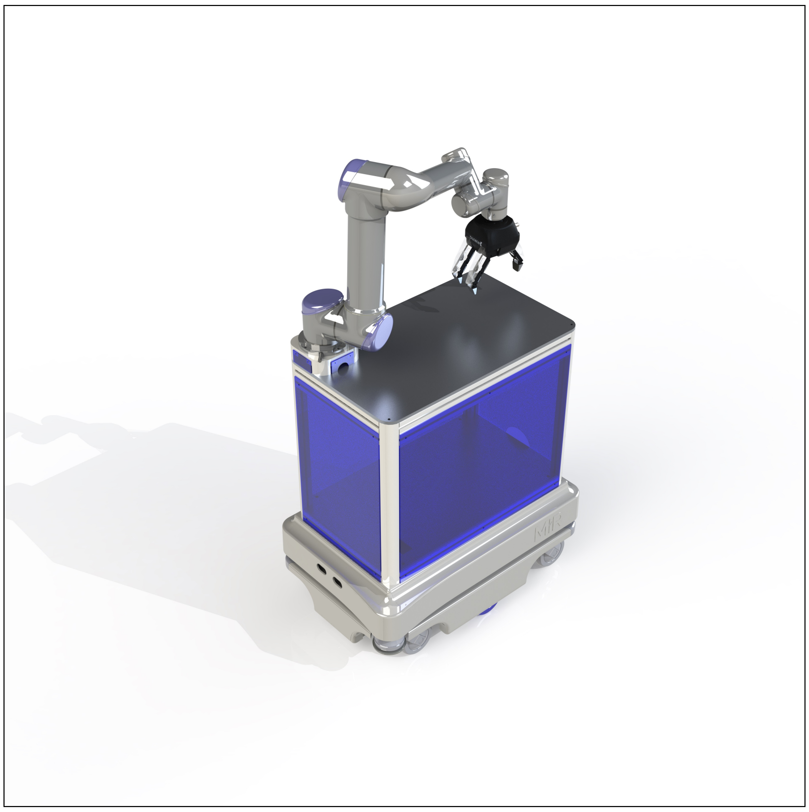
Autonomous Industrial Mobile Manipulator in Smart Production

- Robot Integration -



P5 Project Report

16GR561

Aalborg University
Robotics

AALBORG UNIVERSITY
STUDENT REPORT

Robotics
Aalborg University
http://www.aau.dk

Title:
Autonomous Industrial Mobile Manipulator in Smart Production

Theme:
Robot Integration

Project Period:
5th Semester Fall 2016

Project Group:
16gr561

Participant(s):
Biranavan Pulendralingam
David Cerny
Emil Blixt Hansen
Rasmus Andersen
Steffen Madsen

Supervisor(s):
Simon Bøgh
Dimitris Chrysostomou

Copies: 3

Page Numbers: 84

Date of Completion:
January 20, 2017

Abstract:

The motivation for this project is to bring flexibility and automation to part feeding in an Industry 4.0 context. This is achieved through a development of a new version of the Little Helper (LH6), which is a series of Autonomous Industrial Mobile Manipulators (AIMM), developed by Aalborg University. LH6 consists of a MiR100 as its mobile platform and a UR5 as its collaborative manipulator. In order to integrate these two standalone robots, an aluminium frame is created and a ROS-based program, named Skill-Based System (SBS), is used and expanded to support mobile robots. A specific part feeding challenge is chosen in an Industry 4.0 production line demo, assembling mobile phone dummies, which concerns a tray, containing 12 PCB's for smartphones. LH6 would get a signal from a PLC, controlling the tray conveyor, and then drive into position. LH6 then takes out the empty tray and puts in a full one afterwards. Certain difficulties were encountered regarding MiR100's navigation system and a bug in a TCP server, which combined brought down the success rate to approx. 64%. However, with the bug solved, the solution shows potential in an Industry 4.0 integration.

Contents

Preface

This report documents a project made by 5th semester students of Robotics B.Sc. at Aalborg University. The project was carried out during the time from September to December 2016. The complexity of technical and programming language is on a level, that is expected of students on this education program to use and understand. The topic of this project is integration of an autonomous mobile manipulator in a smart production.

A special thanks to Simon Bøgh and Dimitris Chrysostomou, the project supervisors, who helped the authors and encouraged them in the process of writing the report and developing the solution. A thanks also goes to Casper Schou and Rasmus Skovgaard Andersen for a significant assistance with certain programming challenges, rewiring of an assembly module or robots, and design of an electric circuit.

Reader's Guide

This report uses the modified Chicago style of referencing. All used sources can be found in the end of the report. Source references are written in the end of paragraphs in brackets and in a specific format: (author's name, year). References to figures are written in the same way, just in the end of figure's captions. If a figure is missing a reference, it means that it has been created by authors themselves.
The terms smart production and Industry 4.0 are used interchangeably throughout the report.
Appendices can be found in the end of the report, containing additional information regarding the project. All additional material can be found in a DropBox folder (`https://www.dropbox.com/sh/easiqhiqohkqfq7/AABjUeWcLpskfj-iLzxyMaX-a?dl=0` or in Figure 1) which contains:

- SBS and ROS drivers.
- CodeSys .project file with used PLC program.
- Final test video footage.

Figure 1: Link to project repository.

Biranavan Pulendralingam
<bpulen14@student.aau.dk>

David Cerny
<dcerny14@student.aau.dk>

Emil Blixt Hansen
<ebha14@student.aau.dk>

Rasmus Andersen
<rean14@student.aau.dk>

Steffen Madsen
<smad14@student.aau.dk>

Chapter 1

Introduction

The purpose of this project is to create an automated solution for part feeding challenges in an Industry 4.0 context. This is achieved by designing a version of the series of autonomous industrial mobile manipulators (AIMM) called Little Helpers and integrating it into Aalborg University's Industry 4.0 demo (referred to as the FESTO Cyper-Physical Factory, FESTO CP Factory). This chapter consists of an introduction to the background of the project, a short description of the Little Helper robots, Industry 4.0, FESTO CP Factory and the initial problem statement.

1.1 Project Background

Today many manufacturing companies are experiencing an unpredictable and dynamic production environment. This is among other, due to the following conditions:

1. Increased competition from low labour countries.
2. Increased customization.
3. Low product life cycles, due to ongoing development of emerging technologies.
4. Uncertain and changing markets.
5. Increased requirements for health and environment regulations.

These conditions are testing the manufacturing companies' ability to efficiently adapt their production systems accordingly, and thereby their ability to stay competitive in a dynamic and globalized market. If a company fails to adapt to the conditions, it will have consequences for their competitiveness, and might be a threat to the survival of the company. (Madsen 2016)

To adapt to the conditions sufficiently, manufacturing companies should design their production systems around flexibility. The traditional hard automation can, because of dedicated specialised production equipment, be efficient, but lacks when it comes to flexibility and product variety. Alternatively, manual labour brings flexibility and product variation to a production, but have a low efficiency and is often not economically viable for high wage countries as Denmark. Therefore, to achieve a greater flexibility together with an efficient automated production, there is a need for a general shift in how to design a production system, from hard automation involving mass production, to a flexible automation (here referred to as soft automation) involving a flexible and reprogrammable automation enabling an ideal batch size of 1. The aim of soft automation and this project is to achieve flexibility as well as efficiency in a production system. (Bøgh et al. 2012) (Kalpakjian and Schmid 2014)

1.2 Little Helper

In a series of autonomous mobile manipulators designed by Aalborg University, consisting of five robots, called Little Helper (LH), Aalborg University gives their idea of an element in a flexible production system (See Figure 1.1). The aim of the LH series is to achieve flexibility and efficiency through a high degree of automation, using an Autonomous Industrial Mobile Manipulator (AIMM). By the use of a mobile platform the LH robots can navigate autonomously in uncertain human environments and can by an integration of a robot manipulator handle a variety of tasks. Furthermore, some previous LH's use a system called Skill Based System (SBS), which achieves fast robot reprogramming for non-robot experts.

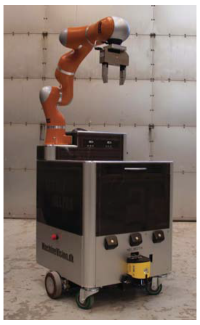


Figure 1.1: Illustrates LH3 consisting of a KUKA LWR manipulator, integrated together with at Neobotix mobile platform. (Carøe, Hvilshøj, and Schou 2012)

1.3 Industry 4.0

The industrial world is in the midst of a fourth industrial revolution. A new digital industrial technology is emerged, called Industry 4.0, which is a combination of new technologies within internet, robotics and machines. The fundamental idea of Industry 4.0 consists of nine technology advances, including autonomous robots, simulation, the industrial internet of things and cloud computing. In Industry 4.0 sensors, robots and IT-systems, are connected by an overall system called a cyper physical system. In this systems the production elements (e.g sensors and robots) can communicate through simple internet protocols. Industry 4.0 achieves to increase flexibility and efficiency in a production system, by gathering and analysing data across machines, sensors, etc. (Rüssmann et al. 2015)

1.3.1 FESTO Cyper-physical Factory

Aalborg University's idea of an Industry 4.0 implementation is a factory demo called Festo CP Factory (see Figure 8.5). The factory is located at Aalborg University, and is referred to as smart production factory. The FESTO CP Factory is built up by modules. Each of these modules is independent, meaning that they can be separated and combined in new ways with minimal effort. The smart production factory has the purpose of showcasing the customisability and flexibility of Industry 4.0, in this case assembly of smart-phone dummies. The phones in the factory will be able to have different customised features, without changing the production system. This is possible as all the sensors, Programmable Logic Controllers (PLC's), conveyors etc. in the factory are connected through a cypher-physical system.

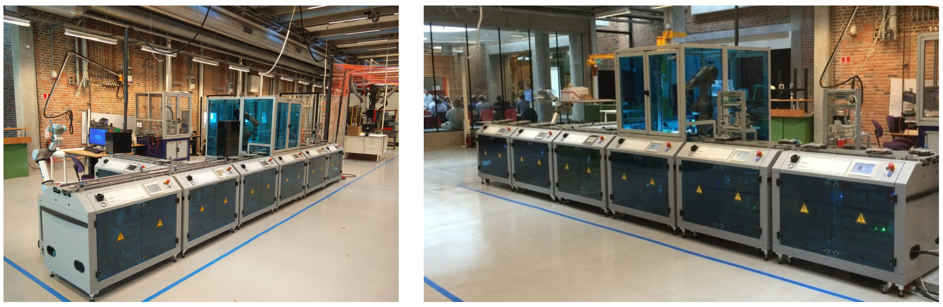


Figure 1.2: FESTO CP Factory at Aalborg University.

The production line consists of eight modules each having two conveyor belts with their own RFID readers. Each conveyor belt is independent and can run in both directions. Carriers with individual RFID tags are moving around on the conveyors. When an order is sent to the system through a Manufacturing Execution System (MES), the given order is connected to its own carrier and RFID tag. By the RFID tag readers the factory is able to read the customised order, and execute the system with respect to the order. In this way it is possible to adapt the production line according to variables chosen by the customer.

1.4 Initial Problem Statement

The focus of this project is to automate part feeding challenges in an Industry 4.0 context. The aim of this project is to create a new and the sixth version of Aalborg University's series of LH robots (LH6) and integrate it as a part of FESTO CP Factory to solve part feeding challenges, and thus adding a new element of flexibility to the FESTO CP Factory.

Chapter 2

Challenges and Integration Proposals

This chapter contains an analysis of context and resources of this project, which yields challenges that need to be considered in proposing a solution and setting up requirement specifications (see Chapter 4).

2.1 Part Feeding Challenges

As mentioned in Chapter 1, this project is focused on part feeding challenges in a smart production factory. It is then important to identify what kind of challenges one faces in part feeding tasks in general.

Every kind of production line needs to have a system, where assembly parts, or raw materials enter the line, no matter if it is a smart production, or mass production. This makes part feeding one of the essential processes in a production line.

Part feeding tasks can be carried out either manually, or they can be automated. There may be different types of part feeding challenges; however, they still share common operations such as:

- Move to production line.
- Pick certain part.
- Place part in production line.
- Move away from production line.

As mentioned in Chapter 1, manual part feeding may lack efficiency. However, it cannot be surpassed in flexibility and adaptation, since a human worker can be told what to do, is able to critically think about what he/she is doing, predict and corrects errors. Alternatively, hard automation is usually effective but lacks flexibility, since any alteration of such a production line can be complicated and expensive.

In case of soft automation, flexibility is much higher. The possibility to have variety in the production line, or customise the product is one of the main focuses of soft automation. That is because a soft automated production line is equipped by reprogrammable computer-controlled machines, that are able to produce various parts with high complexity.

A general problem in a production line is the timing of the part feeding. It is not automaticaly known when part feeding is supposed to happen, which is why detection and communication is important to be implemented in the soft automation. It is an aim of future development, to create flexible manufacturing systems with as high productivity as possible. Such systems naturally need a flexible part feeding, which except of humans can be carried out by implementation of e.g. mobile manipulators. (Kalpakjian and Schmid 2014)

2.2 FESTO CP Factory Use Case

In Section 2.1, different part feeding challenges, one can encounter in an automated production line, were described. This section investigates part feeding tasks of the FESTO CP Factory. This is done in order to give an overview of the potential part feeding challenges related to the FESTO CP Factory. There are three different parts

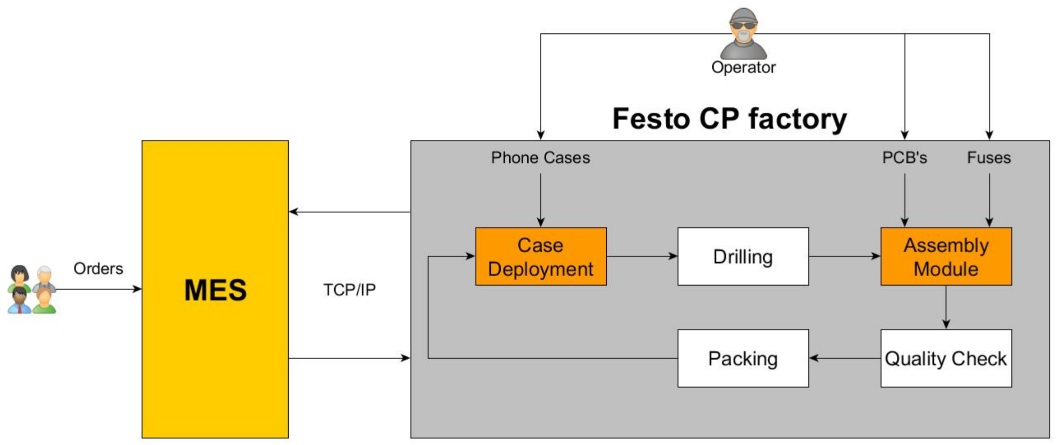
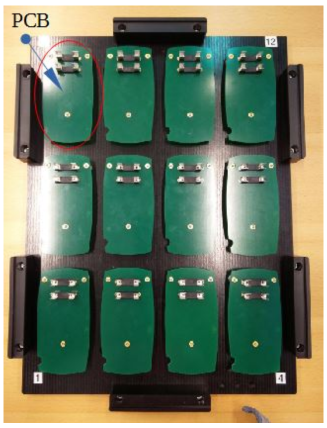
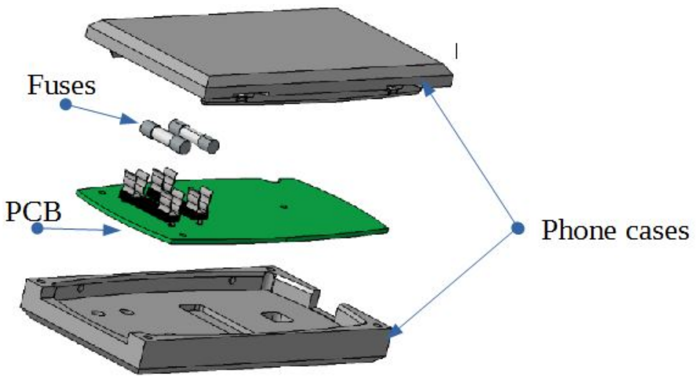


Figure 2.1: Connection of the FESTO CP Factory modules with all the part feeding processes highlighted.

that need to be fed to the modules in the system (see Figure 2.1). When feeding the three parts, there are certain considerations to keep in mind:



(a) The PCB tray.



(b) Exploded view of the phone assembly.

Figure 2.2: The PCB tray and exploded view of the phone, with phone cases, PCB and fuses.

Phone Cases:

The phone cases, as seen in Figure 2.2b, need to be specifically placed in their magazine (see Figure 2.3). Because of backlight vision feedback located in the assembly module (see Figure 2.1), the phone cases can still be processed when they vary in rotation in the horizontal plane, but problems arise when the phone case comes to the magazine flipped. In such a situation the assembly cannot be done successfully, since the system has no way of knowing from the backlight images that the item was flipped.

PCB's:

In case of the PCB tray (see Figure 2.2a), it needs to be inserted into the module in a specific way, where a conveyor then reels the tray inside the module. The orientation is checked by a piece of metal imbued in a part of the tray, which triggers a metal sensor inside the module. If the tray orientation is wrong, the sensor cannot be triggered by the metal piece and the tray is ejected again. Compared with the phone cases and fuses, the PCB's come on a tray, therefore the old tray needs to be removed before inserting a new.

Fuses:

The fuses (see Figure 2.2b) have no specific orientation to be inserted in, but the insertion itself is a task that requires high precision, which can be challenging in case of automated part feeding, since they need to be slid into a small tube that leads inside the module (see Figure 2.3).

