage over the traditional infrastructures. The modular aspect of the manufacturing line is enhanced with the use of distinct modules. Whenever the manufacture of a different good is requested, as a consequence of a recent evolution of the consumer interest, modules can be replaced on a time efficient manner. Accordingly, this approach brings about lower charges.

Another cost-efficient advantage, concerning the reconfigurability, is obtained with the use of plug-and-produce modules. Conventional systems are generally built around a centralized control unit. This unit is in charge of coordinating the flow of the produced goods through the available processes. Although this approach appears convenient in usual systems, issues arise when reconfigurations of the manufacturing line become regular. In such cases, the systematic reconfiguration of the control unit would lead to enormous expenses.

The principle of decentralization, frequently related to the principle of plug-and-produce, was presented as a possible solution. This principle consists in transposing the control of the whole process from a central unit to the independent modules. This idea aims to reduce costs associated with reconfiguration, as the linked modules embed their own control system. Therefore, the need for reconfiguration of a centralized control unit whenever a manufacturing line adjustment occurs is dismissed. Figure 7 shows an example of a decentralized system.

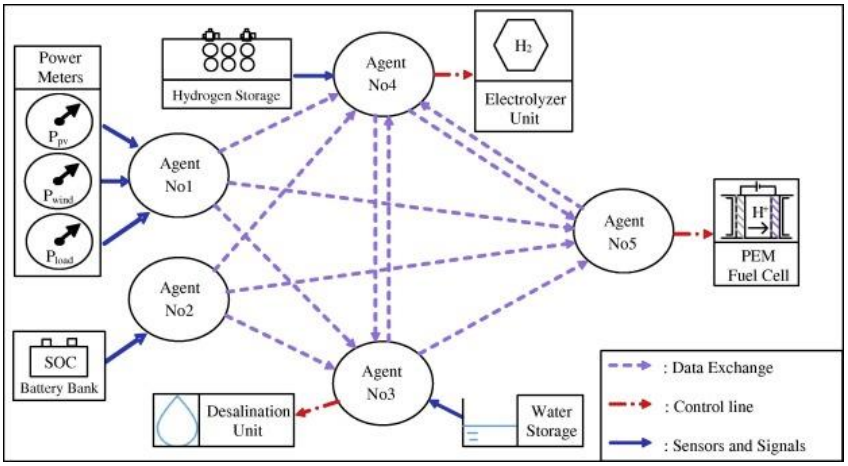

Figure 7 Example of a decentralized system with the associated communications [9, p. 171]

3.1.2 SIF System Architecture

Although a general overview of the SIF-400 system was already discussed in chapter *2 Materials*, this section further clarifies the system architecture of the resource. Regarding the obtained results, the SIF-400 production system can be classified as an agent-based system, where each station is designed as an agent. These stations embed their own control mechanism, implemented by means of a PLC and the necessary sensors and actuators. Furthermore, just as defined by an agent-based system, the individual modules of the SIF-400 do not require a centralized control unit for the management of the operations taking place within this module, but have a separate control unit locally available to handle this job. All processes in the different stations are executed based on the data exchanged with a shared database. The MovService, mentioned in chapter *2 Materials*, serves only as gateway between the stations and the database.

Regarding the integration of the AGV, the movable robot could be implemented as an agent into the existing system. The vehicle already embeds a control system, more specifically the ROS core, and the necessary communication equipment, more specifically the router. Such as the SIF-400 stations, the AGV could be programmed to communicate with the existing database and thereafter action based on the received data.

3.1.3 Conclusion

The previous sections introduced two strategies in the context of the implementation of an Industry 4.0 manufacturing system. The presented paradigms were designed in order to counter excessive costs due to reconfigurations occurring because of fast-changing consumer demands. The focus of the principles lies in the enhancement of the modularity aspect as well as the decentralization of the control unit. In addition, the link between the existing SIF-400 system and the agent-based model was made. Hereby, the integration of the AGV as an additional agent was proposed.

Although these solutions seem applicable in the context of the implementation of the AGV in the existing SIF-400 system, a project constrain forces the adoption of a different approach. With regard to this project, both systems should be interconnected but the number of modifications brought onto the respective systems must be minimized. As the application of both principles described above would require at least some improvements to the existing entities, these solutions are considered to be irrelevant for this Master's thesis assignment. The modifications at the side of the AGV would require the integration of a new logic unit. On the other hand, at the side of the SIFMES database, it would require to be extended with new variables, enabling the interactions with the AGV.

Considering the previous statements, another implementation system architecture was preferred. This system architecture is based on a centralized control service, communicating with both existing systems, as well as with the operators' mobile signalling application through the use of an MQTT broker. A more detailed description of the architecture can be found in section *4.1 System Architecture*. This approach does not meet the requirements of the contemporary principles, however the control service was designed in order to offer the freedom to possibly adapt the project regarding the theoretical models. Finally, these models, and the proposed solution, are interesting regarding a possible multi-AGV expansion.

3.2 Communication protocols

With regards to the previous research question, a second about the communication protocols was drawn up: 'Which industrial communication protocols exist and are suitable for the communication between the AGV, the SIF-400 and the operator's mobile signalling application?'. This research was subdivided by system. Hence, the communication protocols of the SIF-400, the AGV and the mobile signalling application are discussed in the following sections.

3.2.1 SIF-400

As mentioned in chapter *2 Materials*, the SIF-400 consists of 13 independent modules, which action based on the data stored in the SIFMES database. The communication is implemented by means of the MovService which handles as a gateway. In this regard, the used communication protocol is OPC UA. OPC UA was released as a successor to open platform communication (OPC). It uses web service technologies to provide a platform independent protocol that integrates all the functionalities of the OPC specifications [10]. Furthermore, OPC UA uses a client and a server to establish a communication between two systems, whereby a client requests and/or publishes data to/from the server [11]. In the context of the SIF-400, each station's PLC acts as an OPC UA server, while on the contrary the SIF-400 server acts as an OPC UA client. More specifically, the MovService handles the communication between the stations and the SIFMES's applications. To end, OPC UA is also supported by many programming languages such as C, C++, C#, Java, JavaScript and Python [12].

Related to the conclusion of the first research question, it was decided to implement a centralized control service. A direct OPC UA communication between this service and the individual station was determined as a first alternative. However, considering this option, future updates of the PLC programs by SMC International Training may incur. For example, a renaming of PLC's variables, which would possibly cause disfunctions in the resulting implementation.

Therefore, a second option was proposed. This second proposal consists of the update of the already existing API in order to make the stock levels of each station available. Considering this option, the variables made available by means of this interface would at all times remain the same, even if updates of the PLC programs would take place. Table 2 shows an overview of the advantages and disadvantages of both discussed options. The term "project delay" refers to the time one has to wait for SMC International Training to complete the update of the API.

Table 2 Comparison between direct OPC UA and API communication

	OPC UA	API
Advantages	▪ Direct access to all PLC variables ▪ Flexibility ▪ No project delays	▪ Straightforward implementation ▪ Maintainability ▪ Centralized variables ▪ Separation of software
Disadvantages	▪ Less maintainable ▪ Requires extra programming	▪ Possibly a project delay ▪ Less flexible

Firstly, the table includes the ease of implementation and use, which is equivalent for both options. Furthermore, the term centralized variables refer to the centralization of all the stock levels in the SIFMES database, instead of accessing them directly from each individual PLC. Finally, the flexibility disadvantage related to the API concerns a less efficient addition of other necessary PLC's variables in contrast to the direct option.

Following different discussions with the stakeholders, and based on the achieved research, the API option was chosen as the appropriate solution for the communication between the SIF-400 system and the resulting implementation. Consequently, the existing API was updated, and by means of a single API call the stock levels of each station could be retrieved.

3.2.2 AGV

With regards to the AGV, a web interface was already provided by SMC International Training. This interface was integrated by means of a web socket. Hence this communication protocol forms the first option. The implementation of a web socket is achievable using the rosbridge server package, which provides a full-duplex communication between a client and the AGV. This package permits the client to act like an internal node, whereby it is possible to subscribe or to publish to an available topic. Regarding this, the messages and responses are constantly converted to JavaScript Object Notation (JSON) [13]. In addition, any programming language that supports IP sockets is able to communicate via rosbridge [14]. To end, some programming languages provide specific libraries which facilitate the communication, for example the roslibjs library for JavaScript or the roslibpy package for Python. The latter is interesting in the context of this Master's thesis.

The second option, proposed by SMC International Training, was the application of the OPC UA protocol. Here also, packages or libraries exist for multiple programming languages: freeopcua [15] for C++ or python-opcua for Python.

In the end the web socket option was chosen over the ROS OPC UA communication as the most convenient solution because of the less overhead it causes. In addition, it was assumed that SMC International Training had more knowledge about the rosbridge web socket since the existing web interface actually already is using this communication protocol.

3.2.3 App

The last communication protocol to be determined concerns the communication with the operator's mobile signalling application installed on his or her mobile device. The first proposed solution is rosjava. Rosjava is a client library, similar to the roslibpy package for Python, which is often applied in order to create Android applications. Thereby a direct connection between the ROS system and the mobile signalling application is achieved. However, since the decision was taken to develop a centralized control service, this approach was not preferred. The direct connection between the AGV and the mobile signalling application would go against the centralization principle of the defined approach. Hence, further research was necessary. This research resulted in five possible communication protocols: hypertext transfer protocol (HTTP) requests, message queuing telemetry transport (MQTT), constrained application protocol (CoAP), advanced message queuing protocol (AMQP) and data distribution service (DDS).

Firstly, the HTTP request was investigated. This communication protocol is applicable by means of the integration of an API in the service. Thereby the mobile signalling application would act like a client, and hence call the API via HTTP requests. The considerable disadvantage of this method lies in the overhead induced by periodic calls that it would require in order to continuously check if a notification has to be displayed.

Secondly, the MQTT protocol was considered. MQTT is a lightweight messaging protocol that is widely used in the IoT and Industry 4.0 [16]. As for the ROS system, the MQTT protocol is based around topics where the clients can subscribe or publish to. However, a central MQTT broker, which manages all communications from and to the clients, is required. Thereby all clients are connected to the MQTT broker, hence there is no direct communication between clients. The important advantage lies in the publisher-subscriber communication method, whereby no periodic calls are required anymore resulting in less communication overhead.

Furthermore, the CoAP was researched as a third option. CoAP is a client server protocol which is used for communications between resource-constrained devices like wireless sensors. It is based on the commonly used representation state transfer (REST) model and is optimized for IoT and M2M applications [17].

The second last proposed solution is the AMQP protocol. As for the MQTT protocol, it is based around publishing and subscribing to messages. However, contrary to the MQTT protocol, the AMQP protocol does not require a broker to manage all communications. Hereby, a client publishes a message to an exchange, whereafter this message is copied by the exchange to queues. Finally, other clients can subscribe to these queues or fetch messages from these queues to receive the published messages [18].