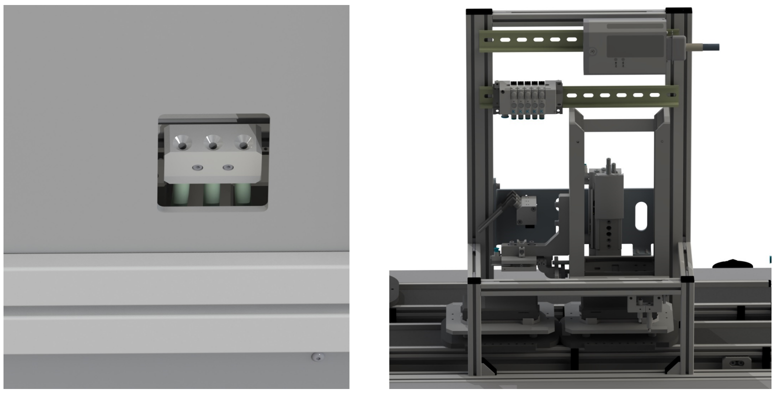


Figure 2.3: The tubes to be fed with the fuses is shown to the left. The magazine for the phone cases is shown to the right.

Initially, the three kinds of parts are fed to the respective modules manually. Having a worker to carry out the task offers a high flexibility, as described in Section 2.1, but may lack efficiency, robustness, and consistency. However, the FESTO CP Factory is a case of soft automation, since all modules are easily reprogrammable via their PLC's and the production line has a variation of product modifications. Moreover, the production line does not output products in consistent rate. The production output is dependent on number of orders placed in the MES, thus high flexibility during part feeding is required.

Automating the manual part feeding task, by using an AIMM, can ideally increase the flexibility together with the efficiency. The following section investigates previous LH's with the purpose of inspiration for a new LH, specifically made for part feeding in the FESTO CP Factory.

2.3 Previous Little Helpers

The LH concept has gone through five iterations as seen in Table 2.1. These five iterations have had different focuses and improvements over their former counterparts. This section investigates the five LH's, with the purpose of reviewing earlier versions and see what can be used for a future version. (Robotics and Group 2016)

In Table 2.1 all of the previous LH's are showcased and compared. It is important to know that these LH's were created with learning in mind, thus the task they were to fulfil is different e.g. LH4 is not meant to drive around autonomously in a factory whereas the LH3 is. Moreover, all previous LH's aimed for a battery life of 7 hours. Given the fact that this was an unchanging requirement in all LH versions, it can be presumed that such a battery lifetime is achievable and desir-

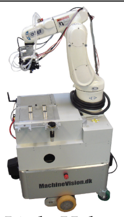
Version	Year	Hardware/Software	Pros(+)/Cons(-)
 Little Helper 1	2008	**Manipulator:** Adept Viper s650 **Platform:** Neobotix MP655L **Software:** Adept Desktop, MATLAB, LabVIEW, VBAI, PlatformCTRL and PltfGUI	+Battery life of 7 hours +Repeatability of ± 0.02mm +Payload of 20kg -Manipulator payload of 2.5kg -Price of 800 000,- DKK -Many different software used
Little Helper 2	2011	**Manipulator:** KUKA LWR **Platform:** Neobotix MP655L **Software:** FRI, TAPAS Project	+Battery life of 7 hours +Repeat ability of ± 0.05mm +Payload of 20kg +Manipulator payload of 7kg -Price around 1 million DKK
Little Helper 3	2012	**Manipulator:** KUKA LWR **Platform:** Neobotix MP655L **Software:** SBS and ROS	+Battery life of 7 hours +Repeat ability of ± 0.05mm +Payload of 20kg +Manipulator payload of 7kg +Few different types of software -Price of 1.14 million DKK
Little Helper 4	2013	**Manipulator:** UR5 **Platform:** None **Software:** SBS and ROS	+Payload of more than 20kg +Manipulator payload of 5kg +Few different types of software +Price of 314 727,- DKK -Repeat ability of ± 0.1mm -No autonomus mobile movement
Little Helper 5	2014	**Manipulator:** UR5 **Platform:** None **Software:** UR Interface	+Payload of more than 20kg +Manipulator payload of 5kg +One software +Price of 250 000,- DKK -Repeat ability of ± 0.1mm -No autonomus mobile movement

Table 2.1: Comparison between the different versions of the LH.

able. Table 2.1 also shows that the payload of the used manipulator first increases from 2.5 kg to 7 kg and drops to 5 kg afterwards. Although all LH's were developed with different task in mind, this can indicate that payload of 2.5 kg is not sufficient for a broad usage.

The KUKA LWR, used for LH2 and LH3 has higher payload and better repeatability than UR5 used for LH4 and LH5, along with built-in torque sensors. However, implementing the KUKA arm drastically increases the price of LH, without adding any significant advantages in a regular part feeding operations, e.g. as described in Section 2.2.

All LH's from 1 to 3 needed a workstation calibration because of their mobile aspect, i.e. imperfections in precision cause their mobile platforms to have an offset when arriving to desired locations, which consequentially impairs precision of the manipulators mounted on them.

SBS was originally developed to be used with a KUKA LWR manipulator, hence its deployment in LH3 implementation. Since then, it has been expanded in order to include UR5 so it can be used with LH4. (Bøgh et al. 2008) (Pedersen 2011) (Carøe, Hvilshøj, and Schou 2012) (Etzerodt, Palmelund-Jensen, and Pedersen 2013)

It can be concluded that LH's can be used for part feeding as long as mentioned aspects are taken into consideration. In Section 2.4 challenges for a new version of LH, dedicated to solving the part feeding cases mentioned in Section 2.2, are described.

2.4 Integration Challenges

This section looks into the challenges of deploying a new version of the LH (LH6) into an Industry 4.0 environment, i.e. the FESTO CP Factory. In order to identify challenges in this section, the challenges from previous LH's mentioned in Section 2.3 are used as an inspiration.

2.4.1 Assembly of LH6 Components

Inspired by the previous versions of LH, the LH6 is to be put together by four parts:

- Mobile platform
- Manipulator
- Tool
- Sensors

These four essentials need to be put together in order to form an AIMM. In case of other LH's, the manipulator is placed in the rear area of a frame, which is placed on top of the platform. Some versions have it mounted in the corner and others in middle of the rear area. No matter which mobile platform is chosen for LH6, a frame is needed to mount the mobile platform and manipulator together. It is also needed that all the hardware components can be stored in a compartment and room for sensors is to be taken into account. Preferably, the height of the LH6 platform should be same as the one of the conveyor belts of FESTO CP Factory

(975 mm), in order to have the dexterous workspace of the manipulator match the workspace of the modules. The manipulator should be placed so that majority of the working space on the platform is available.

LH6 should also be easily reconfigurable in order to comply with the concept of smart production, so it is easy to change the different parts and make it fit a different task. This can be done by e.g. standardizing the different tool mountings of the robot, or make the frame height adjustable.

2.4.2 Battery

As seen in Section 2.3, a big concern for an AIMM is the battery which is needed for it to function. Therefore, the battery capabilities of the product need to be addressed so it can drive for adequate time before recharging.

For the specific part feeding tasks in the given use case (Section 2.2), it is important to have a low downtime in order to keep the flexibility of Industry 4.0. If the concept is to be expanded into performing other tasks like cleaning, assembly and transportation of parts, high battery capacity is needed to keep a high efficiency.

2.4.3 Communication

First part of communication to be investigated, is the communication between the four parts (mobile platform, manipulator, tool and sensors) of LH6. Since the different manufacturers use different programming languages and interfaces the communication is not just plug and play. As seen in previous LH's, a framework called SBS was used as a common interface between those devices.

LH6 also needs to be able to know when the FESTO CP Factory is running low on parts. For this, the LH6 needs to have some sort of communication protocol with the FESTO CP Factory. The PLC's of FESTO CP Factory are all communicating with a computer running MES. This software needs to send a signal to LH6 when a part is running low. This signal could be a simple TCP/IP signal, as illustrated in Figure 2.4.
Without the connection with MES, LH6 will not be able to perform the part feeding tasks efficiently, since proper scheduling cannot be implemented without MES communication.

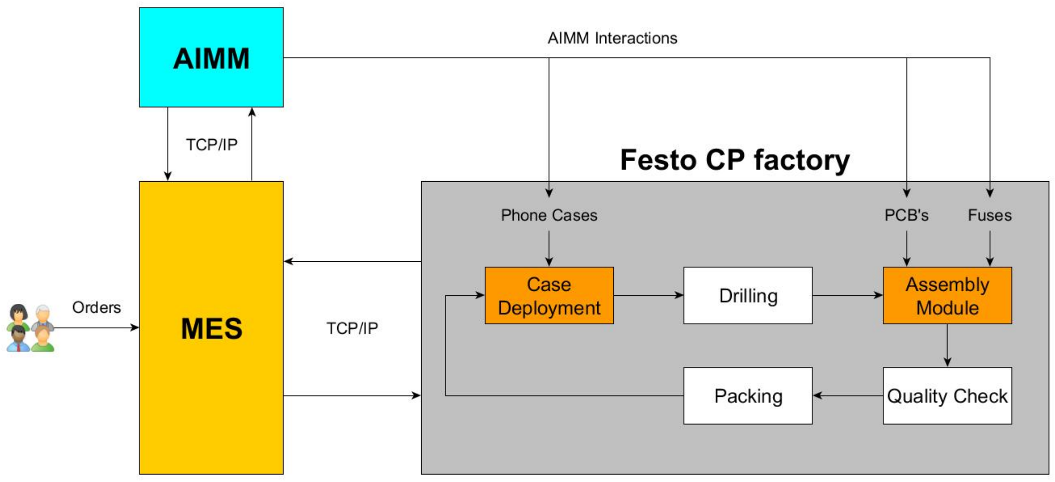


Figure 2.4: Implementation of AIMM part feeder.

2.4.4 Mapping and Navigation

For an AIMM to navigate around in a factory, it needs to know its location and its destination. A method for that is called Simultaneous Localization and Mapping (SLAM). SLAM can be carried out using e.g. laser-scanners, although it can be difficult to see glass and other transparent surfaces. Another sensor technology that can be used for navigating around a factory is ultrasonic sensors, however, they have the weakness of "ghost echos" (noise from reflected waves). Another method that can be used is "dead reckoning", which is a fairly cheap method to implement both computation- and cost-wise, since no additional sensors are required. It works by knowing the diameter and rotations of the wheel and by that it can calculate the distance, speed and orientation. Though it is prone to errors because if the wheel slips on the surface it loses track of the real world and will start thinking it is in another place than it really is. A way to improve on the dead reckoning is to implement a mapping system, using e.g. a laser-scanner. With a laser-scanner it is possible to correct the errors from dead reckoning by scanning the area and combining the information from both. However, this comes with the extra cost of at least one laser-scanner. (Corke 2013)

Furthermore, due to the imprecision of mobile platforms to reach their goal, a calibration method is necessary at all locations where the manipulator is required to interact with surroundings. This can be achieved e.g. by haptic feedback (as in LH2 and LH3), or by using a 3D camera (as in LH1).

2.4.5 Safety

Since the LH6 is an experimental device and not for commercial use, it is not obligatory to follow all the ISO standards. However, some of the ISO standards are still useful as an argument for the design of the LH6 and to fully understand the given task, because of the critical point of view the ISO standards provide. ISO 13850 is focusing on the emergency stop buttons (everything from placement to how the buttons should look), which is relevant for this project, since the manipulator and the mobile platform need to have a synchronized stop button. When the platform is moving around, the button needs to be reachable all around the robot, no matter where the operator is located.

When the manipulator is moving, the mobile platform should be stationary, and by using an alert system (e.g. LED or sound system) LH6 can warn the surroundings about the manipulator moving.

2.4.6 Part Feeding

As mentioned in Section 2.2 there are three different types of part feeding challenges. These three different types of part feeding challenges could have their own dedicated gripper. This is because they are different in shape, size and weight. However, the LH6 is meant to be flexible, thus it is desirable to perform all three tasks with just one gripper.

2.4.7 User-Friendliness

In a production facility there are several operators and not all of them have an engineering degree in robotics, programming etc. Thus an easy intuitive interface with simple commands is wanted. This is done so an employee which has not touched the program before can learn it easily and execute basic commands like pick and place.

One example of such a software framework is SBS, which strives to achieve easy reprogramming of AIMM for non programming experts.

2.5 Skill Based System

The software used for this project is based on a framework called SBS. SBS is a program built for the LH's, which through an intuitive interface enables programming of AIMM's. SBS is built on the open source Robotic Operating System (ROS) and is programmed in C++ through a GUI application development framework from Qt company.

SBS has a hierarchy consisting of different layers. Those layers are decomposed in the following three sections (for an illustration of the layers see Figure 2.5).

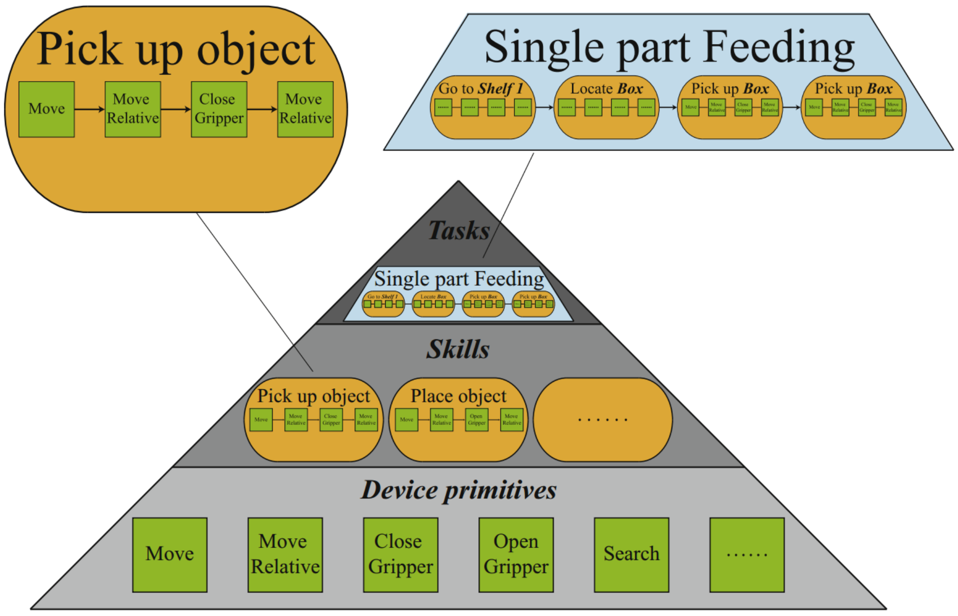


Figure 2.5: The hierarchy of SBS. (Pedersen et al. 2016)

2.5.1 Tasks

The first layer of SBS is tasks. A task is a production-level goal similar to those task a human worker would be instructed to perform, e.g. assemble A, fill feeder B etc. Each task can be decomposed into smaller sequences called skills. A robot is able to perform a task if it is capable of performing all skills that task consists of.

2.5.2 Skills

Skills are the building block of SBS. The amount and types of skills are dependent on what hardware and what types of sensors the robot is equipped with. The more technology implemented the more flexible the system becomes. A skill is e.g. "pick up object" and consists of a number of device primitives. In SBS, skills are the lowest form of programming an operator can use and raises the programming to an intuitive level.

2.5.3 Device Primitives

Device primitives are the basic motions and operations such as open gripper and camera read. These are the low-level commands that form a skill and are directly connected to the controlled hardware. This makes it possible to have a real time control of a robot.

2.5.4 Concept of SBS

As mentioned in Section 2.5.2, skills are the basic programmable element in SBS, which is available to the operator. An exact definition of a skill is a discussable matter. A simple and open definition is that a skill should follow a specific structure, illustrated in Figure 2.6. The presented structure allows to create skills necessary

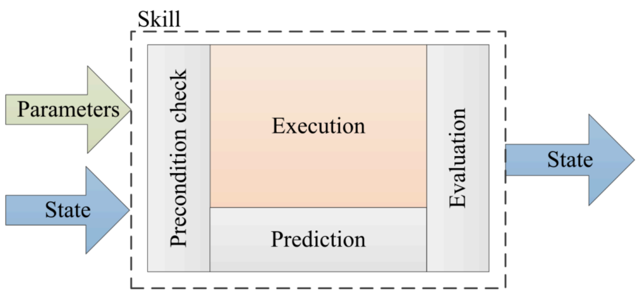


Figure 2.6: Model of a skill infrastructure. (Bøgh et al. 2012)

for intuitive assembly of specific tasks, while securing robustness through implemented preconditions and postconditions. Moreover, skills should keep a certain level of generality for flexible deployment. That is why execution of standard skills is preceded by a teaching session, where the exact behaviour of a skill is taught. The level of skill's generality is specified by the type of the skill:

Generic skills: Basic low-level skills made from sequences of device primitives, that are specified by teaching. Generic skills are an advanced implementation, since work with motion primitives is required, but they provide a high versatility and reusability.

Application-oriented skills: Those are made to solve specific needs of certain application in e.g. a manufacturing company. When suitable generic skills are not available to solve a specific case, a specialized skill can be developed. Such skills can have higher abstraction since they are made of device primitives and also of other generic skills (nested skills). Although convenient in a specific case, application-oriented skills lack the versatility of generic skills and thus the reusability.

The different segments of a skill shown in Figure 2.6 are important to secure desired robustness of a skill and thus safe execution:

Precondition: A number of conditions that need to be met in order to proceed to execution of the given skill. Preconditions are structured to return only true or false (i.e. proceed or fail) and can investigate either readings of the concerned component (manipulator, gripper, etc.), or use sensors or visual feedback for inspection. In case of a pick skill, the precondition can be e.g. "is the gripper opened?", "is the manipulator in range?", or "is the object present?". Last mentioned is an example of an inspection. When all preconditions are met, the skill can be executed. Having strict preconditions for all skills results in a robust program.

Execution: Main body of a skill, which executes orders, taught during a teaching session, after preconditioning was successful.

Postcondition: Postcondition is a term combining prediction and evaluation from Figure 2.6 and, as preconditions, can return true or false. A prediction states what is the expected result of executing a given skill. After execution, the evaluation compares the actual result of the execution and the prediction model, and can optionally output statistics of skill executions for diagnostic purposes. In case of a pick skill the postcondition could be e.g. "is the gripper closed?", or "is the object grasped?". If a postcondition evaluates the result as not satisfactory, the next skill in the sequence cannot be executed. Postconditions can be used for further increasing robustness and also as a diagnostics tool.

A powerful aspect of skills is "object-centring". When programming a pick and place task by traditional methods, one has to specify Cartesian coordinates of a specific object. While using skills, it is the object that is specified, not the coordinates. By using vision, or sensor devices, it is then possible to locate the object at runtime and carry out the grasping function from there. (Bøgh et al. 2012)

This section described the structure and concept of SBS, which is an important knowledge if the LH6 is to be integrated in it.

Chapter 3

Problem Formulation

As mentioned in Chapter 1, there is a general shift in how to design production systems, going from hard automation to soft automation where high efficiency is achieved without the loss of flexibility. The goal of this project is to solve part feeding challenges in an Industry 4.0 production line by developing an AIMM (next generation of the Little Helper project), that is able to raise the self-sustainability and flexibility of a smart production. The FESTO CP Factory used for assembling smart-phone dummies is taken as a context for this project (see Section 1.3), where the LH needs to be designed for feeding in the necessary parts to the dedicated assembly modules. LH is to be created by selecting and combining a fitting mobile platform and a manipulator. Integrating those two platforms poses certain challenges regarding battery life, logistics, communication, physical limitations, safety considerations and gripper selection, that need to be considered in design requirements of this project. Those challenges are explained in Section 2.4.

Final problem statement: *How can the part feeding tasks at the FESTO CP Factory line be solved by utilizing an AIMM with SBS integration?*

3.1 Solution Proposal

The part feeding of the smart production line is to be solved by creating a new version of Little Helper AIMM robot (LH6). A fitting mobile platform and a manipulator need to be selected in order to comply with described challenges in Section 2.4. The mobility is achieved by placing the manipulator on top of the platform by custom-created mounting, and by connecting the manipulator to a battery. When designing the mounting and deciding on the material, it is important to keep in mind weight limits of the platform and desired operational height for the manipulator.

The communication of all devices poses also a challenge. Mobile platform, manipulator, gripper and FESTO CP Factory need to be on a same Local Area Network in order to be able to communicate with each other through TCP/IP protocol. FESTO's MES can afterwards send commands to LH6, when it is necessary to carry out the given tasks. The mobile platform and the manipulator need to be programmed and controlled by SBS to provide higher user-friendliness for a potential operator with no programming experience. If not already, they have to be also integrated into the SBS program and ROS driver packages need to be developed.

Lastly, there are certain considerations to be done regarding safety of LH6. Since the robot is expected to drive in places accessible by other people, it is necessary to asses risks and follow certain rules regarding collaborative robots. The robot needs a unified emergency button, limited operational speed, and connection of safety break signals. Also, the manipulator should not be allowed to move while the mobile platform is driving, and during this time it should remain in defined home position.

The design requirements and considerations are further explained in Chapter 4.

Chapter 4

Requirements Specifications

This chapter presents design requirements of the LH6 (Section 4.1), as well as delimitations for the given part feeding use case (Section 4.2). The requirements discussed here are based on collected data in Chapter 2.

4.1 Design Requirements

This section discusses the requirements that the final LH6 should comply with. The design requirements are split into several groups (Hardware, Software, Safety, and Operation.)

4.1.1 Software

1. Software framework
The software framework for this project is SBS and thus LH6 should be programmable in SBS.

2. Accurate positioning control
LH6 should be able to move the manipulator with respect to the surroundings (e.g. machines it interacts with). This means that the manipulator may have to compensate for imprecise positioning of the mobile platform.

3. SBS Skills
For LH6 to be used with SBS, certain functionalities are needed. These functionalities are Drive, Pick, Place, Move, Communication and Calibration and are required in order for LH6 to solve the part feeding (see Section 2.2).

4.1.2 Hardware

1. Battery life
Due to the nature of smart production facilities it is not possible to predict the

cycletime and thus when a part feeding task is to be carried out. Therefore it is important that the battery life is sufficient long in order to not risk delaying production because of recharging. Since the task the LH6 is carrying out is originally done by an operator, the LH6 should not have more down-time than an operator. In general, the battery life should aim to withstand at least a standard daily schedule (3.5 hr work, 1 hr break, 3.5 hr work).

2. **One common power source**
   For sake of simplicity and convenience, all electric components used for LH6 should be connected to one common power source.

3. **LH6 should be able to carry a minimum of 20 kg**
   The previous LH's were designed with a possibility to carry up to 20 kg, which exceeds the weight of the heaviest part that is to be fed, according to the use case (Section 2.2).

4. **The manipulator should be able to lift at least 5 kg**
   Similarly to requirement 3, the heaviest object that can be expected to be fed, is the PCB tray, which weighs approx. 2.2 kg. The selected manipulator thus needs to be able to lift the mentioned tray, plus the equipped gripper.

5. **Electric gripper is to be used**
   LH6 is an AIMM, where weight and battery life are essential points. This project is limited to electric grippers because they do not need a compressor, which conserves payload and space in the compartment. Furthermore, they offer more functionalities and SBS already contains drivers for certain electric grippers, which is considered advantageous.

6. **Height of the table should be adjustable**
   If the LH should be as flexible as possible, then the table mounting of the manipulator should have legs with adjustable height. This makes the implementation on different production lines easier, thanks to fast adaptability to various heights.

4.1.3 Safety

1. **Unified emergency button**
   LH6 should be equipped with only one line of emergency buttons. This means that whatever emergency button is pressed, all actuators on LH6 will stop. It is needed to avoid confusion of function of each emergency button.

2. **Common safety break**
   If a protective stop on the manipulator or the mobile platform is triggered, all systems in LH6 should enter a safety break, stopping all motions.

3. Movement restrictions
For safety only the manipulator or the mobile platform is allowed to be operating at a time. This means that the manipulator cannot move before the mobile platform has come to a complete stop and vice versa.

4.1.4 Operation

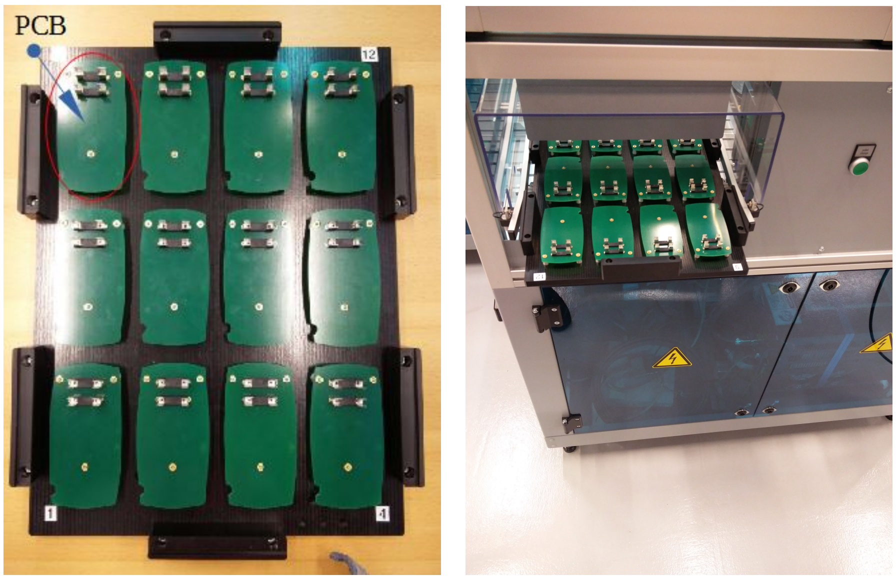
1. Autonomous recharging
The recharging is a task that is expected to occur on a daily basis. It is believed that the most efficient way to carry it out is to program the robot to autonomously visit a docking station for recharging when the battery/batteries goes below 10%.

2. TCP/IP communication interface with FESTO CP Factory
LH6 will be integrated in a smart production facility where it will fill out the role of part feeding. Therefore it is necessary to have an external communication channel for the factory to notify LH6 about low part stock. This communication channel exists only for notifying the LH6 of low stock.

4.2 Delimitations

For the scope of this project, there are certain delimitations that were decided upon. The aim is to solve only part feeding to one module of the FESTO CP Factory, which is the PCB assembly module. Multiple PCB's are inserted in the module using a tray. The opening, in the feeding point, for the tray is 321.65 mm wide and 50.73 mm high and the tray is 300 mm wide, 400 mm long and 38 mm high. LH6 needs to be able to replace an empty tray with a full one (see Figure 4.1). In order to do that, a precision requirement needs to be established, i.e. LH6 should have an accuracy of $\pm$ 10.8 mm, based on the mentioned measurements.

Because of the one-module focus, it has been decided also to fit the construction to the delimited scenario, meaning that the height of the table-mounting for the manipulator will not have an adjustable height. Instead, an optimal height for handling the PCB trays will be specified, and the table mounting will be manufactured in that desired height with no adjustment options. This optimal height is to be based on the height of the tray feed-in point, which is 975 mm above ground (see Section 2.4). Furthermore, the problem of communication has been reduced to only be between the LH6 and the PLC of the particular assembly module that is being used. It is not considered necessary for this project to implement communication with the rest of the PLC's in the FESTO CP Factory.



(a) The PCB's on a tray.

(b) The tray to be reeled into the PCB assembly module.

Figure 4.1: Investigation of the manipulated part and feeding point.

The main focus is put on establishing the communication between the PLC and the LH6 and carrying out the part feeding task itself, thus the issues, like autonomous charging, are excluded from the scope for this project. It is also assumed that LH6 has a full PCB tray available at any given point, since processes before the part feeding are out of scope of this project. Furthermore, skills like Move, Pick, Place are already in SBS and are assumed to be sufficient for solving the given part feeding task.

There is a considerable amount of safety regulations (ISO standards) that should be looked into when designing a mobile manipulator, such as LH6. However, the product of this project is dedicated for research purposes, not for commercial usage, so even though such regulations were investigated into certain extent (see Section 2.4.5), complying with them is not considered a high priority goal. Based on this decision, maximum operational speed limits are not taken into account.

All the mentioned delimitations are applied because of limited time and resources allocated for this project. After taking the delimitations into consideration, a list of final requirements was derived, which can be seen in Section 4.2.1.

Given Hardware

For this project some hardware was provided. Therefore the mobile platform of LH6 is a MiR100 and the manipulator is a UR5. These two solve some of the requirements such as a payload and lifting capability so they become limits of the LH6 instead a requirement. MiR100 also has a router on board which can be used for communication between all subsystems. However, MiR100 brings the need of some new functionalities, like a Drive functionality.
A Robotiq 3-Finger gripper was also given and will act as the tool for this project's part feeding.

4.2.1 Final Requirements

The enlisted requirements in this section are the ones this project focuses on. The rest of the requirements mentioned in Section 4.1 are reserved for future work. Figure 4.2 illustrates the focus of this project, which is to develop an AIMM to carry out part feeding of a specific assembly module, by communicating with the PLC of that module (communication with MES is neglected). The delimited requirements are listed in Table 4.1.

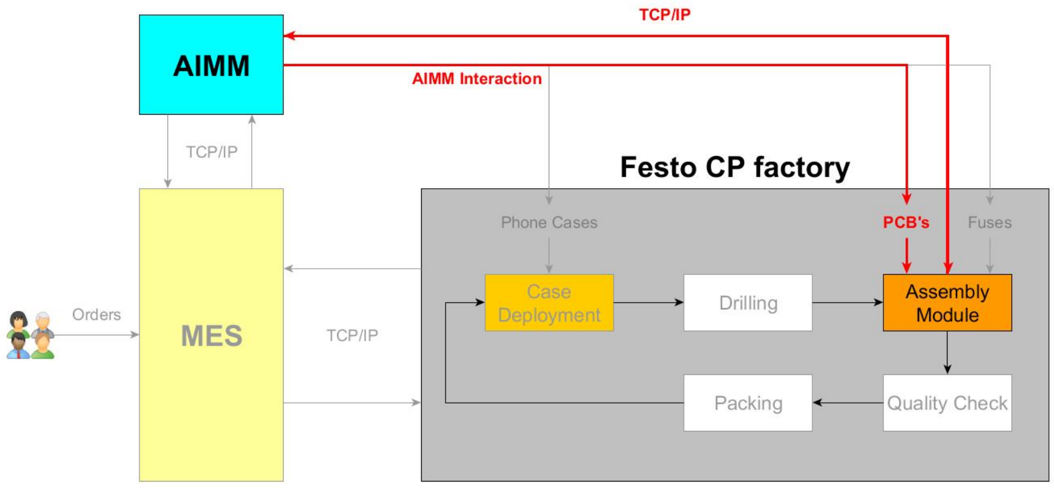


Figure 4.2: Focus of this project is highlighted in red. Features neglected through delimitation are faded.

Item	Description	Value	Unit
1	Battery life	3.5 - 1 - 3.5	Hours
2	LH6 payload	20	kg
3	Manipulator payload	5	kg
4	Manipulator precision	±10.8	mm
5	Electric gripper	-	-
6	Common power source	-	-
7	Programmable in SBS	-	-
8	Drive Functionality	-	-
9	Common emergency buttons	-	-
10	Restrict manipulator movement when mobile platform moves	-	-
11	Restrict mobile platform movement when manipulator moves	-	-
12	Communication with the PLC of the assembly module PLC	-	-

Table 4.1: Table of requirements.

Chapter 5

Solution Proposal for Delimited Case

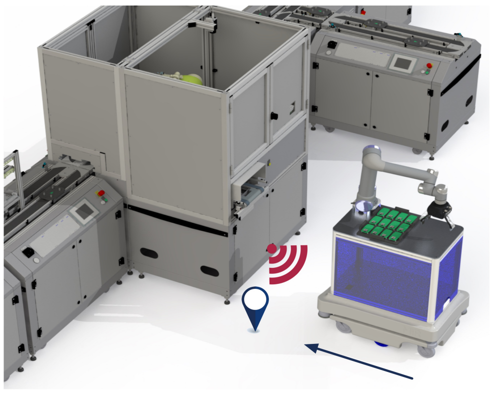
After the part feeding has been delimited, a general description of the case can be made. This chapter gives a step-by-step description of the part feeding process by going through the manual operations and then through a solution proposal on how it all can be automated by a new generation of a LH.

When the assembly module registers that the tray, containing the PCB's, is empty, it reels out the tray for an operator to replace. The tray is then accessible on a conveyor on the side of the module. Once an operator has replaced the empty tray with a new, full one, he/she can press a button located on the module next to the conveyor which reels in the tray and the operation of the module resumes. Alternatively an automation of the delimited case can be divided up into three processes.

Driving to: When the assembly module detects that the tray is empty it sends a signal to the LH6. As mentioned in Section 4.2, it is assumed that the LH6 already has a full tray on its platform. It then drives to the assembly module. This process is illustrated in Figure 5.1a

The arrival: When LH6 has arrived, a calibration that calculates the offset of the LH6 is done. Since MiR100's precision is ± 10 cm, it is needed for precision so that the manipulator does not collide with the surroundings. Once the offset is calculated, the manipulator moves with respect to the surroundings rather than the mobile platform it is placed on. The calibration is shown in Figure 5.1b.

Exchange: The LH6 picks out the empty tray and places it on its table next to the new tray (see Figure 5.2a). Afterwards it places the new tray inside the module (see Figure 5.2b). The last thing to do is to signal to the assembly module that the exchange has taken place and the new tray is ready to be pulled in (see Figure 5.2c).


(a) The assembly module sends a signal, which the LH6 responds by arriving to a given destination.

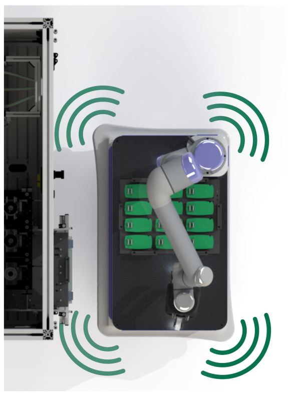
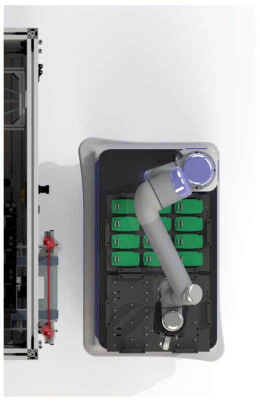
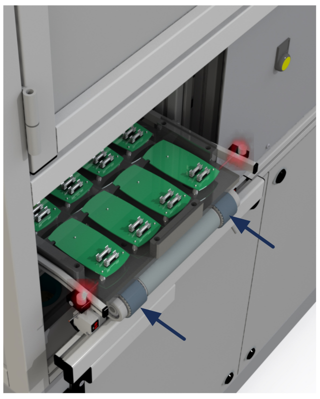

(b) On arrival a calibration is done to calculate the offset of the LH6.

Figure 5.1


(a) The LH6 places the empty tray on its table and then picks up the new tray.


(b) The new tray is placed on the conveyor and pushed halfway in.

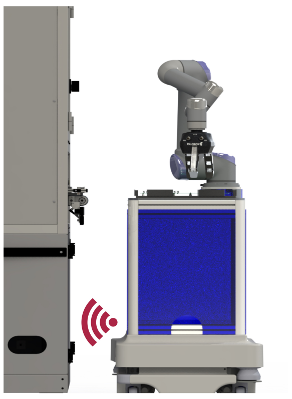

(c) A signal is sent to the assembly module that the tray can be reeled in.