The last option concerns the DDS protocol. DDS is a data centric publish subscribe messaging protocol and is optimized to provide real-time information [19]. The difference with the other publisher-subscriber protocols lies in the way the communications are managed. With regards to the DDS protocol, it is the DDS domain which manages all communications. In addition, all topics are available via this domain.

As conclusion five different communication protocols were proposed as communication solution between the service and the operator's mobile signalling application. The HTTP-request could be a logical choice since it is the standard in service-oriented architectures. However, the periodic calls to the API would cause overhead and could have an impact on the performance of the mobile signalling application. Consequently, further research on the performance differences between the different message protocols was conducted. Tetsuya Yokotani and Yuya Sasaki [20] state that information-centric networking (ICN) architectures are faster and require less resources. Thereby, there are many communication protocols that are based on ICN architecture, so further research was needed. Nitin Naik [21] compares MQTT, CoAP, AMQP and HTTP. Figure 8 shows a relative comparison between the different protocols based on the message size versus the message overhead. The x-axis shows the message overhead on a relative scale, whereas the y-axis shows the message size on a relative scale.

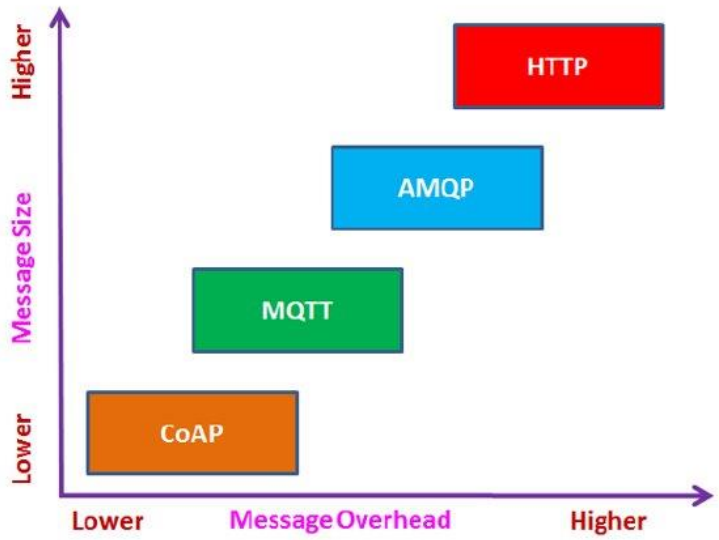

Figure 8 Comparison between the communication protocols: Message size vs. Message overhead [21, p. 4]

From figure 8 it is possible to conclude that CoAP and MQTT have the lowest message size and message overhead in contrast to AMQP and HTTP. Figure 9 shows the relative comparison based on the reliability versus the interoperability. The x-axis shows the interoperability on a relative scale, whereas the y-axis shows the reliability or quality of service (QoS) on a relative scale.

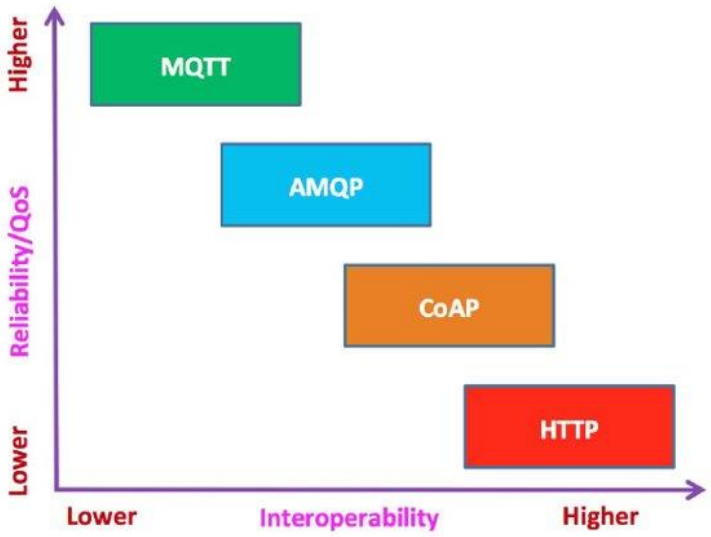

Figure 9 Comparison between the communication protocols: Reliability vs. Interoperability [21, p. 5]

From figure 9 is possible to deduce that MQTT has the highest reliability but the lowest interoperability. On the contrary, AMQP, CoAP and HTTP provide a higher interoperability but a lower reliability.

As a conclusion, based on the figures above, MQTT was chosen over the three other communication protocols. Firstly, MQTT has one of the lowest message size and message overhead while it has the highest reliability of the discussed communication protocols. Secondly, the interoperability of MQTT is lower than the others but, regarding this implementation, the reliability is more important than the interoperability. Finally, DDS could be a solid alternative since it shares characteristics with MQTT but the chosen programming language for the mobile signalling application did not support this protocol.

3.3 Programming languages

From the previous conducted research, a centralized control service, which communicates with the available resources, was defined as the most convenient solution for the implementation. Therefore, an appropriate programming language was necessary. The section below describes the research which was carried out in order to determine the most convenient programming language for the development of the control service.

Besides, the development of a mobile signalling application was required. Here also, different mobile programming languages exist. Hence, a similar study was realized to define the best programming language regarding this application. The sections below describe those studies.

3.3.1 Control Service

The control service is responsible for the communication between the different systems. As a result of the conducted research regarding the communication protocols, the following protocols were chosen: an API for the communication with the actual SIFMES, the rosbridge WebSocket for the control of the AGV and MQTT as the communication method for the mobile signalling application. Consequently, the selected programming language should require the support of those protocols. In addition, because of the possible integration of an automatic refilling extension regarding station SIF-408, the programming language should also include the possibility of a direct OPC UA communication. Table 3 shows an overview of the supported programming languages per communication protocol.

Firstly, a vast amount of programming languages, including C++, C#, Java, JavaScript, PHP, Python, integrate the support of HTTP-request for API interactions. Hence, the API communication does not form the bottleneck regarding this research. On the contrary, only Java, JavaScript and Python support the use of a rosbridge WebSocket [22]. However, custom libraries written in both C++ and C# make the use of this communication protocol in those programming languages possible as well [23]. Furthermore, similarly to the HTTP-request, a large collection of programming languages integrate the support of the MQTT communication. Finally, since OPC UA is widely used in the industrial context, also the most well-known programming languages implement the required support [12]. Table 3 shows an overview of the supported programming languages per communication protocol.

Table 3 The supported programming languages per communication protocol

	HTTP-requests	Rosbridge WebSocket	MQTT	OPC UA
Programming languages	▪ C++ ▪ C# ▪ Java ▪ JavaScript ▪ PHP ▪ Python ▪ …	▪ (C++) ▪ (C#) ▪ Java ▪ JavaScript ▪ Python	▪ C++ ▪ C# ▪ Java ▪ JavaScript ▪ PHP ▪ Python ▪ …	▪ C ▪ C++ ▪ C# ▪ Java ▪ JavaScript ▪ Python ▪ …

From table 3 it is possible to conclude that the limiting protocol is the rosbridge WebSocket. Hence, JavaScript is eliminated of the list of possible candidates since it is mainly used for the development of web applications. The resulting options are both Java and Python. Regarding this, Java is a more static language which needs to be compiled before runtime which makes it a faster programming language but also harder to use and to read. On the contrary, Python is more dynamic and interpreted at runtime which makes it easier to use and to read. In addition, Python has become more popular in recent years. Finally, the Python community offers a large collection of useful libraries [24].

Based on the conducted research and hence on the obtained results, Python was defined as the most convenient programming language for the development of the control service. The ease of use, the popularity and the readability of the programming languages formed the main arguments concerning this decision.

3.3.2 Mobile signalling application

Concerning the development of the operator's mobile signalling application, there were several possibilities. The final product is intended for operators who are in charge of transferring the finished goods and the raw materials between the SIF-400 stations and the AGV. Therefore, the designed software should only include basic functionalities, accessible by means of a straightforward UI.

In the state of art, multiple programming languages combined with an appropriate framework offers numerous attractive advantages. However, each technique has its specific disadvantages, which should be taken into account. Table 4 shows four potential SDKs in the context of this project.

Table 4 The supported programming languages and supported operating systems per mobile application SDK

	Android SDK	Swift SDK	Kotlin SDK	Flutter SDK
Programming languages	Java	Swift	Kotlin	Dart
Operating system(s)	Android	iOS	Android iOS	Android iOS

The Android SDK and the Swift SDK, presented in the first two columns, offer both an effective manner to develop a performant software application. However, both solutions offer only the possibility to compile onto a single operating system, Android and IOS respectively. Hence, both candidates exclude the recent, but fast-evolving, concept of cross-platform development. This concept permits the deployment of a single-code based application onto multiple platforms. The two last presented SDKs, namely Kotlin and Flutter, do support this method, which are much more effective and time-saving. However, Kotlin continues to require the development of native code to apply certain functionalities, whereas the use of platform-specific code is limited when using Flutter [25].

Based on its cross-platform compliance, as well as on the already acquired experience of one of the students, Flutter was chosen as SDK for the development of the operator's mobile signalling application. This SDK, using the programming language Dart, offers all the required tools to build the appropriate software.

4 Implementation

This chapter describes the resulting implementation. The first section discusses the actual implemented system architecture. This section includes the modifications applied to the different resources. The second section explains the developed software of both the control service and the mobile signalling application.

4.1 System architecture

Following the obtained results from the research in combination with multiple discussions with the stakeholders, the undermentioned final system architecture was obtained. This architecture is composed out of five elements: the actual SIF-400 system (top left-hand side figure 10), the personal computer (PC) (centre figure 10), considered as the server; the AGV (right-hand side figure 10) and the operator's smartphone (bottom side figure 10). The fifth element corresponds to the SIF-408 (left-hand side figure 10). The presence of this station as a separate element, in the schematic overview, is discussed in the corresponding section. All the other elements are as well discussed below, except for the SIFMES system on which no changes were applied.

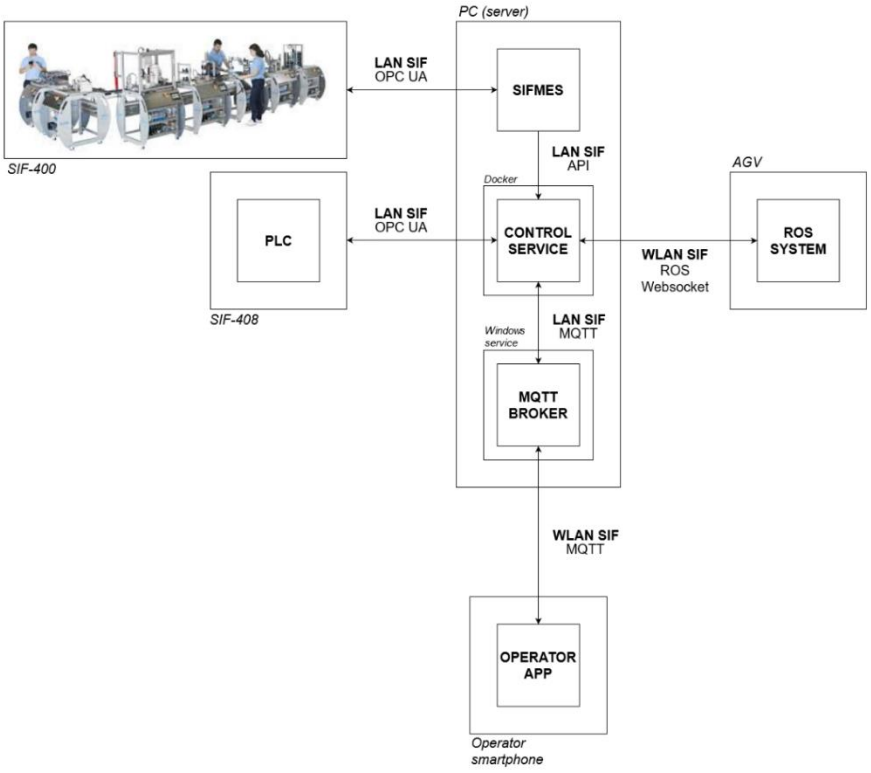

Figure 10 Schematic overview of the resulting system architecture

4.1.1 PC (server)

The PC is considered by SMC International Training as the server, whereon initially only the SIFMES system, comprising the SIFMES web application, the SIFMES database and the MovService, were accommodated. It is connected to the LAN of the SIF-400. Hence the communication between the SIFMES and the SIF-400 stations is fulfilled. As defined by the main requirement regarding the implementation, no modifications were applied to the existing software, in this case the SIFMES. Hence this element will not be discussed further, however more information about the MES system associated with the SIF-400 system can be found on the site of SMC International Training [26].

The implementation of the transport solution required the integration of two new services onto this server. Firstly, the control service, implemented as a centralized unit, is responsible for the logical side of the solution. Secondly, a MQTT Broker, implemented as a continually running Windows service, behaves as a central communication unit between the logical side and the UI, achieved by the mobile signalling application. Both elements, with their associated connections, are further described below.

A Control Service

The control service centralizes all the communications and handles based on the exchanged data. Firstly, the communications with both the SIFMES and the MQTT broker, which are located at the same IP address, are achieved over the LAN of the SIF-400. As discussed in the section *3.2.1 SIF-400*, the communication between the control service and the actual SIF-400 is achieved by means of an upgraded version of the SIFMES API. The API call results in a JSON response, which can easily be decoded in Python. The communication with the MQTT broker is logically based on the MQTT protocol. Furthermore, the specific communication with the SIF-408 is also realised by means of the LAN of the SIF-400. Hereby, a direct OPC UA communication, which enables the reading and writing of PLC variables, is implemented. Finally, the WLAN of the SIF-400 permits the communication with the AGV. This communication was initially achieved over the WLAN of the AGV. However, in order to fully apply the centralisation principle, it was decided to use the SIF-400 networks, more specifically the LAN and the WLAN, for all the necessary connections. This communication is based on the ROS Websocket.

B MQTT Broker

Besides the logical software, an Eclipse Mosquitto [27] MQTT broker was implemented in order to enable the communication with the operators' mobile devices. The broker was installed as a continually running Windows service.

The MQTT clients consist, on the one hand, of the control service and, on the other hand, of the multiple operators' mobile devices. In order to fulfil a proper communication between all the clients, a handshake principle, which is further discussed in section *4.2.1 D Communication with operators*, was implemented in both the control service and the mobile signalling application.

Furthermore, the QOS was set to one to guarantee that a message is delivered at least once to the receiver. In this regard, the publisher of a message stores the message until it receives an acknowledgement from the broker that the message was received. If the publisher does not get an acknowledgement within a predefined amount of time, it resends the message.

Finally, credentials were used to secure the MQTT communication. Hence, unauthorized access will be denied.

4.1.2 AGV

As for the SIFMES, and in conformity with the requirements, no considerable software modifications were applied onto the AGV. However, in order to be able to navigate using amcl the AGV requires a map. Fortunately, the web interface, implemented by SMC International Training, enables the creation of the map, as well as the placement of points of interests (POI) onto it. To start, this interface was used to create the map. Furthermore, POIs, corresponding to places near the desired stations, were defined. In addition, this map was manually adjusted to delimit the allowed working area. More specifically, the walking paths as well as other systems were fenced off. Figure 11 shows the resulting map, whereon the SIF-400 stations are highlighted. The white areas correspond to allowed area, whereas the grey area is off-limits for the AGV.

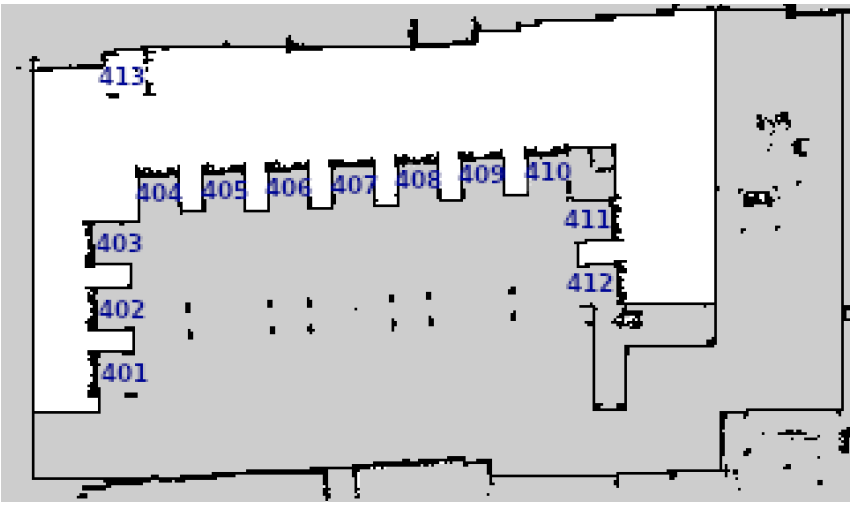

Figure 11 Resulting SLAM map of the SIF-400 environment

Regarding the communication, no further implementation was necessary as the used rosbridge WebSocket was already integrated for the support of the web interface developed by SMC International Training.

Finally, a single physical modification, more specifically the relocation of the Orbecc Astra sensor, was required to permit the stacking of multiple goods on the AGV, without hindering the view of the robot.

4.1.3 Operator's smartphone

The operator's smartphone, or more generally the operator's mobile device, includes the developed mobile signalling application. The mobile signalling application forms the UI of the control service. It is connected to the SIF-400 WLAN. Hence, the communication with the MQTT broker is achievable.

It is possible to connect multiple mobile devices, via the mobile signalling application, to the MQTT broker. Concerning this, all the connected operators are notified whenever a transfer is required. However, the first operator who accepts the transfer is assigned to it. This logic flow, implemented by means of a handshake, is further discussed in section *4.2.1 D Communication with operators*.

4.1.4 SIF-408

The SIF-408 makes part of the actual SIF-400 production line. However, an extension on this Master's thesis was simultaneously worked out by Wouter Van Rompaey, a colleague student. This extension consists in the automatic unloading of the AGV at station SIF-408 by means of the already integrated COBOT [28]. His implementation requires a direct communication with the SIF-408's PLC. Hence, the SIF-408 was integrated as a separate element into the schematic overview of the system architecture, shown in figure 10.

4.2 Software

4.2.1 Control Service

The control service is an essential element in the realized solution. It is responsible for the communications with all subsystems as well as the logical decisions. The service constantly checks the stock levels of the SIF-400 stations, processes the obtained data, and manages the movements of the AGV and the notifications to the operators.

In order to fulfil both objectives, a specific logical structure was adopted. This structure consists of two threads, the *query filler* and the *query handler*, in charge of respectively creating tasks and handling tasks. A schematic overview of both communicating threads is shown in figure 12

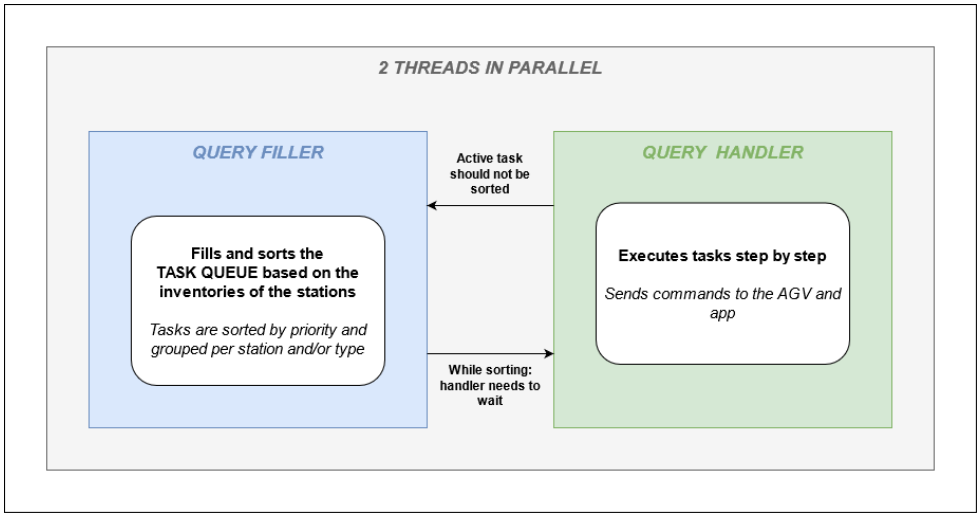

Figure 12 Schematic overview of the implemented communicating threads of the control service

A *task* is defined as a recycling or refilling job, corresponding to respectively the recuperation and refilling objectives, that has to be completed. Furthermore, a task includes a sorted list of steps. A *step*, contrary to a task, is defined as just a single action, for example the movement of the AGV to a specific destination or the triggering of the operator's mobile signalling application. A step, in turn, includes an *inventory* object, which holds the data necessary to execute the step, namely the station and an item. Finally, the *item* consists of an item type, as well as the item position, defining on which position on the station the item has to be placed. The item in the inventory was implemented as optional because movement steps do not require any information about the item, only about the station were the AGV should move to. Figure 13 gives an overview of the models used in the logical process.