Figure 5.2

Part feeding is then fully automated, since there is now no need for any human labour on the assembly module. In order to perform these operations, LH6 needs certain functionalities, both hardware and software wise, which is described in depth in Chapter 6.

Chapter 6

LH6 Architecture

LH6 is an AIMM created by several hardware components. It consists of four main parts:

- Mobile platform (Section 6.1)
- Manipulator (Section 6.2)
- Gripper (Section 6.3)
- Sensors (Section 6.5)

Furthermore, there are other aspects to be taken into consideration such as electric circuit and network connection (Section 6.6) and hardware design (Section 6.4.
As mentioned in the delimitation (see Section 4.2), specific products are provided and used for the solution (MiR100, UR5, Robotiq 3-Finger). This chapter contains a description of the mentioned components and their integration in the ROS environment, together with a description of LH6's hardware design. Regarding the ROS integration, the ROS drivers for the UR5 and Robotiq 3-Finger used in this project are taken from integration in previous LH projects. For that reason they are not described. The ROS driver for the MiR100 platform is, however, created in scope of this project. Its description can be found in Section 6.1.1.

6.1 Mobile platform - MiR100

In order to make it possible for the LH6 to drive and navigate autonomously in a dynamic environment, a mobile platform with this ability is required for the LH6. As a given hardware component, a mobile platform from Mobile Industrial Robots called MiR100 is provided (see Figure 6.1). MiR100 is a flexible mobile platform which by several sensors is able to map e.g a factory hall, and drive to predefined positions in the created map, while detecting and avoiding obstacles. MiR100 has a local network, to which the user can connect using a smart phone, computer or a tablet and program/control the robot through a web interface, which achieves to simplify the programming of MiR100.

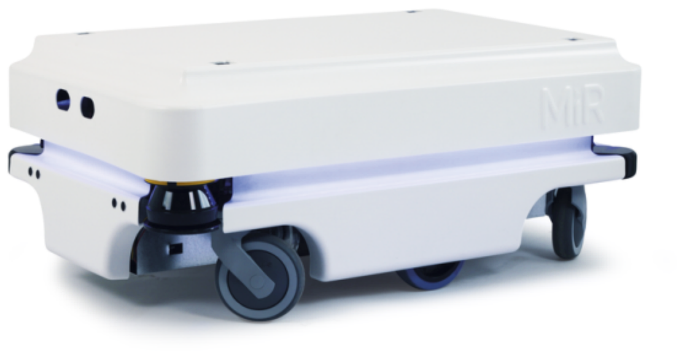


Figure 6.1: Picture of MiR100. (MiR-ApS 2016c)

The MiR100 has a running time of 10 hours, and has a max speed of 5.4 km/h to drive forward and 1 km/h driving backward. It reaches a given destination with a positioning accuracy of 10 cm in radius. It is important to note that the MiR100 has two actuated wheels in the center and in each corner it has a 360 degrees turn wheel.

There is a ROS master running on MiR100 and its topics and services can be accessed by using e.g. Rosbridge, or a REST API. Those are ways to transmit the topics and services in a JSON format over the network. Another way to access the ROS master of MiR100 would be by the following command: *export ROS_MASTER_URI=http://mir.com:11311*. The port 11311 is a default port, however, different one can be specified.

MiR100's ability is to detect and avoid obstacles. It is not just able to coordinate the speed with how close objects are, but it is able to evaluate the situation and stop by itself if needed. This applies not only for objects standing still but also objects in motion. The MiR100 has LED's and speakers, which can be used for warnings and announcements. The last safety feature is the emergency button placed on top of the MiR100.

Since the MiR100 is able to move around, it also needs a navigation system. In total there are three sensor technologies that form the detection and mapping system of the MiR100. The most important sensors are the SICK laser scanners S300. The laser scanners enable MiR100 to have view of 360 degrees, but since these are mounted on the front left corner and the back right corner it has a view of 270 degrees on both of the scanners. However, it is important to notice that the scanners provide only planar reading, i.e. they scan the 2D plane in height of 20 cm above the ground. This can be a challenge, when e.g. trying to detect a table. The second sensor type is a 3D camera. It is able to detect objects ahead 5-50 cm above the floor. The last kind of sensors on the MiR100 are the Ultrasonic scanners. They are enabling the MiR100 to detect transparent objects ahead of it.

6.1.1 MiR100 Driver

In order to prepare MiR100 for any SBS implementation, a ROS driver needs to be created. In the documentation of the MiR100, found on the MiR support forum (https://mirrobot.atlassian.net/servicedesk/customer/portal/), there are tutorials on how to connect to the robot through a Rosbridge server and a REST API, when using Python. These tutorials are used as a base for the MiR100 driver.

From the documentation related to the Rosbridge server, a Python script with functions that allows communication between the Python script and the services or topics, published on the MiR100, is used in the driver.

The documentation related to the REST API was found via a web browser connected to the MiR100 network on `http://mir.com:8080/doc` and in a Python script found on the MiR support forum. The homepage consists of different REST API commands, and the Python script consists of some examples of how to call these commands.

In the MiR100 driver the two communication interfaces have different functionalities. As mentioned, the Rosbridge communicates with the MiR100 through requests sent directly to the topics or services published on the MiR100. Therefore, the Rosbridge in general is used for lower level functionalities, such as setting the command velocity (device-primitives). Setting the command velocity using the Rosbridge communication is done the same way as on many other mobile platforms. Simply publish the velocities to a topic, using the topic `"/cmd_vel"`, or the service in the MiR100 driver `"/mir_set_cmd_vel"`. The same way the modes of the robot can be set using the service of the MiR100 `"/mircontrol/set_mode"` or the service in the MiR100 driver `"/mir_set_mode"`. On the other hand the REST API, can be used for higher level functionalities and can therefore be used directly as a skill, e.g. the `"/mir_drive_to_position"` which is used to execute a "Drive To" skill described in Section 8.4. This means, that in order to make a Drive To

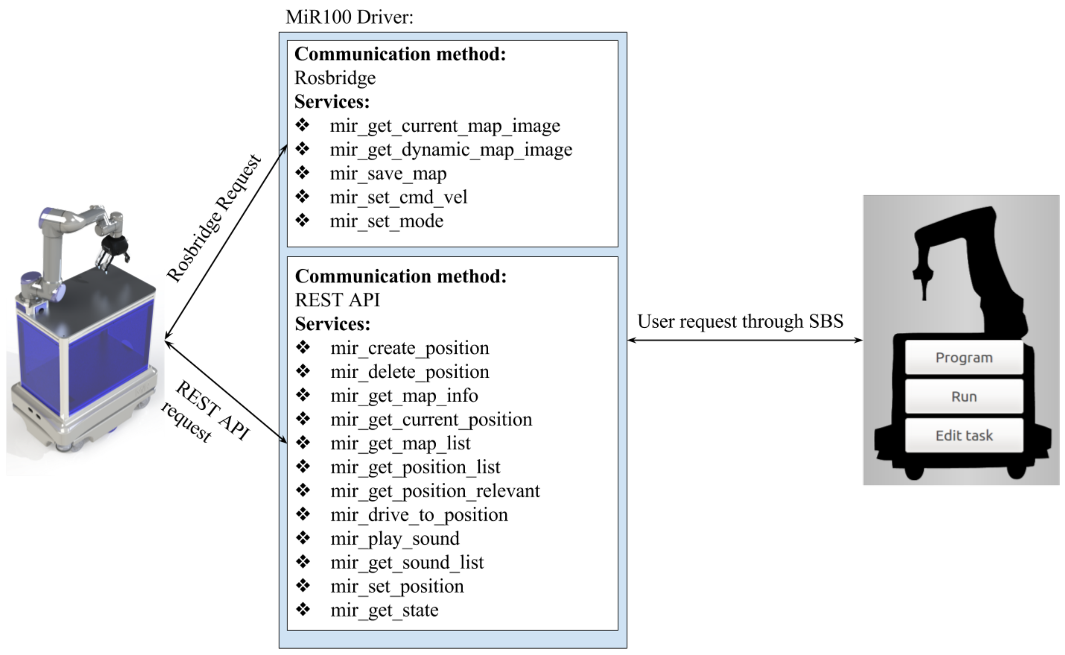


Figure 6.2: Structure of the MiR100 driver.

functionality, the REST API is necessary as it allows creation of both positions and missions. The robot moves to given positions by creating a mission. A mission contains necessary information such as how many times MiR100 should retry to find a path to the goal position if the path is blocked or what the precision tolerance is when it reaches the goal position. In this way the MiR100 driver can take a name of a position and drive to it. For all the functionalities of the MiR100 driver see Figure 6.2.

The following list describes the main functionalities of the MiR100 driver and how they are intended to be used. The functionalities listed in this description are related to the essential MiR100 functionalities in the SBS (for a list of all the services and their communication method see Figure 6.2).

`"/mir_set_cmd_vel"`:
: This service makes it possible to manually drive the MiR100. The intended use of this functionality is to manually drive the robot to positions in order to teach them.

`"/mir_create_position"`:
: This service creates a position on the currently selected map. The service needs the following arguments: `"name"`, `"pos_x"`, `"pos_y"`, `"orientation"`. The intended use of the service is when to save the current position of

the MiR100. This service is often used together with `"/mir_get_current_position"`.

`"/mir_get_current_position"`:
: When requesting this service, it returns the current position and orientation (`"pos_x"`, `"pos_y"`, `"orientation"`) of the MiR100 with respect to the active map.

`"/mir_get_position_relevant"`:
: The response of this service returns a list of position names existing on the active map of the MiR100. The intended use of this service is to choose among created positions when executing the `"DriveTo"` functionality.

`"/mir_drive_to_position"`:
: This service executes the Drive To functionality discussed in Section 4.1.1. The service takes the following arguments: `"name"` (position name), `"num_retries"` (set by default to 10), and `dist_to_goalthreshold` (set by default to 0.1). This service is in other words executing the `"DriveTo"` functionality.

6.2 Manipulator - UR5

In order to follow the conventions of previous LH's and increase the flexibility and variety of tasks that the LH6 can solve, a manipulator is integrated together with the MiR100. For this project a specific manipulator, UR5 from Universal Robots, is used (see Figure 6.3). UR5 is a collaborative industrial manipulator.

This section assesses UR5's specifications, safety considerations, communication possibilities, and identifies possible challenges related to the robot.
If the UR5 implementation is to be successful, the specifications of the robot need to be evaluated if they are sufficient for the desired application. The list of basic attributes of UR5 can be seen in Table 6.1. Considering the given use case scenario described in Section 2.2, the payload of 5 kg seems to be more than enough for handling all the mentioned parts, especially the tray with PCB's, since the delimitation of this project is only to carry out the PCB part feeding (see Section 4.2). The weight of the robot alone complies with MiR100's weight limits (see Section 6.1) and keeps a high payload reserve for the rest of the components.
While on the same network, the UR5 is able to communicate with other equipment through a TCP/IP protocol, where usually the robot acts like client and a computer as server. When there is an attempt to communicate with the robot by a ROS node, it is required to upload a script written in URScript (similar to Python), which then translates all the commands sent by the node to the robot. (Universal-Robots-A/S

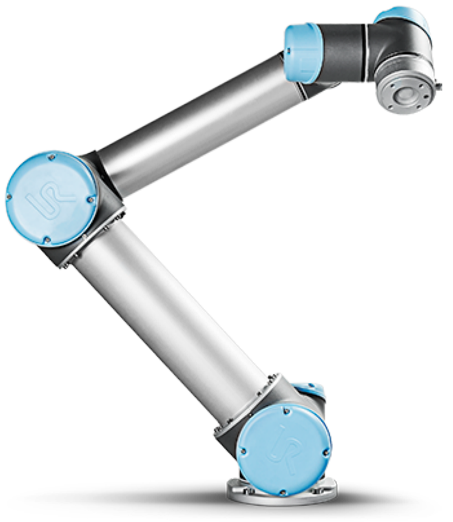


Figure 6.3: UR5 robot. (Universal-Robots-A/S 2016b)

Weight	18.4 kg
Payload	5.0 kg
Reach	850 mm
Degrees of Freedom	6

Table 6.1: Basic specifications of the UR5 robot. (Universal-Robots-A/S 2016d)

2016a)

Since the UR5 is supposed to be operating in non-enclosed environment as a collaborative robot, safety of the implementation must be considered. All products of Universal Robots posses an algorithm for applied force estimation, which should ensure that safety break of the arm will activate before reaching applied force of set threshold (150 N by default). However, certain tests proved that depending solely on the algorithm without having any supplemental force/torque sensors is not necessarily reliable, meaning that, due to negligence of safety and without risk assessment, serious accidents can still happen. (Bonev 2014)

The power supply of the UR5 poses a challenge. Because of the required mobility, the robotic arm cannot be traditionally connected to a power socket. It has been also estimated that connecting UR5 to the battery of MiR100 would significantly reduce the battery life of the LH6. It might be necessary to add extra power source on the platform, in order to increase the battery life of the LH6 to an acceptable level.

6.3 Gripper - Robotiq 3-Finger

For this part feeding challenge it is needed to manipulate the PCB tray. This tray weighs around 2.2 kg and is hard to grasp around for a gripper. For this project the Robotiq 3-Finger Adaptive Robot Gripper has been given. The Robotiq 3-Finger gripper has three fingers and has a weight of 2.3 kg. It is able to lift up to 10 kg.(Robotiq 2016a)

Because of the design, weight of the gripper and tray, and the payload of UR5 (5 kg), it can be assumed that UR5 should be able to lift the tray without any difficulties.

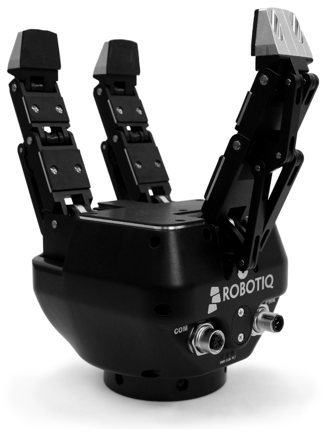


Figure 6.4: Robotiq 3-Finger. (Universal-Robots-A/S 2016b)

6.4 Design of LH6 Mounting

In order to integrate the MiR100 and the UR5, there are several considerations that need to be done, related to the physical frame integrating the two robots.
For the frame, aluminium alloy profiles from Paletti are chosen. The legs of the frame are not placed exactly on top of MiR100's mounting points. They are instead moved a few centimetre beyond those points in order to increase the workspace of the tabletop and enabling LH6 to have two trays on top of it (see Figure 6.5). The height from ground to tabletop is chosen to be 975 mm so the height is the same height as the conveyor on the FESTO CP Factory.

On top of the frame a raised stand for the UR5 is placed. This stand is placed on the front right corner to make it easier for the UR5 to reach into the assembly module and grasp the PCB tray. Furthermore, it is raised above the table to make sure the UR5 does not hit the trays on the table while turning.

Acrylic glass is mounted all around the frame, in order to create a compart-

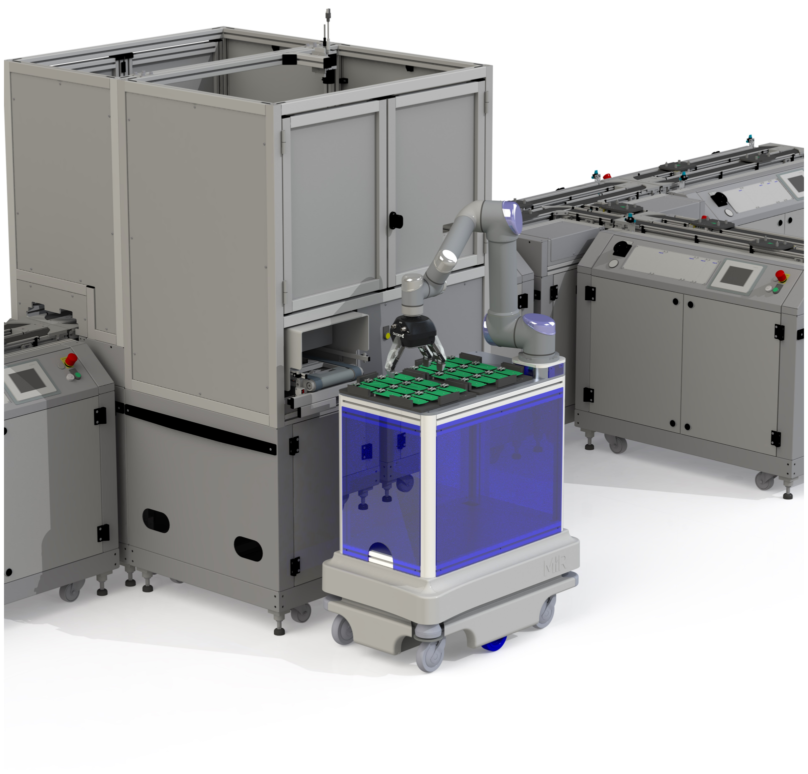


Figure 6.5: The complete assembly of LH6 and FESTO CP Factory.

ment for all internal components of LH6. The glass is mounted with four and six screws so it is easy to open it and do adjustments to the internal components. In Figure 6.5, a simulation of LH6 and FESTO CP Factory is shown.

6.5 Sensor - 3D Camera Calibration

When a robotic manipulator is programmed to move to a given position in 3D space, it does so assuming that (0,0,0) coordinate is placed at the base of the robot. This means that when it is placed on a mobile platform, like LH6, all the programmed points are placed in the space with respect to the platform. This is not a problem if the manipulator is only interacting with object, placed on the LH6 platform. Since MiR100 has a position accuracy of 10 cm radius (see Section 6.1), interacting with the world around it becomes a problem for the manipulator since it can not be sure that objects are placed 100% in the same spot every time. This section consists of the architecture of how this problem is solved, using a ROS driver and a theory from a paper, investigating fast calibration for AIMM's. (Andersen et al. 2013)

A solution to this is to have a fixed point in the world coordinate system to which all the robot movements are done with respect to. This means that, even if the mobile platform did not hit the same spot as when the manipulator was programmed, by doing a calibration the manipulator can account for the offset and thus interact with the surroundings as intended.