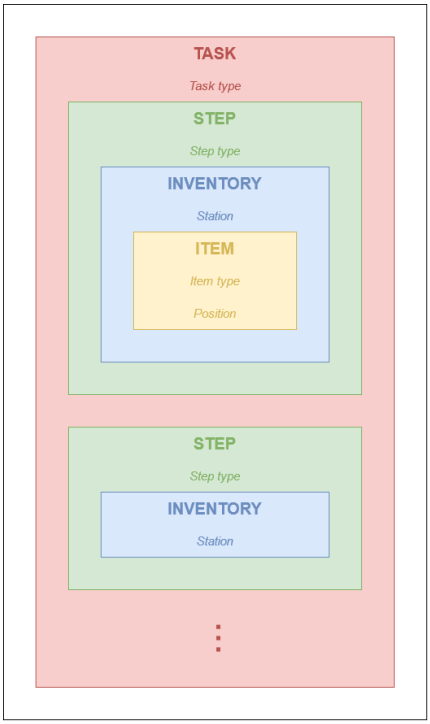

Figure 13 Schematic overview of the models used in the control service

A Query filler

Before the explanation of the working process of the query filler, it is necessary to mention the used variables. Firstly, the *tasks queue* is a queue containing the tasks to be executed. Secondly, the *active task* is defined as the task that is currently being executed. This task does not make part of the *tasks queue* anymore. Furthermore, the *previous task* refers to the task that was previously executed. This variable is used to prevent the previous task to be readded to the *tasks queue*, since there is a small delay before the SIFMES database, and as such the SIFMES API, is updated.

The query filler is a thread in charge of creating the tasks and adding them to the *tasks queue*. The thread constantly checks the stock level of each SIF-400 station by means of API calls. The resulted JSON responses are analysed and converted into *inventory* objects. For each *critical inventory*, defined as an inventory representing a high stock level (SIF-412) or a low stock level (other SIF-400 stations), a *new task* is initiated. Thereafter, the thread checks if the *new task* was already added to the *tasks queue* or if the *new task* corresponds to the *active task*. In addition, the *new task* is compared to the *previous task*. If neither of the above conditions applies, the *new task* is added at the end of the queue.

Finally, the query filler checks if the *tasks queue* was expanded. If it is the case, a sorting method, described in section *4.2.1 C Sorting*, combines and sorts the tasks based on different criteria.

B Query handler

The second thread, namely the query handler, handles the tasks in the *tasks queue* one by one. Meanwhile, the thread pays attention to the battery level of the AGV. The distinction is made between two cases: when tasks are executed and when the *tasks queue* is empty. When tasks are being executed, the critical level of the AGV's battery is checked. If the battery has reached this level, the AGV is sent to dock in order to reload. From then, only if the battery has reached sufficient energy, more specifically has reached the minimum battery level, the AGV is released and it continues its work. On the contrary, if no tasks are available in the *tasks queue*, the AGV is only sent to the docking station if the minimum battery level is reached, and is directly released if a new task appears.

Tasks in the *tasks queue* are one-by-one defined as *active tasks*. Whenever an *active task* is defined, its steps are executed. The execution of a step is handled by a separated method, which calls communication functions based on the step type. Hereby, the state of the emergency stop is taken into account.

C Sorting

The sorting algorithm, implemented as a separate function, is executed whenever new tasks are added to the *task queue*. The sorting method is used for efficiency purpose, more specifically to avoid the AGV from moving back and forth between stations, and such draining much battery. Three different sort and/or combination algorithms are applied.

Firstly, the *tasks queue* is sorted based on the tasks' priorities. Those priorities are assigned at the tasks' initiations and depend on the values entered in the configuration file. Whenever two tasks are combined, as explained below, the highest priority of both tasks is defined as the priority of the combined task.

Secondly, tasks with the same station, but a different low stock level, which follow upon each other in the *tasks queue* are combined together to form one unique task. Consequently, the combined task contains a combination of the *inventories*.

Finally, a maximum of two refilling tasks which follow upon each other in the *tasks queue* are combined together to form one unique task as well.

These implementations were made since it is possible to stack more than one item type onto the AGV. Hence, the number of movements of the AGV is reduced.

During sorting, no *active task* and *active step* assignments are possible in the query handler. This is done since the *task queue* is a shared resource between both threads.

D Communication with operators

An additional requirement consisted in the handling of multiple operators. Considering this, the workflow was defined as follows. Firstly, a transfer request is sent to all the available operators whenever a transfer between the AGV and the station, or the inverse operation, is necessary. Following, an operator is able to accept this request. Consequently, the control service assigns the transfer to the operator who accepted the transfer. Furthermore, the task is completed by the assigned operator. The operators who did not accept the task are redirected to the idle screen of the mobile signalling application.

In order to avoid double assignment, if for example two operators accept the transfer at the same time, a sort of handshake principle was implemented. Figure 14 shows a schematic overview of the handshake. Although this example integrates only two operators, the handshake principle is applicable for a larger group of operators.

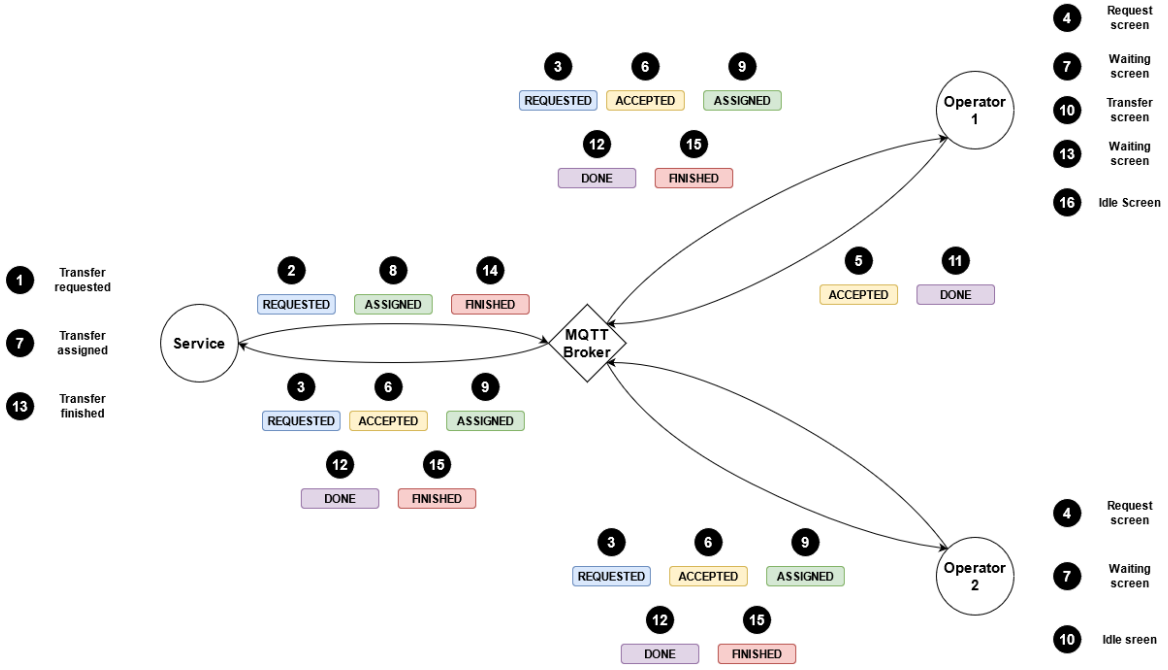

Figure 14 Schematic overview of the transfer handshake between the control service and the operators' mobile signalling applications

It is important to notice that this communication makes use of a single MQTT topic, defined as 'TRANSFER'. Furthermore, there are five possible transfer states: requested, accepted, assigned, done and finished. In addition, each client subscribes to the topic. Hence, all messages which are sent by a certain client are received by all clients, even by the one who sent the message.

More generally, the black numbers within the black circles define the steps to fulfil the handshake. Firstly, the requested message is sent to all connected operators (1, 2, 3, 4). Secondly, operator 1 accepts the transfer, whereby the operator identifier (ID) is sent as payload (5, 6). Following the received accepted message, all mobile signalling applications' view are set to the waiting screen (7). Simultaneously, the control service assigns the transfer to the operator who accepted it (7). As for the accept message, the operator ID is defined as payload of the assigned message (8, 9). Based on this message, each mobile signalling application sets its view to either the transfer screen, if the operator IDs correspond, or to the idle screen if the transfer

was assigned to another operator (10). When the transfer was accomplished, a done message is sent to the control service by the assigned operator (11, 12, 13). Finally, the control service finishes the handshake with a finished message (14, 15, 16).

Besides the 'TRANSFER' topic, three other topics are used for the communication with the operators: the 'OPERATOR' topic, the 'AGV' topic and the 'TELEMETRY' topic. The 'OPERATOR' topic is used to notify the control service whenever a new mobile device is connected. However, currently only a message is printed out by the service whenever a connection is established. Furthermore, the 'AGV' topic is used regarding the implemented emergency stop. By means of the mobile signalling application, an operator is able to start or stop the AGV. Concerning this, a similar handshake principle was applied. It enables the update of all the mobile signalling applications' view when a single operator modifies the AGV's state. Finally, the 'TELEMETRY' topic was added to communicated telemetric data to the UI. Currently, only the AGV's battery percentage is communicated.

4.2.2 Mobile signalling application

The developed operator's mobile signalling application forms the UI of the control service. This section discusses on the one hand the logical implementation of the software and on the other hand the designed views.

A Logic

The logical implementation was based on the model-view-controller (MVC) architecture, from which figure 15 shows a schematic overview. This architecture consists of three communicating elements: model, view and controller. The model corresponds to the data. This data is displayed to the user by the view. The controller connects both elements by providing the data to the view. Whenever the data changes the model notifies the controller. Consequently, the controller updates the view. On the contrary, the view communicates the user interactions to the controller, which then updates the model [29].

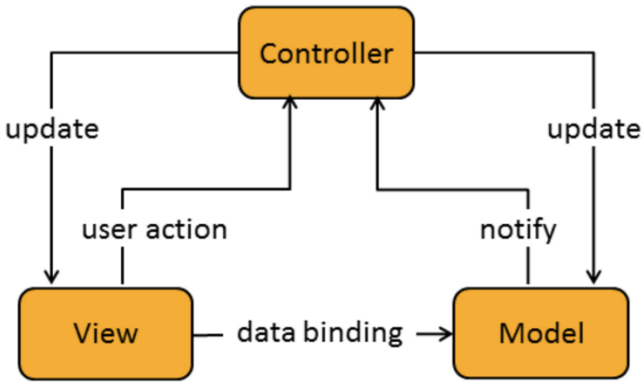

Figure 15 Schematic overview of a MVC architecture [30, p. 1]

Regarding the developed software, the model was implemented by means of two Dart classes, more specifically the *MQTTClient* and the *MQTTHandler*. The *MQTTClient* refers to lowest code layer relative to the MQTT broker. It includes the essential subscribing and publishing functions, implemented by means of the *mqtt_client* package. Furthermore, it notifies the *MQTTHandler* whenever a message on a subscribed topic is received. The *MQTTHandler* implements higher order functions, including the decoding and handling of the received messages. Regarding this, it holds the last received message of each subscribed topic. Moreover, it notifies the controller when it changes one of these messages.