Using a ROS driver, which is able to return a transformation matrix from a 3D camera to a QR-code, this fixed point can be found. In order for the ROS calibration driver to work, a 3D camera and an RGB camera are needed. In this project an ASUS Xiton 3D Camera is used and mounted on the LH6. Furthermore, it will be necessary to have a QR-code at each location where the LH6 needs to be calibrated, in order to interact with its surroundings. The QR-code acts as the reference point for the robotic manipulator so all movements are done with respect to its position in space.

When a calibration is initiated, the computer takes an image from the camera and scans it for a QR-code. When the QR-code is located, the three corners of the QR-code from the depth image are used to create a plane in 3D space. This plane can then be used to calculate the transformation from the camera to the QR-code.

In order to obtain a world frame, a transformation matrix from the QR-code to the tool of the manipulator (${}^{Q}_{t}T$) is needed. ${}^{Q}_{t}T$ is expressed in Equation 6.1 and illustrated in Figure 6.6.

$${}^{Q}_{t}T = {}^{Q}_{C}T \cdot {}^{C}_{B}T \cdot {}^{B}_{t}T \tag{6.1}$$

${}^{Q}_{C}T$ is the transformation from the QR-code to the camera (obtained by taking the inverse of what the ROS driver returns), this is dynamic and changes with respect to the orientation and position of the mobile platform. ${}^{C}_{B}T$ is the transformation from the camera to the base of the manipulator. This transformation is constant. Lastly, ${}^{B}_{t}T$ is the transformation from the base of the manipulator to the tool. A static ${}^{B}_{t}T$ is used to calculate ${}^{Q}_{t}T$ when teaching. During the teaching ${}^{B}_{t}T$ is obtained by a kinematic model of the manipulator.

When the mobile platform moves, and arrives back to the position from where the calibration is done, ${}^{B}_{t}T$ becomes dynamic, and is the transformation needed to be calculated in order to take account for the offset. The equation for ${}^{B}_{t}T$ can be seen in Equation 6.2.

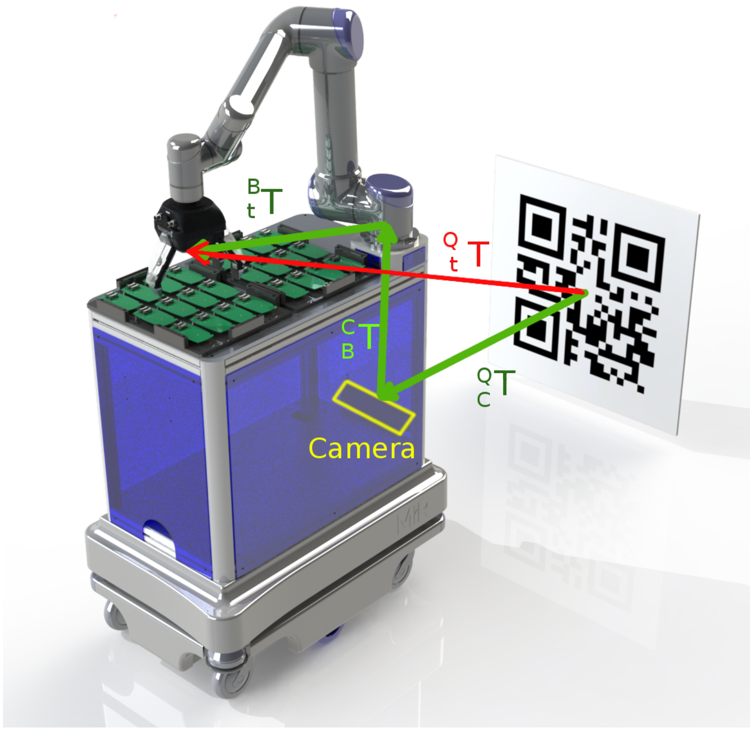


Figure 6.6: Transformation from the QR-code to the tool.

$${}^{B}_{t}T = ({}^{Q}_{C}T \cdot {}^{C}_{B}T)^{-1} \cdot {}^{Q}_{t}T \tag{6.2}$$

Since the camera will not be placed at the base of the manipulator, it is necessary to calculate the transformation from the camera to the base (${}^{C}_{B}T$), which is used as a constant for any given case. ${}^{C}_{B}T$ can be obtained either by measuring or calculated using the camera and a QR-code placed in the tool of the manipulator (see Figure 6.7). The transformation from the QR-code to the tool (${}^{Q}_{t}T$) can be measured, using the center of the QR-code and the geometric of the tool. ${}^{B}_{t}T$ can be calculated using the kinematic model of the manipulator. ${}^{Q}_{C}T$ can be determined by using aforementioned method. ${}^{C}_{B}T$ is unknown and the parameter to be determined. The calculation for ${}^{C}_{B}T$ will thus look like Equation 6.3.

$${}^{C}_{B}T = {}^{C}_{Q}T \cdot {}^{Q}_{t}T \cdot {}^{t}_{B}T \tag{6.3}$$

Note that in this equation ${}^{Q}_{t}T$ is the transformation from the QR-code to the tool directly, i.e. a measured distance.

A camera calibration skill is already implemented in SBS using the ROS driver

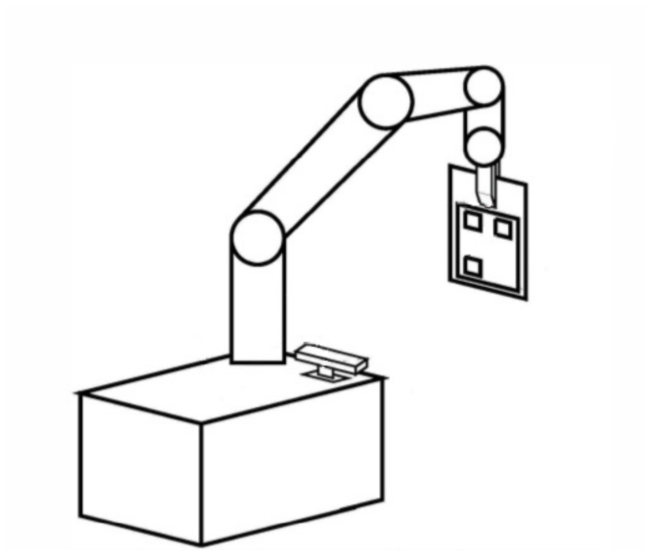


Figure 6.7: Illustration of setup.(Andersen et al. 2013)

called QR-pose along with openni2 to connect to the camera. QR-pose uses a program called zbar to locate and read the QR-codes. All these three programs needs to be installed and running before the skill can be used.

6.6 Network and Electric Circuit of LH6

With LH6 assembled accordingly to Section 6.4, the last thing to do is to design a fitting electric circuit in order to power up all electric components onboard. Moreover, all the main components (MiR100, UR5, Robotiq 3-Finger) need to be on the same network with a PC running the dedicated drivers. In case of both the network connection and the power supply, it is unacceptable for the deployed solution to impair mobility in any way.

6.6.1 Network Setup

In order to establish the TCP/IP communication between LH6 main components and their ROS drivers, a network needs to be set up. To not influence the mobility of LH6, the computer running the drivers needs to connect to all the components wireless. The wireless router integrated inside of MiR100 can be used for this purpose, i.e. if UR5 and Robotiq 3-Finger are connected to this router, then all three devices are accessible through connection to MiR100's WiFi. For future convenience, UR5 and Robotiq 3-Finger are not going to be connected directly to the router. Instead, they are to be connected to a network switch inside LH6's compartment and the switch is to be connected to the router. Such installation makes network troubleshooting and future developments of LH6 easier, since connection of present devices can be investigated and new devices can be connected without the need to take the frame down from MiR100, whenever a modification is necessary. The logistics of the LH6 devices' network can be seen in Figure 6.8 along with all devices' fixed IP addresses. A starting point in deciding on those addresses was MiR100, since it has a static IP which is not easily changeable. That is why all the other devices (UR5, Robotiq 3-Finger, and the PC) need to be given MiR100's network prefix (192.168.12.x). Their host number (x) is an arbitrary 8-bit integer (0 - 255). With the network set up correctly, the drivers can now run on the PC

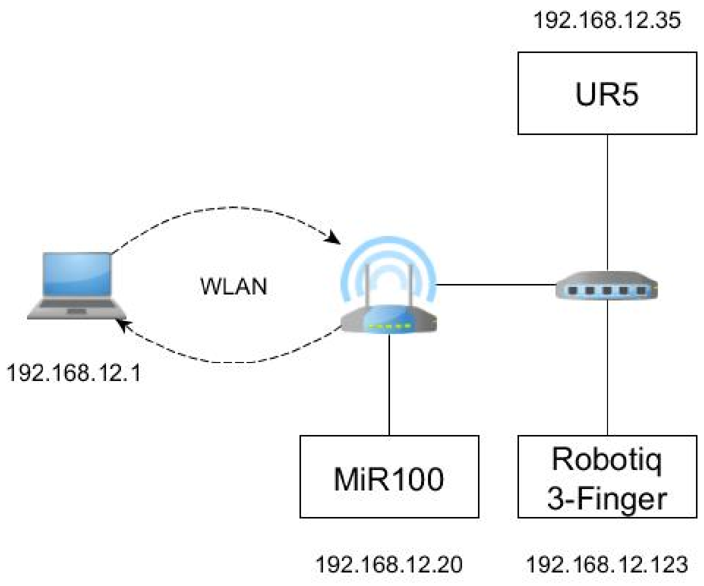


Figure 6.8: Network connection of LH6.

simultaneously and control all the devices at the same time.

6.6.2 Electric Circuit

This section describes the design of a circuitry equipped by LH6. There are certain considerations that need to be taken into account, since all the electric equipment on LH6 needs to be powered by batteries, in order to preserve mobility of LH6, which includes even the components that are usually connected to a socket (230 V). However, the components have also a possibility to be powered by certain voltage in DC, which can be found in their individual specification sheets. For the individual DC voltage needs, see Table 6.2. Some of the power consumption data could not be found directly in the specification sheets, but had to be calculated

Component	DC Voltage	Input Power
UR5	48V	200W
Robotiq 3-Finger	24V	36W
Linksys switch	12V	6W
MiR100	24V	94W

Table 6.2: Electrical specifications of the LH6 equipment. (Universal-Robots-A/S 2016c) (MiR-ApS 2016a) (Cisco 2016) (Robotiq 2016b)

from other provided information instead (see Equation 6.4).

$$\begin{aligned} P_{switch} &= 12V \cdot 0.5A = 6W \\ I_{MiR} &= \frac{39Ah}{10hr} = 3.9A \\ P_{MiR} &= 24V \cdot 3.9A = \ 94W \end{aligned} \tag{6.4}$$

The information contained in Table 6.2 can be used for deciding on appropriate power source. The choice of power supply is of high importance, since it also directly determines the up-time of LH6. MiR100 can provide its battery to the circuit, which is a Li-NMC 24 V 39 Ah cell pack. In Equation 6.5 it is calculated, how much Ah is consumed on average, if the system is operated for three hours. The calculations are approximate since they do not include efficiencies of implemented converters.

$$\begin{aligned} P_{total} &= 336W \\ E_{3hr} &= 336W \cdot 3hr = 1008Wh \\ Q_{3hr} &= \frac{1008Wh}{24V} = 42Ah \end{aligned} \tag{6.5}$$

By reversing the process, the running time of the system while connected to the single battery can be expressed (see Equation 6.6).

$$\begin{aligned} Q_{battery} &= 39Ah \\ E_{battery} &= 39Ah \cdot 24V = 936Wh \\ t_{battery} &= \frac{936Wh}{336W} = 2.79hr \end{aligned} \tag{6.6}$$

This calculation concludes that the MiR100 battery alone should be able to drive the whole system for 2.79 hours. It has been decided to implement an extra battery of the same kind into the circuit. It needs to be noted that the two batteries are to be connected in parallel, which yields the same voltage of 24 V and the capacity is doubled (78 Ah), doubling also the average running time (approx. 5.5 hours).

The voltage of 24 V can be used to directly power only MiR100 and Robotiq 3-Finger gripper (refer to Table 6.2). In order to supply the rest of the equipment, a step-up converter is required for supplying the UR5 and step-down converter for the switch. A schematic of the final circuit with all the described components can be seen in Figure 6.9.

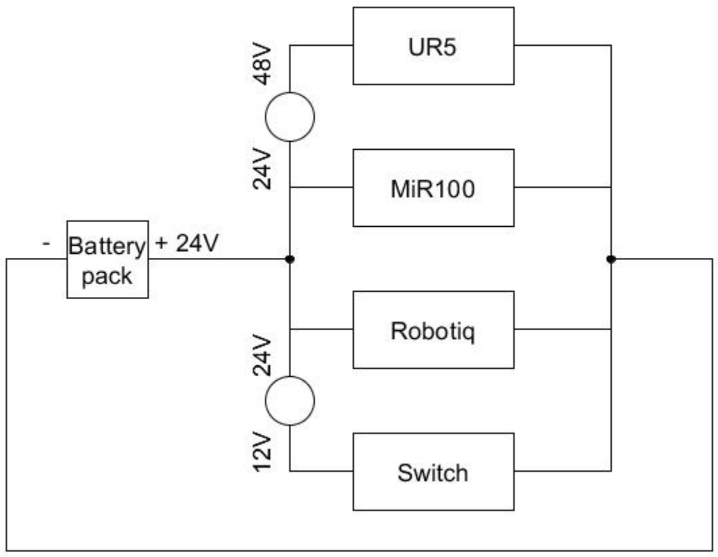


Figure 6.9: Scheme of the electrical circuit used on LH6.

Chapter 7

Integration

Related to a solution for the part feeding problem described in the delimitation (see Section 4.2), considerations are done with respect to the integration of the LH6, its components, the FESTO CP Factory and the different processes within the part feeding task. In the assembly module on the FESTO CP Factory there is a KUKA

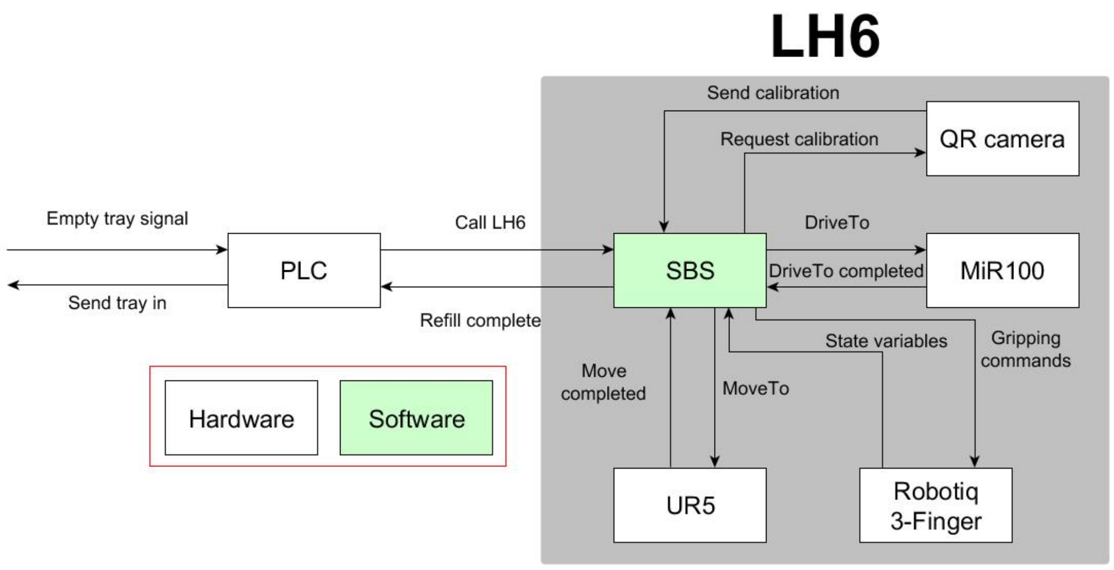


Figure 7.1: The different processes and components in the the part feeding solution.

manipulator. The KUKA controls the tray conveyor and receives signals from the sensors related to the part feeding task. Note, that the KUKA and the PLC are running two separate programs.

The procedure for the integration of the different components are illustrated in

Figure 7.1 and summarized in the following list:

1. The KUKA ejects the empty tray.
2. The PLC connects to computer running SBS and sends TCP signal.
3. SBS initiates a skill that makes the LH6 drive in the designated position.
4. Feedback from the MiR100 signalizes to SBS when the platform reached the final area.
5. Feedback from 3D calibration camera is used to counter the offset of the MiR100 platform.
6. SBS gets the UR5 to execute the feeding task.
7. A signal is sent to the PLC when the new tray is in.
8. The KUKA program activates the conveyor to reel the new tray into the cell.

This chapter consists of a detailed description of how these processes and components are integrated. The chapter is divided into two parts, as in the following description:

LH6 Integration with SBS:
In order to comply with the design requirements (see Section 4.1), the LH6 should be programmable through SBS. This means that all the ROS drivers related to the hardware components of LH6 (Robotiq 3-Finger, UR5 and MiR100), should be integrated into SBS. Since the drivers for the Robotiq 3-Finger, and UR5 are integrated into SBS, the focus of this part will be the integration of the drive to functionality (Section 7.1) and the QR-code calibration (Section 7.2).