The application includes multiple views, associated with different actions. Those views, which are associated to the state of the application, are further discussed in section *4.2.2 B Views*. The user's interactions on those views are handled by the application's controller, implemented as the *HomeController* class.

The *HomeController* is in charge of providing the data to the view. However, in comparison with a conventional MVC architecture, the state of the application is held by this class. This state is updated based on the *MQTTHandler*'s messages. The application has six different states: connection, no connection, idle, waiting, transfer and emergency. The state is set to the connection state when the mobile application is connecting to the MQTT broker. On the contrary, the no connection state is defined when no connection with the broker was established. The idle state refers to the state in which the mobile application waits for a request from the control service. When a request is received, or when the transfer was assigned to the operator, the state is switched to the transfer state. The waiting state was added with regards to the implemented handshake, explained in section *4.2.1 D Communication with operators*. This state is defined when a response from the control service is expected. Finally, in case of an activated emergency stop, the state is defined to the emergency state.

In addition, the controller includes an *NotificationHandler* as well as an *AppLifecycleStateHandler*. The combination of both handlers enables the sending of a notification when the operator's mobile device is locked. Hence, the operators are always notified, even if one is not actively using his or her device.

B Views

The views associated with the application states were designed to be simple, but complete. The structure of each view consists of a title bar, a body and an action bar. The top bar, shown in figure 16, includes the application title and a progress bar showing the current AGV's battery percentage. This bar is fixed for each view. On the contrary, the body as well as the available actions contained in the action bar differ from view to view. However, the views representing a specific state, for example the connection or the emergency, include a similar simplified body structure. The structure consists of a related image in combination with a proper title and description. Conversely, the transfer view includes a more elaborated body, which consists of an image of the item to be transferred as well as two images representing the source and destination modules. An appropriate title and description are also provided to the user. The following paragraphs describe the views and their associated action(s) one-by-one.

Figure 16 Top bar of the mobile signalling application

The connection view is shown in figure 17. The action bar includes no specific action, but a loading animation is shown, which indicates that the mobile application is connecting.

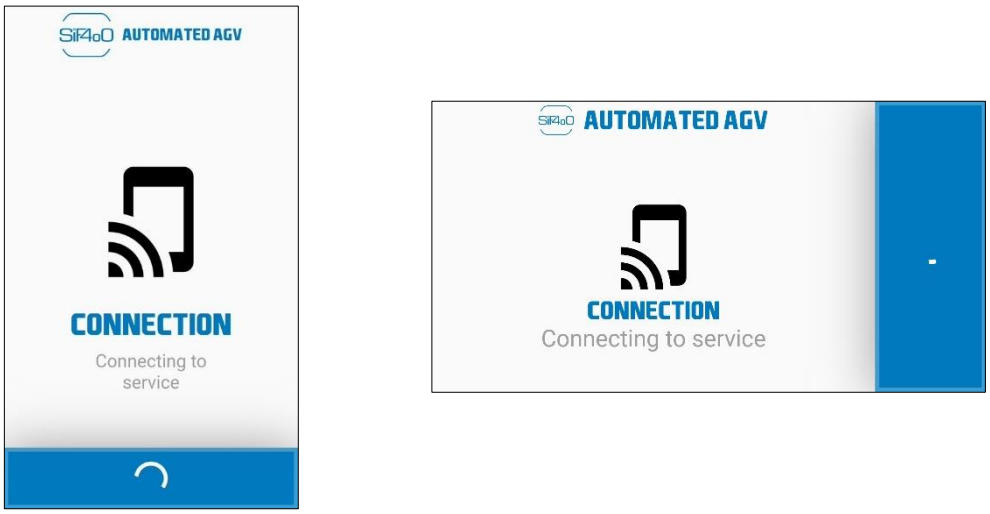

Figure 17 Connection view of mobile application: portrait (left) and landscape (right)

The body of the no connection view, shown in figure 18, informs the user that the connection with the MQTT broker was not successfully established. Hence, the action bar includes a single action button, by means of which the user can retry the connection process.

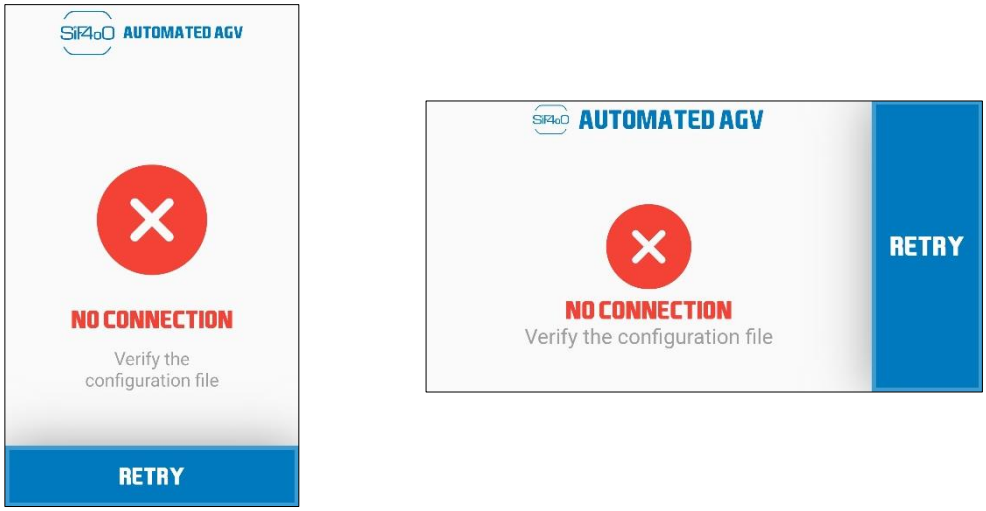

Figure 18 No connection view of mobile application: portrait (left) and landscape (right)

The idle view, associated with the idle state, is displayed when the mobile application waits for a request from the control service. This view is shown in figure 19. The action bar includes an emergency stop action button. This button enables the user to activate the emergency stop and as such momentarily stop the AGV's movements.

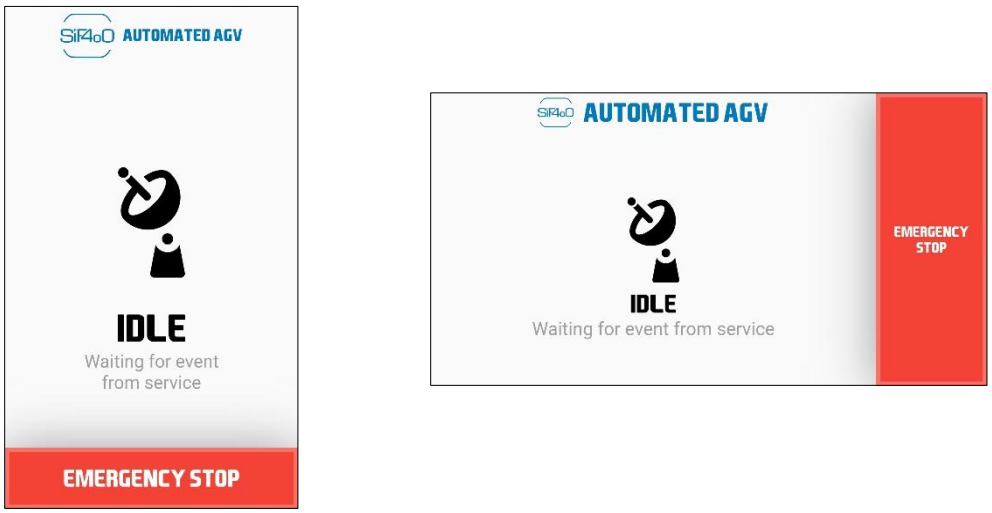

Figure 19 Idle view of mobile application: portrait (left) and landscape (right)

The waiting view, shown in figure 20, is associated to the waiting state, which is defined when a response from the control service is expected. However, as the communication is generally achieved instantly, this view is often almost not visible to the user. Although, in case of miscommunication, the cancel action button enables the user to cancel the waiting process and as such returning to the idle state.

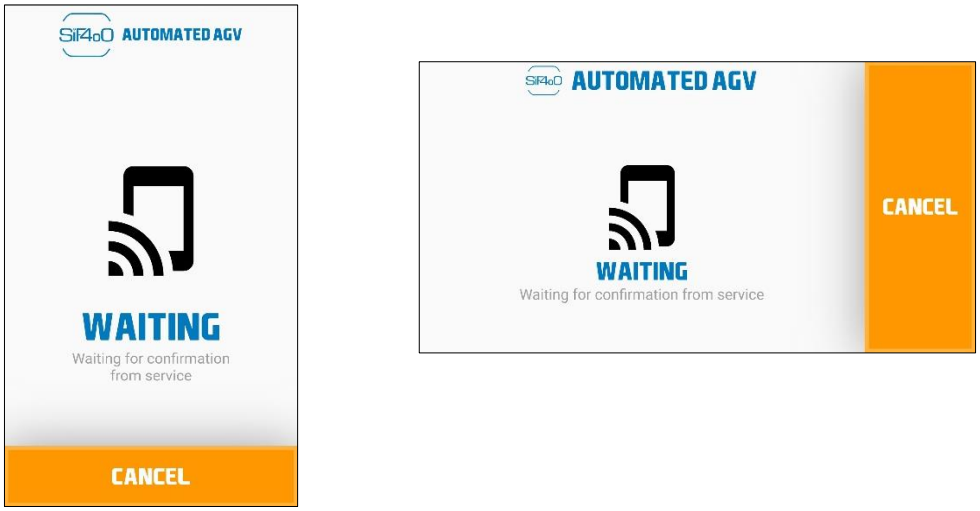

Figure 20 Waiting view of mobile application: portrait (left) and landscape (right)

The emergency view is displayed when the emergency stop is activated. The release AGV action button enables the user to release the AGV, or more generally to deactivate the emergency stop. This view is shown in figure 21.