Integration with FESTO CP Factory:
In order to automate the part feeding task related to the FESTO CP Factory, it is required to create an interface which establishes communication between the LH6 together with the PLC in the assembly module. With this communication the FESTO CP Factory should be able to inform LH6 when a feeding task needs to be carried out and LH6 should inform the FESTO CP Factory when a feeding task is done (see Figure 7.1). This section consist of a description of how this interface is designed, and how it is integrated into SBS (Section 7.3).

7.1 SBS Integration of DriveTo Skill

In order to solve the part feeding task using LH6, a functionality which is able to drive the LH6 from an initial position to the feeding point, is essential. This section consists of a description on how this drive to functionality, obtained by the MiR100 driver (see Section 6.1.1), is integrated into SBS. Initially SBS consists of skills and

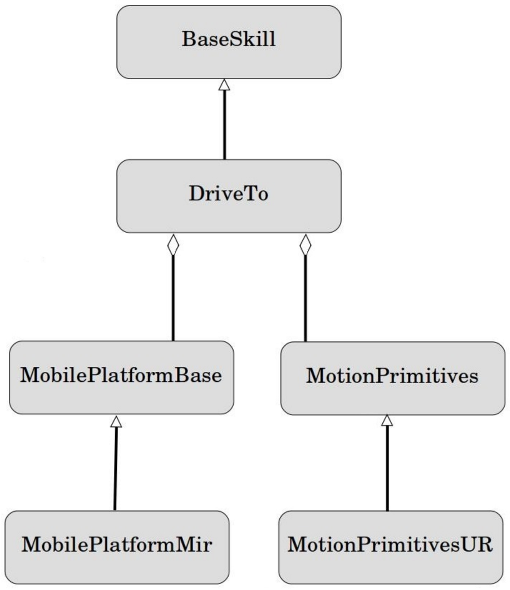


Figure 7.2: UML diagram of the classes connected to the DriveTo skill.

device primitives obtained using the ROS drivers for the UR5, the Robotiq 3-Finger and the QR-code calibration. However, SBS does not have a device primitive for mobile platforms, that is why the newly created MiR100 driver makes a good base for implementation.

To make the drive to functionality applicable with the initial skills in SBS and thereby increase the flexibility of the drive to functionality, a skill named DriveTo is created. This is done by implementing a class (also named DriveTo) into SBS, which inherits from the BaseSkill class (see Figure 7.2). To make the DriveTo skill able to communicate with the MiR100 driver a class called MobilePlatformMiR and a corresponding base class, called MobilePlatFormBase, is implemented. Through the MobilePlatformMir SBS handles all communication with the MiR100 driver using ROS services.

After the implementation of the DriveTo skill it can be chosen in the skill menu and used as well as other skills in SBS. The DriveTo skill is an objected centred ability as it drives the mobile platform to any given position. As seen in Figure 7.3

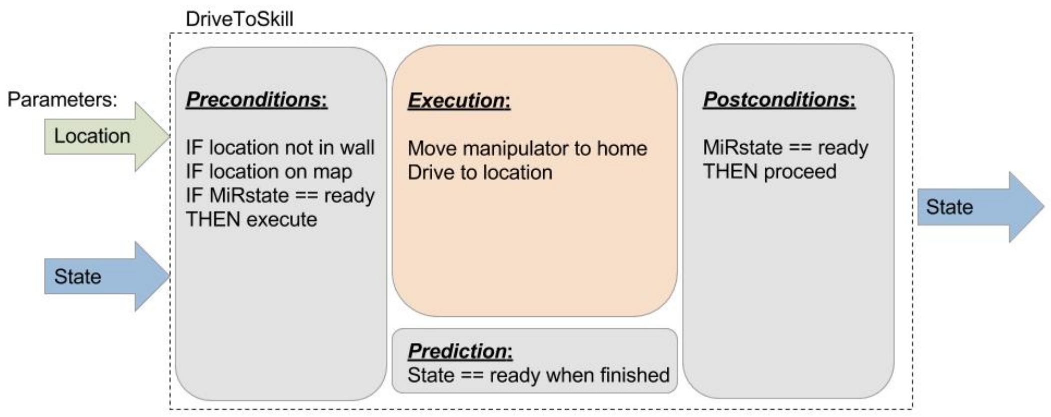


Figure 7.3: DriveTo skill concept.

the preconditions for the DriveTo skill is to check that the mobile platform is in a state where it is capable of receiving a position and then going to that position. It can also be whether the targeted position is accessible, i.e. not in a wall or outside the map. The postcondition is to check if the predicted state (ready state) matches up with the actual state. This is convenient since it is not desirable to have the manipulator move, if the mobile platform did not reach its targeted position. The Table 7.1 lists all the possible states the MiR100 can be in. It is only when the MiR100 is in state 3 (ready) it is available to execute the skill.
For safety, the manipulator moves to a predefined home position, before the platform starts driving.
The DriveTo skill is taught through the teach GUI in SBS, from where the MiR100 can be driven to a desired location which can be stored (see Figure 7.4).

	MiR100 States	Description
0	None	
1	Starting	MiR100 starting up
2	Shuttingdown	MiR100 Shutting down
3	Ready	Ready to execute
4	Pause	Pause from executing
5	Executing	Running a mission
6	Aborted	Mission aborted
7	Completed	Done executing
8	Docked	In the dock and charging the batteries
9	Docking	Navigating to docking station
10	Emergencystop	Emergency button is activated
11	Manualcontrol	A pause state where MiR100 can be moved manually
12	Error	General error state - requires error handling

Table 7.1: All possible states the MiR100 can be in.

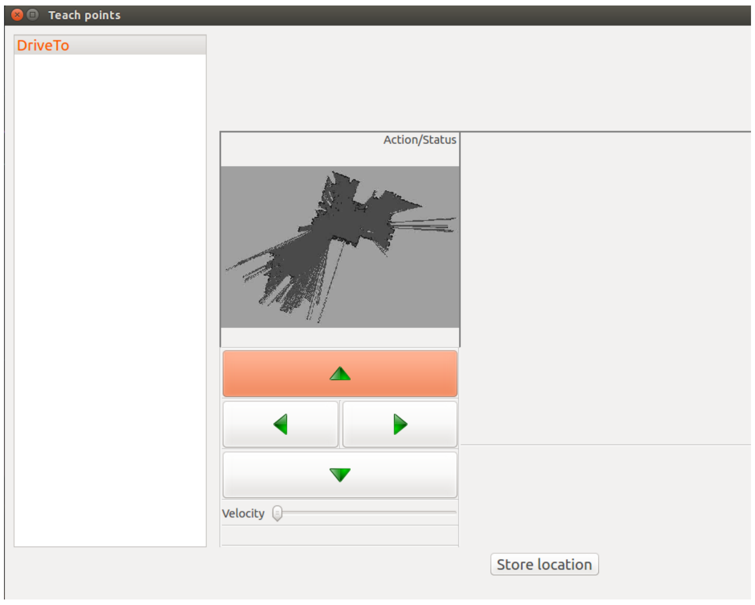


Figure 7.4: The teach GUI for the DriveTo skill.

7.2 SBS Integration of 3D Camera Calibration

SBS already supports usage of a 3D camera through an already existing skill, called QR Skill. The skill was originally intended to be used with a KUKA LWR manipulator, where it is possible to change the base of the robot directly on the robot controller, and thus for doing the calibration this functionality is needed. While an UR5 manipulator does not have this functionality, it is necessary to implement functions in SBS which are able to set the base of the UR5 according to what the 3D camera measures.

In the device primitives of SBS, for the UR5, a function which is able to carry out this functionality is implemented. The argument of the function is an array containing the Cartesian coordinates of the base that is to be set (the coordinates of the original base of the robot as seen from the QR-code).

In order to get the Cartesian position of the tool with respect to the new set base, a function that by default returns the Cartesian position of the tool with respect to the original base was modified. By doing so the function is able to return the Cartesian position of the tool with respect to any new set base (the QR-code).

Lastly, to do the calibration, a function which sends movement commands to the UR5 controller is modified. By the knowledge obtained in the previous mentioned functions, the coordinate of the taught position and an implementation of the theory from Section 6.5, the goal position of the tool with respect to the base will be calculated and sent to the UR5.

7.3 Integration with FESTO CP Factory

This section consists of a description of how the communication between the FESTO CP Factory and the LH6 is established and how it is integrated into SBS.

7.3.1 Part Feeding Automation

In order to fully automate the part feeding problem by LH6, the PLC connected to the assembly module is required to activate the conveyor related the module and receive feedback from the photo eye sensors (see Figure 7.5).

However, all the photo eye sensors and the conveyor button are connected to the assembly module's KUKA controller as its I/O's (see Figure 7.6). Since there is already an existing connection between the PLC and the KUKA manipulator in the module, an ideal solution would be to control those I/O's from the PLC through the KUKA controller. This implementation would require some additional programming on the PLC's side and editing of the default program on the KUKA controller. However, due to time constraints a simpler solution is chosen. By branching the relays of the button and the photo eye sensors, it is possible to connect those to the PLC's I/O's, as illustrated in Figure 7.7.

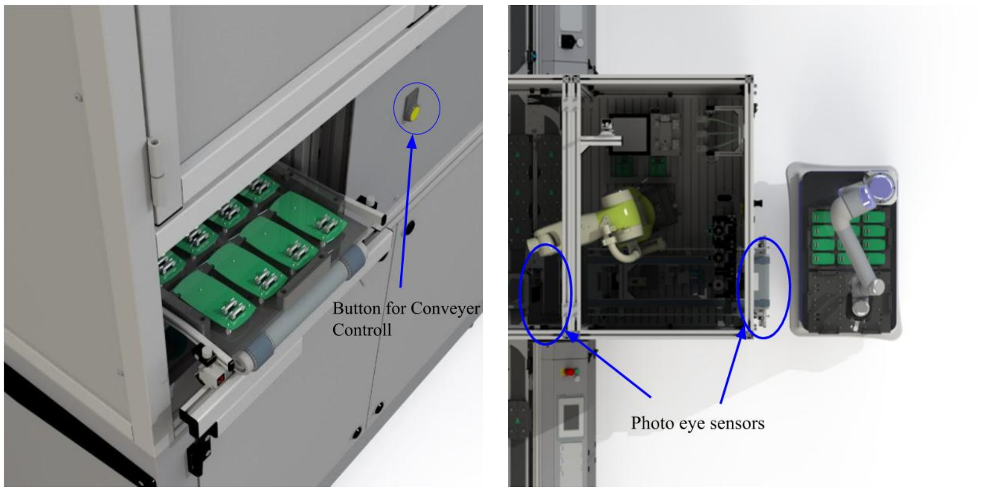


Figure 7.5: The sensors related to the assembling module.

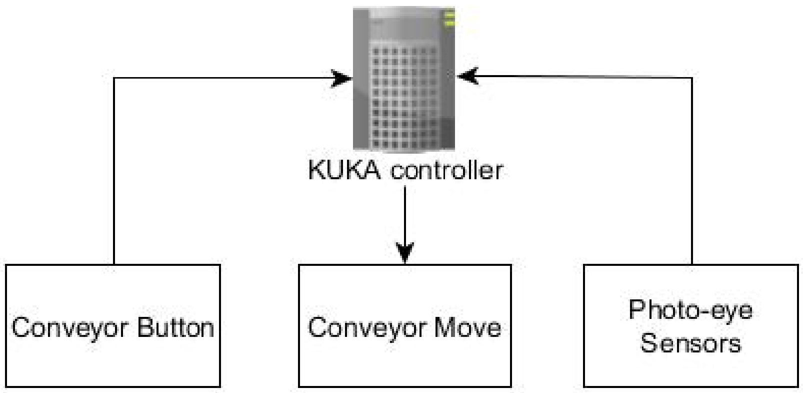


Figure 7.6: The initial control of the photo eye sensors and the button.

By connecting the PLC's output to the relay leading from the button, and assigning a variable to that output, it is possible to emulate the act of pressing the button via a program in the PLC without the need to make any changes in the KUKA program. Conveniently, the default program running on KUKA makes the tray roll out when there are no PCB's left, which can simplify the way the PLC program for this project is implemented. By doing the same with both the front and back photoeye sensors, except mapping it as an input, simply enables the PLC to receive readings from those sensors as well.

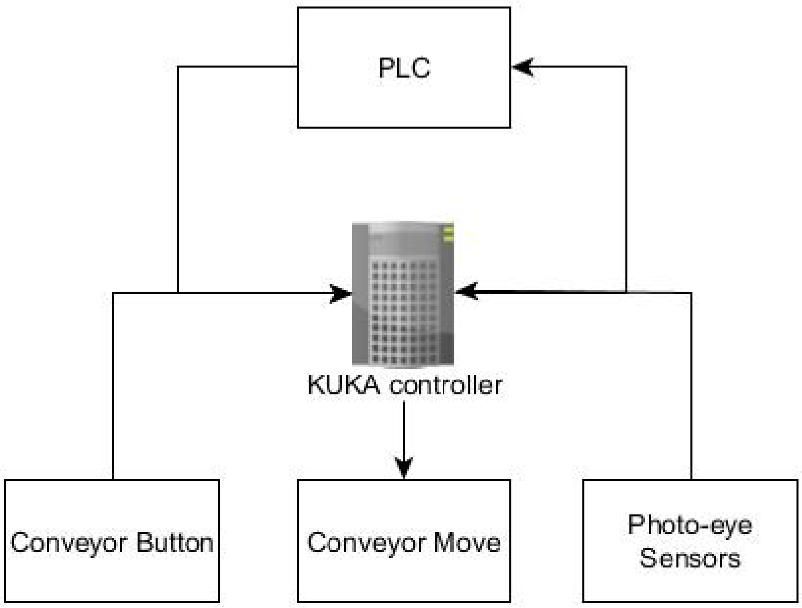


Figure 7.7: Rewiring of the relays.

This way it is possible to control the conveyor by using just the PLC. In order to make the system work, it is necessary to establish a communication between the PLC and LH6, so the PLC can inform LH6 when the tray is ready for exchange, or LH6 can inform the PLC, when the tray has been exchanged. The communication issue is discussed in Section 7.3.2.

7.3.2 PLC-LH6 Communication

In order for LH6 to carry out its task properly, communication needs to be established between SBS and the PLC in the assembly module, so SBS can listen for status updates about the PCB tray. The PLC needs to receive a signal from SBS, when the refill is done, so it knows when to reel the tray in and resume regular work. This information exchange is achieved by establishing a TCP connection between the devices on the network.

Network Setup

The network setup in this project is illustrated in Figure 7.8. The computer is connected to the PLC by Ethernet cable and to LH6 devices through WiFi on a separate LAN. The computer serves as an interpreter and ensures that PLC can communicate with the rest of the devices. It was assumed that such an approach has prevailing simplicity. Both MiR100 and the PLC have fixed IP addresses that

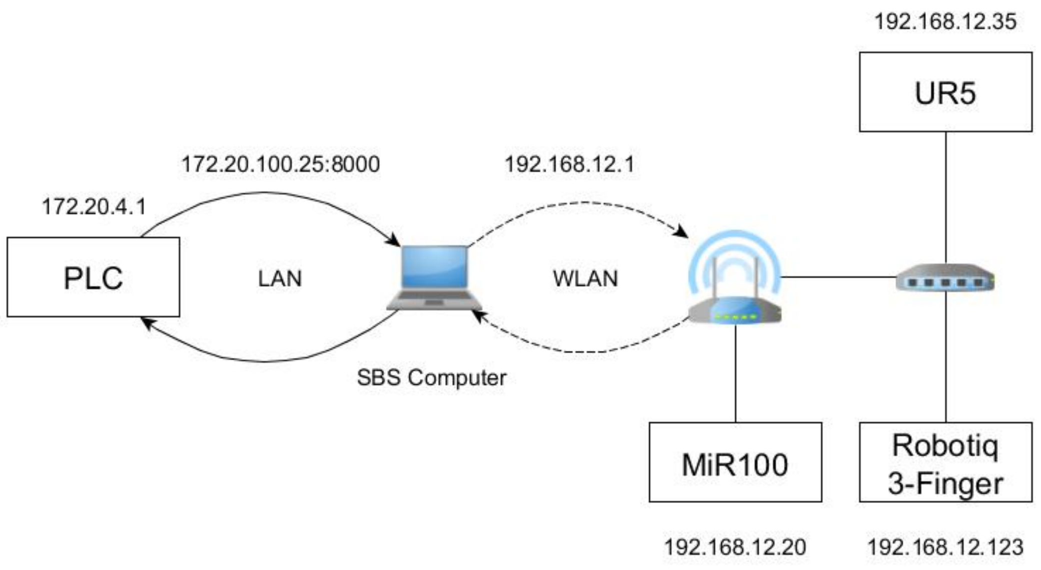


Figure 7.8: Network setup.

are complicated to change, which means that it is difficult to assign them to a same network. For this reason it has been decided to connect the computer to the PLC's and MiR100's network separately.

TCP Server and SBS Integration

The setup allows the PLC to communicate with the computer. This section, however, looks deeper into the communication methodology and discusses what needs to be done software-wise.

The default program of the PLC in the FESTO CP Factory comes with an installed library (Solution Center Library), which among other functionalities provides a function block with a set of methods that handle TCP network communication. Those tools simplify an implementation of TCP communication in a PLC program. However, in order to send a TCP signal to the SBS computer (see Figure 7.8), a TCP server needs to be created. Such server can then be integrated into a ROS node, which allows it to read from, and publish state messages to chosen topics, depending on outcomes of handled requests constantly incoming from the PLC. Those topics can then be read, or published to, by skills in SBS, enabling an event-based control of LH6 through SBS. The flow of information is illustrated in Figure 7.9.