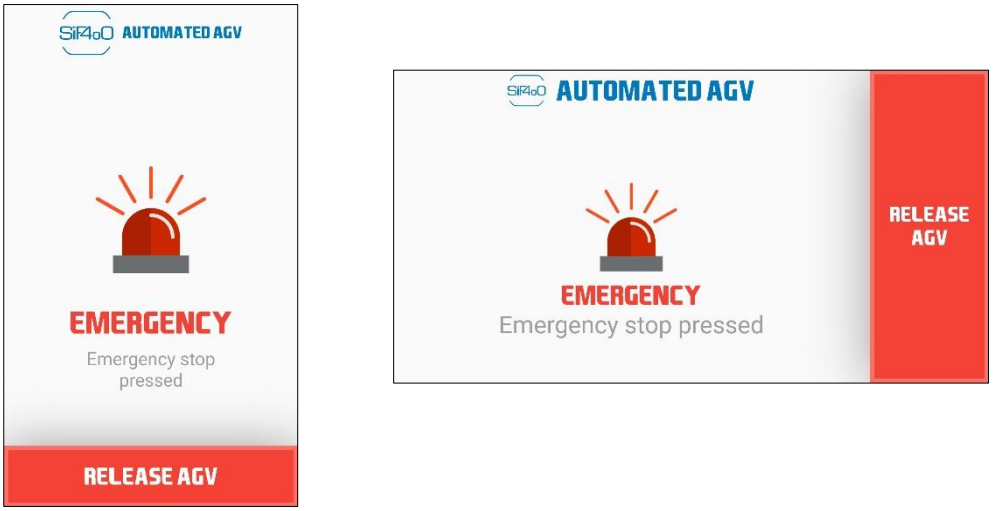

Figure 21 Emergency view of mobile application: portrait (left) and landscape (right)

As already mentioned, the body of a transfer view is more elaborated. The foreground scheme includes the source module at the left-hand side and the destination module at the right-hand side. In between, an image of the item to be transferred is provided. In the background, an arrow graphic interchange format (GIF) shows the transfer direction.

This view is associated with the transfer state. However, based on the current transfer message held in the *MQTTHandler*, the state represents either a transfer request or the actual transfer. Hence, a subtitle, at the top of the body provides the proper information to the user. Figure 22 and 23 show respectively the transfer view associated with a request and the transfer view associated with an actual transfer.

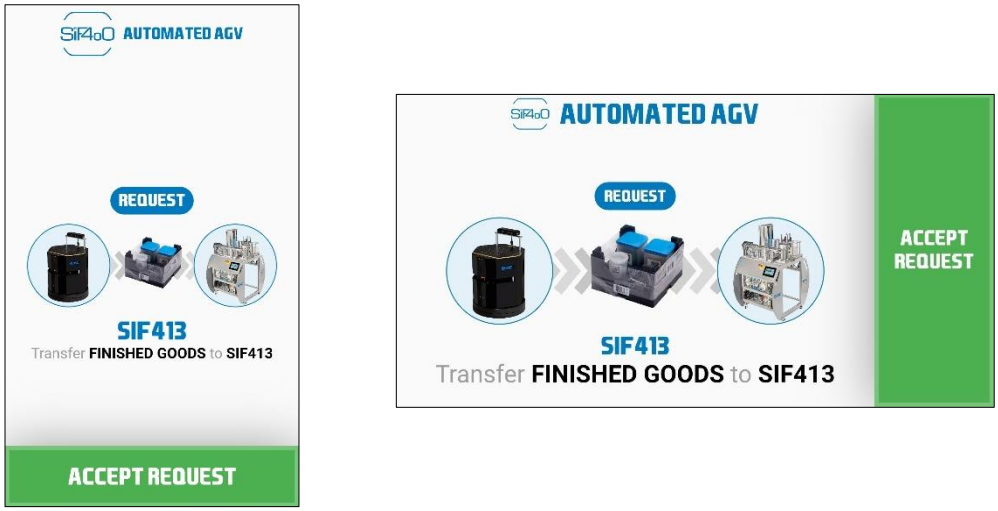

Figure 22 Transfer (request) view of mobile application: portrait (left) and landscape (right)

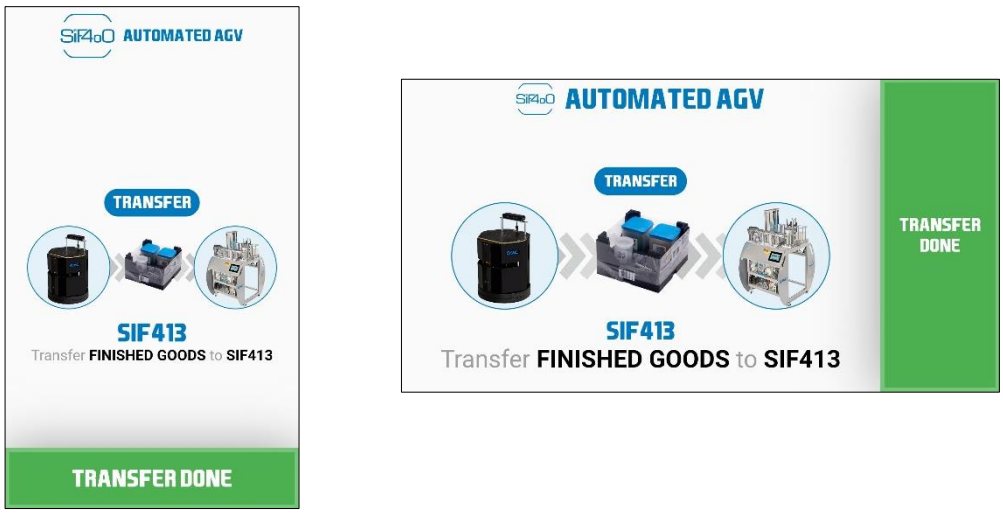

Figure 23 Transfer (transfer) view of mobile application: portrait (left) and landscape (right)

The keywords in the description are highlighted so that the transfer is easier to interpret. In addition, the user is able to zoom in on the body, in order to better identify the item to transfer. The action bar includes, depending on the actual state, an accept request action button or a transfer done action button.

The design and integration of landscape versions for each view was achieved because of the possible use of the mobile application on tablets. The landscape versions do not differ much from the portrait ones, although the action bar is integrated at the right-hand side of the screen in each landscape version.

Figure 24 shows a QR code which redirects to a video showing the animated transfer screen as well as the zooming in action. In addition, the uniform resource locator (URL) is given.

shorturl.at/jrS03

Figure 24 QR code redirecting to a video showing the animated transfer screen

5 Results

5.1 Project results

This Master's thesis results in the integration of an AGV in a SIF-400 production system. This integration was achieved by the development of both a centralized control service, responsible for the management of the communication and the coordination of the resources, and a mobile application which forms the UI of the control service. Both parts are discussed in detail in chapter *4 Implementation*. Considering the main requirement, defined as the prohibition of modifying the existing resources, the required control system was integrated as standalone software modules. These are executed in parallel with the existing automation system of the SIF-400 and can smoothly communicate with it by means of appropriately selected communication channels and protocols. Besides, the requirement forced the implementation of a centralized control service, instead of defining the AGV as an independent agent contained in an agent-based system, which is currently presented as the state of the art. Moreover, such an agent-based approach could have enabled the further evolution towards a multi-AGV system. However, future work is further discussed in section *6.1 Future work*.

Furthermore, both software applications contain a configuration file. Currently, the configuration is limited to on the one hand, the credentials for the required connections, including the IPs and port numbers, and on the other hand, the station priorities regarding the AGV servicing. However, those files can be expanded to enable a higher configurability and as such meet everyone's needs.

Besides, in order to run independently from the other programs, the control service was installed on the server within a Docker container. Hence, the Python interpreter and the other required libraries were not installed on the server itself. Moreover, the Docker container ensures hardware and operating system independence and can therefore be easily shared and thus installed on other servers.

5.2 Project documentation

Finally, to ensure a smooth handover, a manual was written to guide those concerned to manage and control the automated recuperation and stock refilling extension. Unfortunately, in accordance with SMC International Training, it was decided to not make this manual publicly available. However, the following paragraph briefly describes the content of it.

Firstly, the manual contains a chapter devoted to the AGV, whereby the addition, modification or removal of both an entire map or a single POI is explained. Secondly, a chapter discusses the start-up procedure and the debugging of the control service. Hereby, the focus lies onto the specific steps one has to follow in order to properly start up the service. Furthermore, the debugging section describes the logging feature that was developed. To end, the last chapter concerns the mobile signalling application. A first section explains step by step the installation of the Android version of the mobile application. The two following sections are devoted to respectively the connection and the configuration of the mobile application. Finally, the global use, more specifically the steps to successfully perform a transfer, is provided to the readers.

5.3 Project validation

To ensure the correct functioning of the resulting implementation, multiple tests were performed. During those evaluations, the movement of the AGV to the correct stations, the execution of the handshake with the operators and the reactive operation of the other implemented features, for example the emergency stop, were checked. In addition, the docking and undocking processes were also evaluated.

Initially, the stock level of each individual station was manually set to a critical value. Hence, the execution of a single task was analysed. Furthermore, the multiple stations were simultaneously set to a critical stock level. Hereby, the focus of the test lied on the correct functioning of the developed sorting algorithm. This type of test was repeated several times, whereby the station priorities, configured by means of the configuration file, were modified. Moreover, multiple operators were connected to the control service. Hereby, a close attention was paid to the performed handshake, and consequently to the displayed views. To end, the same tests were completed whereby the SIF-400 production system was put in integrated mode. By means of this mode, the simulation of a real production line is achieved.

In addition, several tests were performed on the automatic unloading of the AGV at station SIF-408 [28], which was realised by Wouter Van Rompaey. The focus of those tests lied on the direct communication by means of OPC UA as well as on the implemented handshake.

More generally, about 70% of these tests were initially successfully achieved. The encountered bugs in the remaining tests were fixed. Those bugs concerned on the one hand the operator handshake, whereby an infinite loop appeared, and on the other hand the direct communication with the SIF-408, which failed when operating in integrated mode.

Figures 25, 26 and 27 show the QR-codes redirecting to respectively a video of the automated recuperation of end products at the SIF-412, a video of the automated stock refilling of the SIF-404 and a video of the implemented emergency stop in action. In addition, the URLs are given.

shorturl.at/lnJPQ

Figure 25 QR code redirecting to a video showing the automated recuperation at the SIF-412

shorturl.at/mtMNP

Figure 26 QR code redirecting to a video showing the automated refilling at the SIF-404

shorturl.at/dAMR4

Figure 27 QR code redirecting to a video showing the implemented emergency stop

5.4 Evaluation of productivity improvements

This Master's thesis was suggested to enhance the productivity of an operator and SIF-400. This section describes the measurements and calculations that were performed to make a quantified comparison between the original way of working and the integration of an AGV in the SIF-400 environment.

5.4.1 Number of movements

The main difference lies in the number of movements an operator has to perform. The assumption was made that the operator spends most of his time at the station where he needs to perform most manual operations. This is the case for station SIF-413, where the operator needs to manually disassemble the end products and convert them back into the raw materials. Therefore, all requested service actions requiring attention from the operator will initiate movements that begin and end at the SIF-413[1].

A Recuperation

An operator must perform the following movements in the original end product recuperation method:

1. the operator sees the red lights of the SIF-412 or the operator checks the SIFMES and moves from the recuperation station SIF-413 towards the SIF-412 to empty the full dock;
2. the operator returns to the SIF-413 to recuperate the end products.

An operator must perform the following movements to recuperate end products with the integration of the AGV:

1. the operator receives a notification from the mobile signalling application and moves from SIF-413 to the SIF-412 to place the end products on the AGV;
2. the operator gets a new notification from the mobile signalling application as soon as the AGV has arrived at the SIF-413 and thus moves to the SIF-413 to retrieve the end products from the AGV.

The number of movements remains the same which makes it harder to make a quantified comparison between the original recuperation of end products and the recuperation with the integration of the AGV. Nonetheless, the implementation of the AGV for recuperation of end products has advantages. The biggest advantage of the implementing the AGV for the recuperation of end products is in labour efficiency. The operator does not have to carry the product around, which lightens his workload. Moreover, it does not have to continuously check the stock levels of each SIF-400 station.

[1] This is an assumption to be able to make a first estimate of the benefits of the project. In practice the operator might be combining some service requests which might arrive closely in time. Moreover, in case of a deterministic or well predictable ordering pattern, the work organisation could also be arranged cyclically, neither requiring the operator to return to the recuperation station. These work organisations have not been common practice in the operation of the SIF-400 for the moment and are considered out of scope.

B Stock refilling

An operator must perform the following movements in the original way of stock refilling:

1. the operator checks the SIFMES and moves from the recuperation station towards the corresponding station to check which component is out of stock;
2. the operator moves to the SIF-413 to collect the necessary components;
3. the operator moves back to the station to refill the stock;
4. the operator returns to the SIF-413 to restart its disassembly job;

An operator must perform the following movements to perform stock refilling with the integration of the AGV:

1. the operator receives a notification from the mobile signalling application with the component and station which is out of stock, then collects the necessary components and moves to the station;
2. the operator refills the stock and moves back to the SIF-413.

The stock refilling with the integration of the AGV reduces the number of movements from four to two. The occupancy rate of each station was calculated to quantify this difference.

5.4.2 Measurements and calculations

This section explains how the occupancy rate was measured and calculated.

A Cycle time

The cycle time of each production station was measured to determine the bottleneck time in the production line. The cycle times of the SIF-402, the solid material filler, are distinguished per product type. Likewise, the cycle times of the SIF-408, the product merging station, are also distinguished per number of combined products. The results of this measurement can be found in appendix A.

From the recordings of the cycle times as described above, it is clear that the SIF-409 creates the bottleneck time when executing a make to order (MTO), defined as an order of a pack with one single container, where the end product is dispatched from the SIF-412. When executing an MTO of multiple products, the SIF-402 or SIF-408, depending on the ordered products, creates the bottleneck time. On the other hand, when executing a make to stock (MTS), the SIF-402 or SIF-405, depending on the ordered product, creates the bottleneck. An MTS is defined as an order whereby a single product is stored in the SIF-406.

B Inventory turnover time

The maximum number of units produced per inventory item were counted to determine the inventory turnover time per inventory item. The results of this calculation can be found in appendix B. The number of production units of the pebbles of the SIF-402 station depends on the product type and more in particular the quantity used per product type. The required quantities per product type considered are 15 g, 30 g or 45 g for square containers and 5 g, 10 g or 15 g for round containers. The maximum quantity of the inventory for the pebbles in the loaders is 400 g. To calculate the number of production units per product type, the maximum stock quantity was divided by the required quantity per product. The results of this calculation were rounded down,

as there would not be enough stock available to produce a full production unit for the next product.

The following formula was used to calculate the inventory turnover time per inventory item:

$$t_{inventory\ turnover\ for\ inventory\ item\ i} = t_{bottleneck} \cdot Q_{\max stock\ for\ inventory\ item\ i}$$

The inventory turnover time, for MTO of one product, can be found in the first table of appendix C. The inventory turnover time, for MTO of four products, can be found in the second table of appendix C. These inventory turnover times are calculated based on the slowest single coloured product to make: four times 45 g of yellow pebbles.

C Occupancy rate

The travel time to the station was measured to calculate the occupancy rate. The first table of appendix D shows the results of this measurement and the total travel time for all movements, per station. The occupancy rate can be calculated using the following formula:

$$Operator\ occupancy\ rate_{inventory\ item\ i} = \frac{t_{service\ time\ operator\ for\ inventory\ item\ i}}{t_{inventory\ turnover\ for\ inventory\ item\ i}}$$

$$AGV\ occupancy\ rate_{inventory\ item\ i} = \frac{t_{service\ time\ AGV\ for\ inventory\ item\ i}}{t_{inventory\ turnover\ for\ inventory\ item\ i}}$$

The service time for the operator and AGV comprises on the one hand, the time to move to and from the station and, on the other hand, the time to replenish the inventory, since both are assumed to remain on the location while performing the inventory replenishment operation. As a first approximation the time to replenish the inventory is assumed neglectable compared to the total travel time. This will give an exaggeration of the estimated benefits.

The occupancy rates per inventory item, based on above assumptions, for MTO of one product, can be found in the second table of appendix D.

The occupancy rates per inventory item, based on above assumptions, for MTO of four products, can be found in the third table of appendix D. These occupancy rates are calculated based on the slowest single coloured product to make: four times 45 g of yellow pebbles.

5.4.3 Observations

Two production scenarios were chosen to calculate the occupancy rates of an operator for the replenishment of raw materials with and without the implementation of the AGV.

The first scenario is the continuous production of an MTO of one round container with 5 g of blue pebbles. The second scenario is the continuous production of an MTO of four square containers with 45 g of yellow pebbles.

The overall occupancy rate of an operator is calculated by following formula:

$$Operator\ occupancy\ rate_{overall} = \sum_{all\ inventory\ items\ i} Operator\ occupancy\ rate_{inventory\ item\ i}$$

Therefore:

$$\begin{aligned} Operator\ occupancy\ rate_{overall} \\ = occupancy\ rate_{blue\ pallets} + occupancy\ rate_{containers} \\ + occupancy\ rate_{pebbles} + occupancy\ rate_{caps} + occupancy\ rate_{packs} \\ + occupancy\ rate_{white\ pallets} \end{aligned}$$

Table 5 shows the result of the occupancy rates of the operator for the two described scenarios, with and without the implementation of the AGV.

Table 5 Comparison between the occupancy rates

	Occupancy rate operator (%)	Occupancy rate AGV (%)
1 round container 5 g blue pebbles	8.91	4.46
4 square containers 45 g yellow pebbles	15.3	7.63

With the implementation of the AGV, an operator's occupancy rate for refilling raw materials is reduced by 50%, which he or she can devote to other tasks. This means an operator has more time left for handling of additional tasks.

Although these savings in occupation rate seem quite small, one should be aware that in real life production plants:

- replenishment operations are next to handling production interruptions, for instance quality issues, one of the major sources of operator occupation. The operators are assigned a number of production lines so that their overall occupation is nearly 100%[2]. In this case the absolute savings from the implementation of the AGV can be quite large, as the operator could run a second line or alternatively save the plant one operator;
- manipulation of raw materials requires handing these goods by forklifts from a warehouse, which may not be as close as the SIF-413 recuperation station in the SIF-400. Therefore, in real life situations, the total travel time can really make a very significant contribution to the overall occupancy rate of the operators and the previously made assumption of neglecting the replenishment time can be a quite good approximation.

[2] Taking into account all tasks including the necessary time for personal care etc.

Returning to the context of the SIF-400 operation, the above considerations show that, the operator is far from being fully burdened by the replenishment operations alone. On the other hand, the main manual disassembly tasks are situated at the recuperation station, as mentioned earlier. Therefore, it is instructive to see what the occupation rate from this disassembly operation will be.

The manual disassembly of end-products by the operator takes an average of 87 seconds, divided by the cycle time of the bottleneck process step, provides a first estimate of the operator occupancy related to disassembly. Since the disassembly takes about 87 seconds and the supply rate is about one every 96 seconds, the resulting occupation rate for this task is about 90%, which is considered to be quite high especially in combination with the occupation rate for replenishment. Therefore, operating the line with only one operator will be challenging without task optimisation.

Next to this, the occupancy of the pebble colour sorting function available at the SIF-413 was checked. The time to sort one gram of coloured pebbles in the SIF-413 is on average 24 seconds. While on average there will arrive one pack with one container of five grams product at least every cycle time as determined by the bottleneck process step. In this case approximately one every 96 seconds. This results in a task of 120 seconds for each pack delivered every 96 seconds and therefore results in an over occupation of the sorting station.

6 Conclusion

This Master's thesis results in the integration of an AGV in the SIF-400 production system. This integration currently provides an efficient transport and operator signalling solution for the recuperation of end products and the replenishment of raw materials.

It was also demonstrated that with this implementation, a significant efficiency gain of 50% could be obtained in relation to the operations mentioned above, as opposed to the manual transportation principle that had been used previously. However, as the integration was only made on the SIF-400, which is a scaled version of a real scale production system, the recorded travel times were relatively small and represent only a relatively small part of the overall occupancy rate of the operator. Therefore, in this context, they did not result in large absolute savings. They are expected to have a significantly larger impact in a real-life production system, as the travel times and occupancy rates associated with this type of operations are significantly higher in this context.

The integration of the AGV, in addition to the increased efficiency, expands the capabilities of the SIF-400 system as an inspiring and didactic demonstrator. This an important aspect since the SIF-400 is primarily a simulation and training system for Industry 4.0. The extension of automated recuperation and stock refilling with an AGV provides new ways to inspire and train the key concepts of Industry 4.0.

This implementation will be provided to stakeholders as a complete modular extension of the current SIF-400 didactic demonstrator. The code of all developed software will be handed over as well as a complete documentation wherein the use of each module is thoroughly discussed. Hence, this extension can easily be deployed on different SIF-400 systems.

Besides the transport solution, the implementation could be expanded to further meet the required needs. However, due to the project requirements, a centralized architecture, as opposed to the state-of-the-art agent based one, was preferred and implemented accordingly. Therefore, it might be more interesting to convert the current architecture into an agent-based system, in order to facilitate further expansions of the SIF-400. Some of the possible expansions are described in the section *6.1 Future work*.

6.1 Future work

The current integration is expandable in different ways. Concerning this, the current project structure has been developed with the aim of facilitating further evolution. The sections below describe possible upgrades or innovative implementations regarding the project.

6.1.1 System architecture

A first possible adjustment would concern the adopted system architecture. As limited by the project requirements, a centralized system architecture was preferred for this project. However, according to the current state of the art, a decentralized approach, integrating the AGV as an agent into an agent-based system, might result in a more convenient and up-to-date solution.

Furthermore, this approach would enable the implementation of a multi-AGV system. Since the occupancy rates for the refilling of raw materials is relatively low for the SIF-400, a multi-AGV system might not be necessary from that point of view. However, a multi-AGV system may contribute to the inspiration and demonstration purposes of the SIF-400.