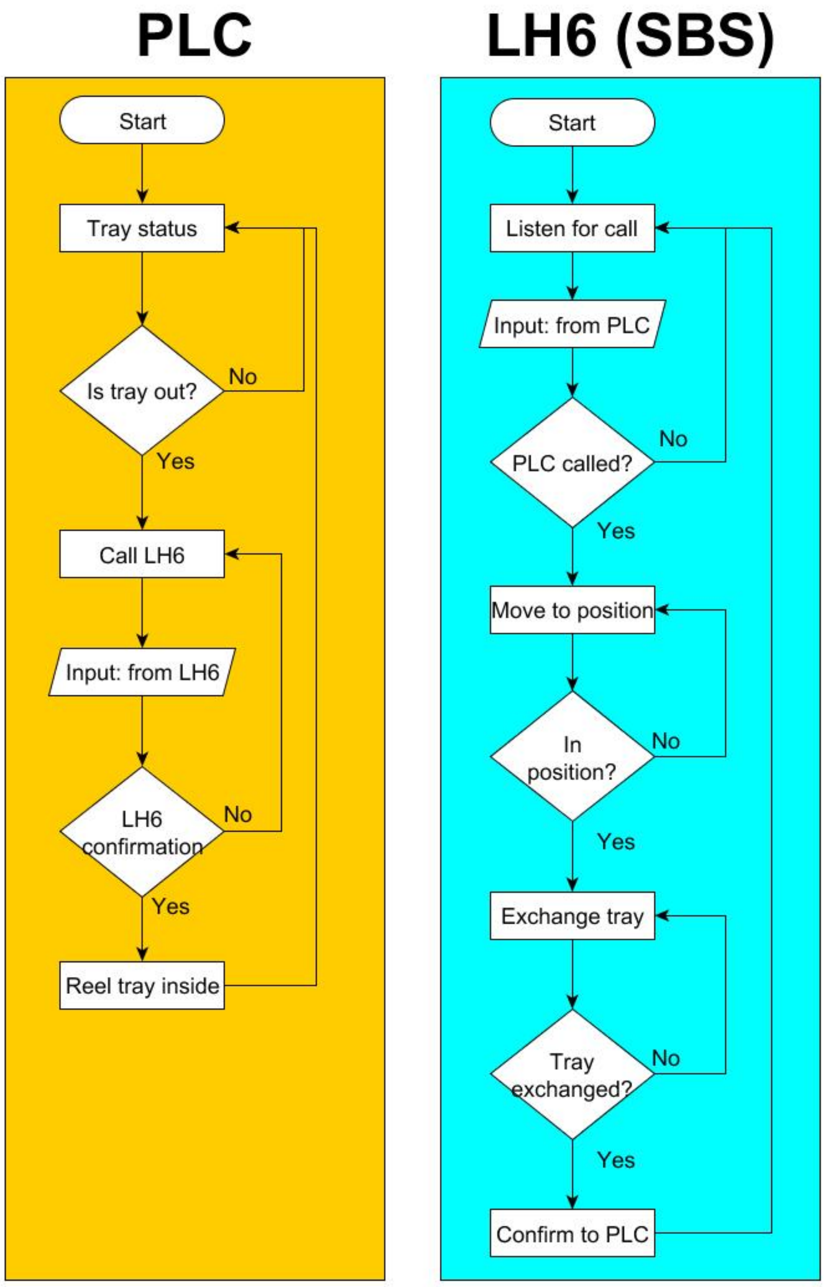


Figure 7.9: Flow of information.

7.4 Summary

This chapter consisted of a description of how new functionalities, hardware and different processes are integrated into SBS. To add a new Drive To functionality the MiR100 driver (see Section 6.1) was integrated into SBS. Furthermore, the 3D camera calibration driver (QR_pose, see Section 6.5) was also integrated in order to account for MiR100's imprecision. By implementation of functions, and using the theory from Section 6.5, the calibration was made possible. Lastly, to establish communication between the PLC and SBS, a ROS TCP driver was created. For an illustration of the new integrations of drivers used for this project, see Figure 7.10. The next Chapter 8, consists of a description of experiments made related to the integration, the main iterations leading to the physical design of LH6, the design of the MiR100 driver and a final test of the whole integrated system.

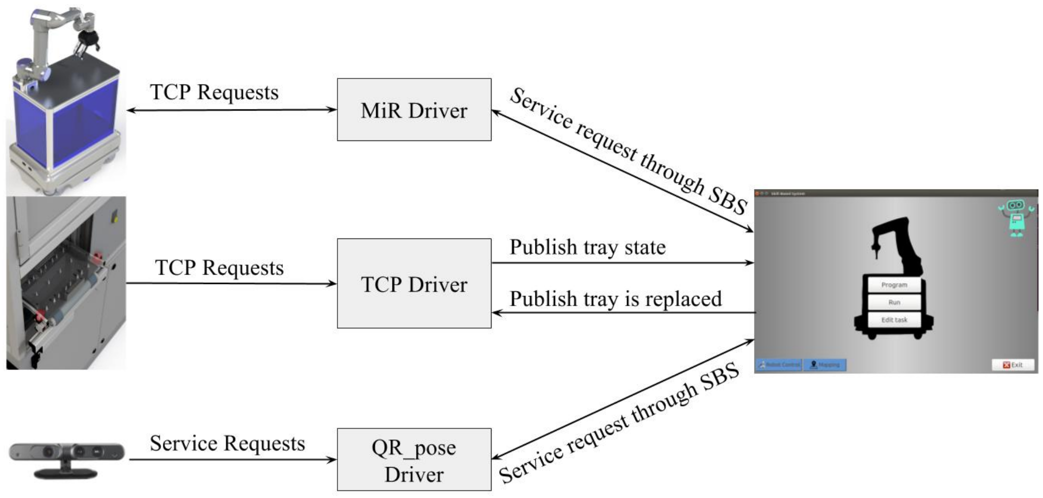


Figure 7.10: ROS drivers integrated in SBS in this project.

Chapter 8

Experiments and Results

This chapter contains a collection of experiment results for the main iterations done during this project. The Lean Startup Model was the method used to structure the work flow for this project. Every aspect of LH6 was designed in small iterations, thus developing on the knowledge gained from prior iterations. Since many iterations were small and quickly lead to new iterations, only the most important ones are described. It can be seen in Figure 8.1 how different interdisciplinary iterations were developed upon and combined into the LH6.

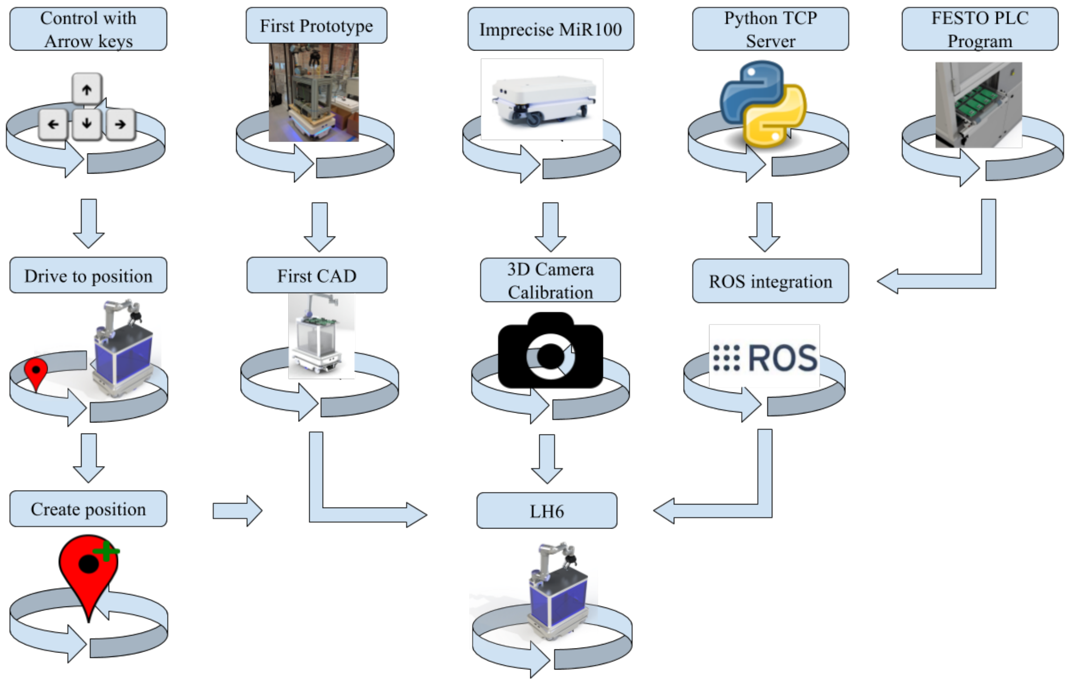


Figure 8.1: All main iterations done in this project.

8.1 Prototype and Construction

The design of the LH6 shown in Section 6.4 was reached by taking several iterations, which are described in this section.

8.1.1 Early Prototype of LH6

In order to investigate the capabilities of MiR100 and its mounting, a mock-up was made, which consisted of a wooden platform on top of MiR100. An alloy table with UR5 mounted on it was placed on top of the wooden platform (see Figure 8.2). The

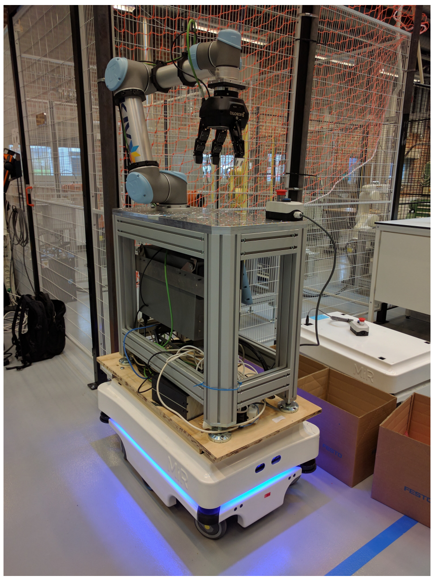


Figure 8.2: LH6 mock-up.

mock-up was used to test different types of scenarios like manipulation of the tray. It also laid foundations for the CAD models of the first and second prototype.

8.1.2 First CAD Model

The mock-up led to the initial work on the CAD model. The first iterations had the frame legs placed directly above the mounting holes on the MiR100. It also featured a raised UR5 mounting in the back center. The raised UR5 mounting was done so the UR5 would not hit anything on the tabletop while turning. The mounting with UR5 was placed in the rear part of the platform like in case of the other LH's. The tabletop was designed with a height of 975 mm above the ground, which put the tabletop in the same height as the conveyors of the FESTO CP Factory. Because MiR100 is smaller than the Neobotix platform used in the LH1 through LH3, the tabletop is also smaller. This led to the issue that it would not be possible to fit two

PCB trays on the tabletop. The placement of UR5 posed a problem that it was not easy for LH6 to reach inside the assembly module and reach the PCB tray, because of UR5's physical limitations. Finally, the mounting on MiR100 would be difficult to make since the screws would have had to be mounted directly into the legs of the frame. In Figure 8.3 the first CAD model is shown.

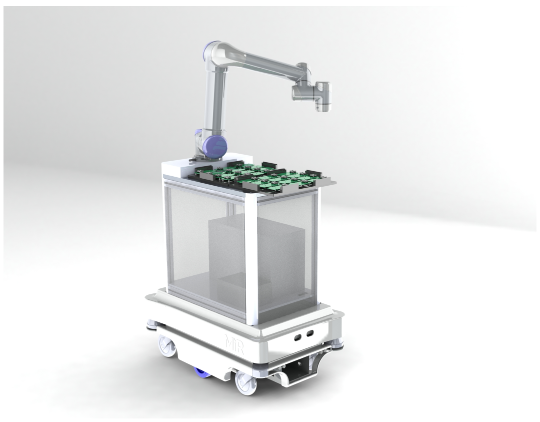
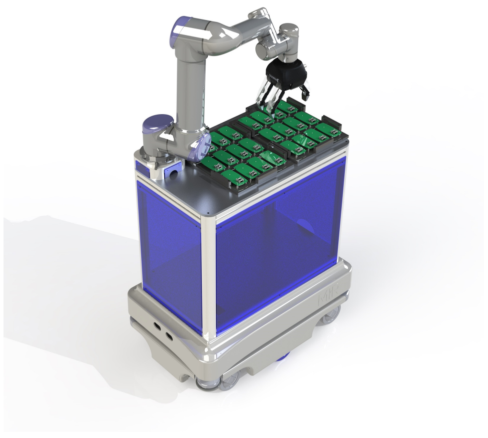


Figure 8.3: First CAD model of LH6.

8.1.3 Second CAD Model

Because of the difficulties faced in the first CAD iteration of LH6, a new CAD model was made from scratch. This iteration has extended bottom frames, thus the mountings are now put into the bottom frames and not into the legs. This also created an opportunity to make the tabletop larger and thus make space to fit two PCB trays. Although all prior LH's have the manipulator in the back, LH6 now has it in the front right corner. This design makes it easier for LH6 to reach the tray in the assembly module. The iteration also has a reworked UR5 platform. The first iteration had the raised platform filling the whole back. The new design has square shape and is big enough to fit the base of UR5, thus the most tabletop space is maintained. The alloy profiles to support the tabletop was changed from a 40x40 mm to a 40x80 mm to offer better support and add a possibility of mounting accessories on the sides, e.g. a Kinect camera. In Figure 8.4 the second CAD iteration is shown along with an exploded view of the frame.


(a) Second CAD iteration of LH6.

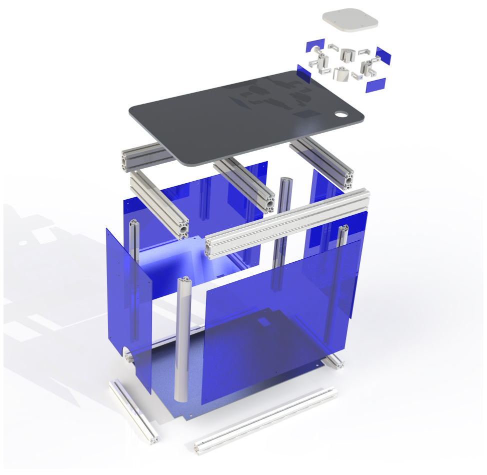

(b) Exploded view of the frame with UR5 stand.

Figure 8.4: Renders of LH6's second CAD and exploded assembly.

8.2 TCP Communication Experiments

Setting up the TCP communication described in Section 7.3.2 was an iterative process, consisting of three main iterations:

- Python TCP Server
- Integration into ROS Node
- Communication with SBS

Those iterations are separately described in this section with all their approaches and test results.

In the setup used for all iterations, the computer, which is supposed to communicate with PLC and run SBS with all the supplementary drivers, is connected via Ethernet cable to PLC and through WiFi to LH6.

8.2.1 Python TCP Server

First stage of the TCP server's development was to create it as a stand-alone program and have it to communicate with the PLC program. It has been chosen to write the server handler in Python, because Python by default offers modules containing server/client handling classes, which simplifies deployment of a server, especially if the request handling is expected to carry out a simple task, which is the case of this project. The TCP server has been designed to listen for requests on a specific IP address and port. The connection and its stability was tested by sending a single request from the PLC. A trial PLC program was made for this purpose utilizing carriers and available peripherals inside the assembly module (metal sensors and pneumatic carrier stoppers) in the following manner:

1. IF carrier reaches sensor at stopper THEN establish connection
2. Send message
3. Receive response
4. IF response is <specified condition> THEN let carrier through

During debugging, it has been discovered that separating receiving and sending in the PLC program into two functions results in unexpected behaviour, i.e. after establishing connection, sending a message, and resetting, if it is desired to establish connection again for reading purposes, the program gets stuck on the connection part. This problem does not occur if the sending and receiving is present in the same request, i.e.:

1. Establish connection
2. Send message
3. Wait until message was sent
4. Receive message
5. Reset connection

8.2.2 Integration into ROS Node

In the next stage, the working communication between the TCP server and PLC needed to be extended to components of LH6 (MiR100, UR5, Robotiq 3-Finger) via SBS. Because of this goal, the TCP server was created as a new ROS node in the LH6 workspace. After handling every single request, the server is able to read and publish to dedicated topics, that can be used to pass changes in state variables to SBS at a rate of 10 Hz.

A new PLC program was developed for the purpose of this project, which now uses the newly connected relays, described in Section 7.3.1, to control the tray in the following manner:

1. Wait for the tray
2. IF tray rolls out THEN call LH6

3. IF received message is <specified condition> THEN reel tray back in
4. reset program

Note, that the PLC program also constantly sends and receives TCP signals during each of the described steps.

8.2.3 Communication with SBS

Finally, SBS is required to interpret those signals in order to control the behaviour of LH6. This is done by creating two new simple skills (or rather functionalities) in SBS: Listening skill and Call skill. Those functionalities do not require any teaching, since there are no online parameters to be obtained. When executed, the Listening skill starts to constantly read from a topic, mapping the state of the tray. The skill does not finish until the right value, which announces the ejected tray, is read. The Call skill just simply publishes a value to a topic, made for LH6 state check, upon execution. When the value is published, the TCP server reads it and sends a TCP signal to the PLC, which after receiving the message reels the tray back in.

Functionality of the communication was tested by a simple trial task, created in SBS, which consisted of Listening skill, MoveTo skill, and Call skill. As expected, upon running the task the Listening skill started checking the specified topic. When the tray was ejected, the Listening skill exited and the MoveTo skill performed a movement pattern with UR5. After that, the Call skill was executed, which was noticed by the tray autonomously going back into the assembly module.

This last test has proven that the developed communication system is ready for deployment in the final test, where the part feeding is to be attempted.

8.3 3D Camera Calibration Experiment

This section describes the test of the LH6's manipulator's position before and after calibration in order to investigate if a calibration is necessary.