6.1.2 Control system algorithms for prioritizing and grouping of service requests

In terms of software, the optimisation of the sorting algorithm in function of the prioritization of the service requests gathered from the different stations might be an interesting option. This can possibly be done in following ways:

- by determining the priorities based on the current and future MTO and MTS orders instead of directly using the first in, first out (FIFO) calls dedicated by the minimum inventory level sensors in combination with the priorities configured in the configuration file;
- by determining the possible service request combinations based on the actual layout of the product holder installed on top of the AGV and the occupied or still available positions;
- by determining the priorities based on a developed machine learning (ML) algorithm which would learn by means of the current critical stock levels and the required timing for the execution of the task.

6.1.3 Mobile signalling application

Currently, only the necessary transfer actions and the AGV's battery level are monitored by means of the mobile signalling application. Hence, further control and supervision options could be integrated as part of the UI. For example, the expansion of the displayed telemetric data can form a first improvement. Besides, an overview of the connected operators, in combination with the task they are executing, if a multi-AGV system is adopted, can form an interesting feature to add.

Bibliography

[1] SMC International Training, “SIF-400 - Photo gallery,” SMC International Training, 2021. [Online]. Available: https://www.smctraining.com/en/webpage/indexpage/1210. [Accessed 15 05 2021].

[2] *SIFMES Software Manual,* 2021.

[3] T. Foote, “The TurtleBot 2e”.

[4] J. Lentin, Robot Operating System (ROS) for Absolute Beginners, Apress, 2018.

[5] D. Singh, E. Trivedi, Y. Sharma and V. Niranjan, “TurtleBot: Design and Hardware Component,” *International Conference on Computing, Power and Communication Technologies (GUCON),* no. Sep 28-29, 2018, pp. 805-809, 2018.

[6] ROS community, “gmapping,” [Online]. Available: http://wiki.ros.org/gmapping. [Accessed 2021].

[7] J. Friedrich, S. Scheifele, A. Wilhelm Verl and Lechler Armin, “Flexible and Modular Control and Manufacturing System,” *Procedia CIRP,* vol. 33, pp. 115-120, 2015.

[8] T. J. S. Belden, “The road to plug-and-produce,” EngineerIT, 18 06 2017. [Online]. Available: https://www.ee.co.za/article/the-road-to-plug-and-produce.html. [Accessed 15 05 2021].

[9] C.-S. Karavas, G. Kyriakarakos, K. G. Arvanitis and G. Papadakis, “A multi-agent decentralized energy management system based on distributed intelligence for the design and control of autonomous polygeneration microgrids,” *Energy Conversion and Management,* vol. 103, pp. 166-179, 2015.

[10] W. Mahnke and S.-H. Leitner, “OPC Unified Architecture,” *ABB Review,* no. 3/2009, pp. 56-61, 2009.

[11] S. Cavalieri and F. Chiacchio, “Analysis of OPC UA performances,” *Computer Standards & Interfaces,* no. 36, pp. 165-177, 2013.

[12] M. Zeiss, “List of Open Source OPC UA Implementations,” 10 March 2020. [Online]. Available: https://github.com/open62541/open62541/wiki/List-of-Open-Source-OPC-UA-Implementations.

[13] ROS community, “rosbridge_server,” 4 May 2013. [Online]. Available: http://wiki.ros.org/rosbridge_server.

[14] C. Crick, G. Jay, S. Osentosiki, B. Pitzer and O. C. Jenkins, “Rosbridge: ROS for Non-ROS Users,” pp. 1-12.

[15] destogl, “FreeOpcUa/freeopcua,” [Online]. Available: https://github.com/FreeOpcUa/freeopcua.

[16] MQTT, “MQTT: The Standard for IoT Messaging,” [Online]. Available: https://mqtt.org.

[17] C. Bormann, “CoAP,” [Online]. Available: https://coap.technology.

[18] VMware, “AMQP 0-9-1 Model Explained,” RabbitMQ, [Online]. Available: https://www.rabbitmq.com/tutorials/amqp-concepts.html.

[19] “What is DDS?,” DDS Foundation, [Online]. Available: https://www.dds-foundation.org/what-is-dds-3.

[20] T. Yokotani and Y. Sasaki, “Comparison with HTTP and MQTT on Required Network Resources for IoT,” *International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC),* 2016.

[21] N. Naik, “Choice of Effective Messaging Protocols for IoT,” 2017.

[22] mvollrath, “RobotWebTools/rosbridge_suite,” 8 December 2020. [Online]. Available: https://github.com/RobotWebTools/rosbridge_suite.

[23] kyberszittya, “horverno/rosbridge-csharp-connection,” 14 September 2015. [Online]. Available: https://github.com/horverno/rosbridge-csharp-connection.

[24] W. Rowe and J. Johnson, “Python vs Java: What’s The Difference?,” 25 November 2020. [Online]. Available: https://www.bmc.com/blogs/python-vs-java/.

[25] D. Karczewski , “Flutter vs Kotlin: Which Technology Will Be Better For Building Your Mobile App?,” Ideamotive, 25 01 2021. [Online]. Available: https://www.ideamotive.co/blog/flutter-vs-kotlin-which-technology-will-be-better-for-building-your-mobile-app. [Accessed 15 05 2021].

[26] SMC International Training, “SIF-400 - The training system for Industry 4.0,” SMC International Training, [Online]. Available: https://www.smctraining.com/en/webpage/indexpage/1208.

[27] Eclipse Foundation, “Eclipse Mosquitto,” Eclipse Foundation, [Online]. Available: https://mosquitto.org.

[28] *Automated resupply management in existing production line with a cobot and an automated guided vehicle,* 2021.

[29] T. Dalling, “Model View Controller Explained,” 31 05 2009. [Online]. Available: https://www.tomdalling.com/blog/software-design/model-view-controller-explained/. [Accessed 15 05 2021].

[30] A. Gupta, “An insight into Model, View, Controller (MVC) in the context of SAP UI5,” 02 01 2016. [Online]. Available: https://blogs.sap.com/2016/01/02/an-insight-into-model-view-controller-mvc-in-the-context-of-sap-ui5/. [Accessed 15 05 2021].

Appendix

Appendix A: Cycle time per station

	Product type		t_{cycle} (s)
SIF-401			46.07
SIF-402	Square container	15 g blue pebbles	49.11
		30 g blue pebbles	50.47
		45 g blue pebbles	51.25
		15 g red pebbles	49.18
		30 g red pebbles	50.36
		45 g red pebbles	50.83
		15 g yellow pebbles	55.41
		30 g yellow pebbles	56.9
		45 g yellow pebbles	57.51
	Round container	5 g blue pebbles	48.67
		10 g blue pebbles	49.88
		15 g blue pebbles	49.59
		5 g red pebbles	48.83
		10 g red pebbles	49.57
		15 g red pebbles	49.79
		5 g yellow pebbles	55.02
		10 g yellow pebbles	55.19
		15 g yellow pebbles	55.56
SIF-404	Product from left belt		38,37
	Product from right belt		38,05
SIF-405			39.93
SIF-407			45.97
SIF-408	1 product		96.27
	2 products		138.2
	3 products		180.81
	4 products		222.77
SIF-410			116.89
SIF-411			17.65
SIF-412			9.96

Appendix B: Inventory turnover time per inventory type expressed in production units per product type

	Inventory type	Product type	$Q_{max\ stock}$
SIF-401	Blue pallets		22
	Square containers		10
	Round containers		14
SIF-402	Square container	15 g pebbles	26
		30 g pebbles	13
		45 g pebbles	8
	Round container	5 g pebbles	80
		10 g pebbles	40
		15 pebbles	26
SIF-404	Customized square products		8
	Customized round products		12
SIF-405	Square caps		21
	Round caps		21
SIF-408	Packs		5
SIF-410	White pallets		11

Appendix C: Inventory turnover time per product type

Inventory turnover time per product type for MTO of a pack comprising one single container

	Component	$t_{turnover}$ (s)
SIF-401	Blue pallets	2572
	Square containers	1169
	Round containers	1636
SIF-402	15 g pebbles	3039
	30 g pebbles	1520
	45 g pebbles	935.1
	5 g pebbles	9351
	10 g pebbles	4676
	15 g pebbles	3039
SIF-404	Customized square products	935.1
	Customized round products	1403
SIF-405	Square/round caps	2455
SIF-408	Packs	584.5
SIF-410	White pallets	1286

Inventory turnover time per product type for MTO of a pack comprising four containers

	Component	$t_{turnover}$ (s)
SIF-401	Blue pallets	1265
	Square containers	575.1
	Round containers	805.1
SIF-402	15 g pebbles	1495
	30 g pebbles	747.6
	45 g pebbles	460.1
	5 g pebbles	4601
	10 g pebbles	2300
	15 g pebbles	1495
SIF-404	Customized square products	460.1
	Customized round products	690.1
SIF-405	Square/round caps	1207
SIF-408	Packs	1150
SIF-410	White pallets	2530

Appendix D: Occupancy rates

Travel time per station

	t_{travel} (s)	# movements operator	$t_{total\ travel}$ operator (s)	# movements AGV	$t_{total\ travel}$ AGV (s)
SIF-401	6.83	4	27.3	2	13.7
SIF-402	5.43	4	21.7	2	10.9
SIF-404	1.87	4	7.48	2	3.74
SIF-405	2.42	4	9.68	2	4.84
SIF-408	4.84	4	19.4	2	9.68
SIF-410	7.22	4	28.9	2	14.4

Occupancy rate per inventory for MTO of a pack comprising one single container

	Component	Occupancy rate operator (%)	Occupancy rate AGV (%)
SIF-401	Blue pallets	1.06	0.531
	Square containers	2.34	1.17
	Round containers	1.67	0.835
SIF-402	15 g pebbles	0.715	0.357
	30 g pebbles	1.43	0.715
	45 g pebbles	2.32	1.16
	5 g pebbles	0.232	0.116
	10 g pebbles	0.465	0.232
	15 g pebbles	0.715	0.357
SIF-404	Customized square	0.799	0.400
	Customized round	0.533	0.267
SIF-405	Square/round caps	0.394	0.197
SIF-408	Packs	3.31	1.66
SIF-410	White pallets	2.25	1.12

Occupancy rate of the operator per inventory for MTO of a pack comprising four containers

	Component	Occupancy rate operator (%)	Occupancy rate AGV (%)
SIF-401	Blue pallets	2.16	1.08
	Square containers	4.75	2.38
	Round containers	3.39	1.70
SIF-402	15 g pebbles	1.45	0.726
	30 g pebbles	2.91	1.45
	45 g pebbles	4.72	2.36
	5 g pebbles	0.472	0.236
	10 g pebbles	0.944	0.472
	15 g pebbles	1.45	0.726
SIF-404	Customized square	1.63	0.813
	Customized round	1.08	0.542
SIF-405	Square/round caps	0.802	0.401
SIF-408	Packs	1.68	0.842
SIF-410	White pallets	1.14	0.571