8.3.1 Calibration vs. Initial Precision

In order to investigate the effect of using the calibration method, three tests are created.

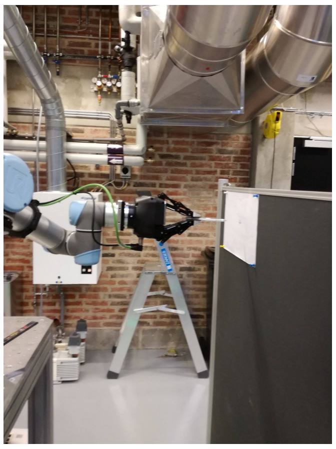
- Testing the precision of the LH6 without using camera calibration.
- Testing the precision of the LH6 using camera calibration (QR-code size = 183 mm).

- Testing the precision of the LH6 using camera calibration (QR-code size = 267 mm).

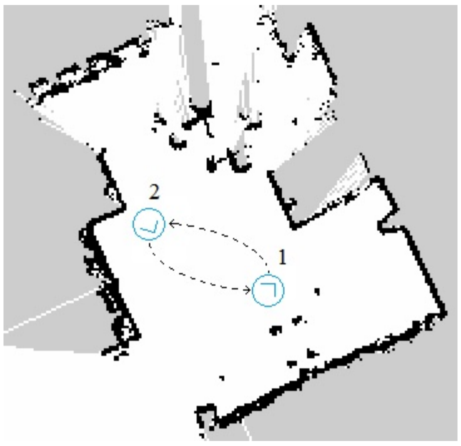
The tests achieve to visualise and investigate the effect of using the calibration method, over a non calibration, and furthermore, investigate which effect the size of QR-code has on the calibration accuracy. The test description is listed in the following description (Note that step 2 and 6 are only used in the tests using calibration):

1. LH6 is driven to a position from where it can be calibrated (position 2, see Figure 8.5a).
2. The QR calibration is done.
3. The UR5 is taught a position. The position is used a reference for the test (see Figure 8.5a).
4. LH6 is driven to a new position (position 1, see Figure 8.5b).
5. LH6 drives back to position 2.
6. The QR-code calibration is done.
7. The UR5 moves to the taught position (see Figure 8.5a).
8. The error, measured as deviation in the hight and the width from the reference position is measured and noted.
9. LH6 drives to position 1.
10. Step 5, 6, 7, 8 and 9 are repeated 10 times for each test.

As illustrated in Figure 8.6, the two tests, obtained using calibration, have a significantly lower error, than the non calibration test. The boxplots in Figure 8.6 illustrate that the error median for the non calibration test in the height is 22 mm and the width 45 mm, where the median in the tests using calibration is respectively 2 mm (QR size=183 mm) and 3 mm (QR size=264 mm) in height, and 1 mm (QR size=183 mm) and 2 mm (QR size=268 mm) in the width. From this it can be concluded that using the calibration method has a significant effect on the precision of UR5 and LH6's ability to interact with the environment, and that by using the method the accuracy tolerance of ± 10.8 mm is fulfilled (see Section 4.1). Looking at the non calibration test, it can be concluded that the precision of the MiR100 in general is too low to fulfil the accuracy tolerances. This indicates that in order to solve the part feeding task, a calibration method is needed for LH6. Looking



(a) Reference position.



(b) 1:Position 1, 2:position 2.

Figure 8.5: Camera calibration test setup.

at the different sizes of QR-codes, it appears as if the size of QR-code does not have a significant effect on the precision of the calibration. However, looking at the boxplots in Figure 8.6, it seems that the precision of the QR-code with a size of 183 mm has a tendency to have a marginally better precision than the QR-code with a size of 264 mm.

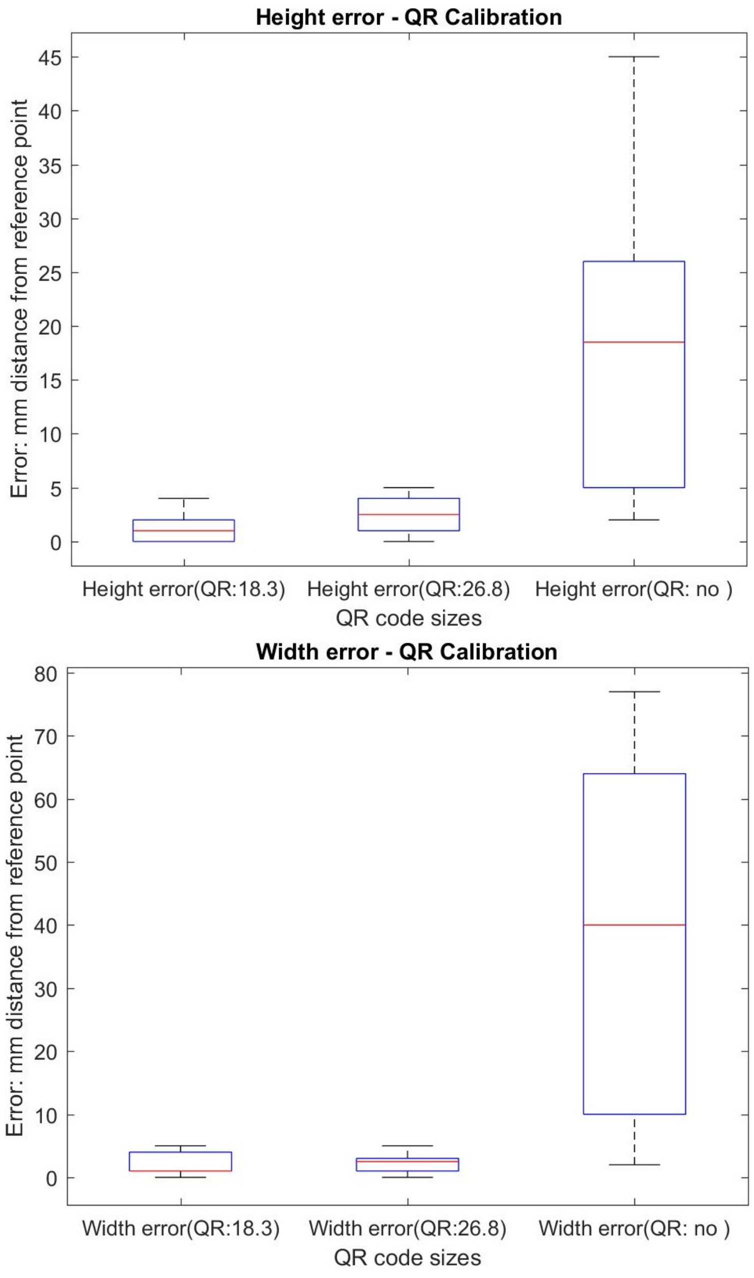


Figure 8.6: Boxplots illustrating the data obtained form the test. Note that the QR-code sizes are in centimetres.

8.4 MiR100 Driver

The MiR100 driver used in this project was developed as an iterative process that started with simple motions which later were upgraded or expanded to more complex motions such as Drive To capabilities. This section describes this process. The final driver includes multiple functionalities, each of which were separated program before being integrated.

8.4.1 Control with Arrow Keys

The very first program made to control MiR100, was done with the purpose of being able to control the platform with the arrow keys on a keyboard. It was done by using the Rosbridge to publish to a topic setting the command velocity when an arrow key was pressed. Both linear and angular velocity could be set.

8.4.2 Drive to Position

The next iteration was to be able to drive to positions created with the out-of-the-box interface MiR100 offers. This would create the opportunity to be able to drive autonomously to any given position and is thus considered high priority for this project. The first program to do this used the REST API on a port 8000 on the MiR100. A Python library created by MiR100 was used to handle the requests to and from MiR100. It turned out later that this method was out of date, and when the MiR100 software was updated to the latest version it did no longer function. Since the library was no longer fully functional, it was decided to move the Drive To program to a newer REST API located on port 8080. This created certain issues since there e.g. was not any ready-to-go library but only documentation of how to add a new mission, action and parameter without any examples. Using the Python request library, it was possible to request and post data to MiR100. In order to create a drive to mission, first a mission has to be created. That mission has to have, among other parameters, a GUID (globally unique ID). This GUID is used to identify the mission when it has to be executed. One of the other parameters of this mission is an action(s) which also has some parameters. In the case of Drive To, these parameters are the name of the position MiR100 will drive to, the number of retries that should be done if the path it finds is blocked and what the precision should be when arriving to the destination. When creating a drive to mission, a series of GUIDs were created and posted to MiR100 in the appropriate order at the appropriate address, in this case `http://mir.com:8080/v1.0.0/missions`. This will create a mission at the URL `http://mir.com:8080/v1.0.0/missions/mission_guid`. Now an action can be posted to the address `http://mir.com:8080/v1.0.0/missions/mission_guid/actions` and is accessible at the URL `http://mir.com:8080/v1.0.0/missions/mission_guid/actions/action_guid`. An example of how a mission can be created can be seen in Listing 8.1.

```
mission_guid = str(uuid.uuid1())

message = {"guid": mission_guid,
        "name": str("Taxa␣to␣" + position_name),
        "description": "Mission␣example",
        "read_only":0,
        "session_id": None,
        "show_in_app": 1,
        "hide_in_missionlist": 0,
        "delete_after_finish": 1}

mission_url = 'http://mir.com:8080/v1.0.0/missions'

response = requests.post(mission_url,
                         data=json.dumps(message),
                         headers=headers)
```

Listing 8.1: Example of creating a mission with Python

Note, that it is possible to set whether the mission should be deleted after finishing. This is convenient to avoid polluting the mission list with old missions. After all these steps the mission is ready to be executed. This is done by sending a post request to MiR100 containing the parameters (i.e. GUID), which specify the mission to be executed. Listing 8.2 shows how this is done.

```
mission_queue_url = 'http://mir.com:8080/v1.0.0/mission_queue'

msg = {'mission':mission_guid, 'taxi':None}

response = requests.post(mission_queue_url,
                         data=json.dumps(msg),
                         headers=headers)
```

Listing 8.2: Example of adding a mission to the mission queue with Python

Once all this was implemented, MiR100 could successfully drive between positions.

8.4.3 Create Position

The end goal with the driver is to make it usable in SBS which creates a program based on a teaching process. This means that the operator will physically move the robot to a location and store it. Instead of storing these locations in SBS alone, they can be stored on MiR100 and then referenced by SBS. The old REST API was only able to move existing positions created with MiR100 software. Therefore the process was to create a series of dummy positions that could be moved. This limited the amount of possible positions an operator could have and is thus not very flexible. Instead also this program was moved to the newer REST API which was capable of creating positions.

By using the new REST API it was possible to create a GUID and then post it to MiR100 along with other parameters (position name, coordinates, etc.) with the request library. These new positions can also be seen on the MiR webpage and MiR100 is able to reach them.

8.4.4 Integration Into a ROS Node

After all separate programs were tested successfully, they were put into a single function and copied into the same document name: `mir_driver.py`. The new main function creates a ROS node which creates a list of services as seen in Section 6.1.1. Each of these services are linked to a callback function responsible for executing the expected operation when a service is called. This could, for instance, be when the `mir_create_position` service is called, the function that contains the code for creating a position will be called.

8.4.5 SBS Mobile Platform

Since SBS does not have a simple way of communicating with a mobile platform, it needed to be integrated. The way SBS handles communication with drivers for manipulators and grippers is that it creates two motion primitive objects. One that consists of almost purely virtual functions called the MotionPrimitiveBase and one that inherits from this object. This allows them to be equal and the base object can now contain all information in objects that inherits from it. By doing this for the mobile platform as well, it is simple to add another mobile platform, like Neobotix, by creating an object that inherits from the base object and filling out the member functions. These member functions will then call the appropriate services from the MiR100 driver in order to handle the communication.

8.4.6 Drive To Skill

The Drive To skill is implemented like all other skills. When it is selected in the AddSkill window, it will set three offline user defined parameters and when it is teaching it will save the position coordinates and create a position on MiR100. When executing it will call the Drive To functionality of the MiR100 and drive to the position. A simple joystick was also added to the teaching interface so the operator could drive the mobile platform to a position before storing it.
To test if the Drive To skill worked, a small program was created in SBS. It contained a Drive To skill and a simple Move To skill that moves the manipulator. The Move To skill was added to ensure that the Drive To skill would not halt the program and visually show that the manipulator would not move before the MiR100 stopped. When teaching it was apparent that the teaching interface was shifting

downwards. This was due to some error when implementing the joystick. Due to time constraints this error could not be fixed. Even though the interface shifted, it was still possible to save the location, and when executing the program, MiR100 would drive to the taught position but the manipulator would also move. The reason for this was the after the MiR100 driver had created the mission to drive to the position and put it in the mission queue, it would then return and SBS would continue executing the next skill in the list. The fix to this error was to implement a loop that would block the MiR100 driver from returning until it had reached the end position. After this implementation, the skill worked as expected.

8.5 Final Test

The final test is carried out with the goal of solving the task described in Section 5. This section describes the test setup along with the results. Due to time limits the test is carried out with success/fail criteria.
At the time of the test, all the parts for the LH6 had not yet arrived. Therefore the test was carried out with the prototype (see Section 8.1). This serves as a proof of concept until the final LH6 has been assembled.

8.5.1 Test Setup

The test is carried out as two sequences that happen chronologically. In the first sequence the tray is taken out and placed on the LH6's table. In the second sequence the tray is picked up from the table and put back into the assembly module. Since the prototype does not have room enough for two trays on the table, the same tray is placed and picked up. The two sequences can be set up in the following list:

Sequence 1:

1. Go to start position.
2. Wait for signal.
3. Go to assembly module.
4. Move the manipulator to a general startposition in front of the tray output.
5. Calibrate the position according to a QR-code.
6. The gripper places its thumb at the edge of the tray (see Figure 8.7a).
7. Pull the tray out half way by using the thumb of the gripper (see Figure 8.7b).
8. Grasp the tray and pull it all the way out (see Figure 8.7c).

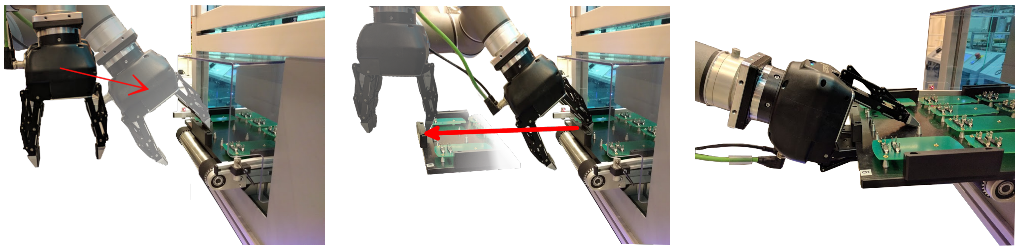

(a) Place thumb of gripper on the tray. (b) Pulling out the tray halfway out. (c) Grabbing the tray.

Figure 8.7: The process of pulling and grasping the tray.

9. Move the manipulator to be above the table of the LH.
10. Place the tray on the table (half way out the table side).
11. Push in the tray so it is placed at the center of the table.

Sequence 2:

12. Pull the tray half way off the table.
13. Grasp the tray and lift it off the table.
14. Move the manipulator to a general startposition in front of the tray input.
15. Calibrate the position according to a QR-code.
16. Put in the tray half way.
17. Push in the tray all the way.
18. Send signal to the assembly module that the tray has been replaced and the tray will be rolled in.
19. Move the manipulator to a safe position.

As mentioned in Section 7.3.1 the assembly module automatically pushes the tray out when it is empty. This is simulated in all tests by using the push button to pull out the tray manually.
The process is taught using SBS and the Drive To skill. Since the process is an advanced case of a pick and place it was hoped that the skills in SBS with the same name would suffice in making the sequences. This turned out not to be the case as they did not store enough via-points for the manipulator for either sequence to be carried out efficiently. Ideally an application-oriented skill for both the pick and place task would be created, but due to time limits, a series of MoveTo's was used

to create the program. Furthermore, the velocity in all MoveTo's was set to 30% of the maximum velocity, except when the tray was being put into the assembly module where it was set to 10%.
When trying to grasp the tray, it was discovered that for the gripper to have a firm grip around the tray, two of the PCB's needed to be removed. There is, however no PCB's on the tray when performing the tests, to avoid them being damaged.

The process takes place on a map created with the MiR web-interface. SBS is then used to teach the position. The map can be seen in Figure 8.8 along with the start position and the assembly module position.

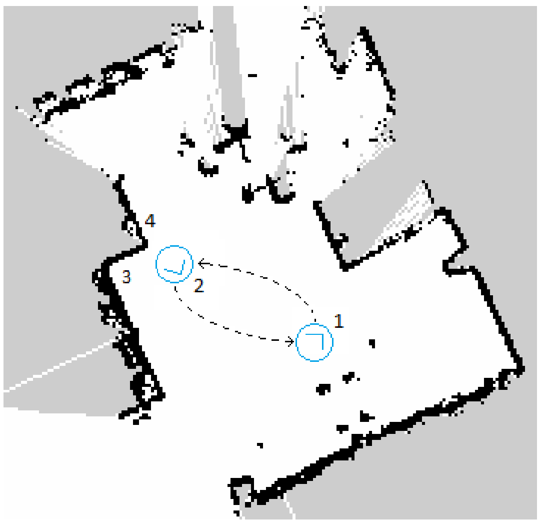


Figure 8.8: Map for replacing tray. 1) Start position, 2) Assembly module position, 3) QR-code, 4) Tray input/output.

After teaching the two sequences the program was executed.

8.5.2 Test Results

The test is carried out 25 times and is graded with a success/fail. In total 5 parameters are measured:

1. **Drive:** LH6 can drive and reach its target location.
2. **Calibrate:** The manipulator will compensate for the offset created by the mobile platform.
3. **Grasp:** The gripper can grasp the tray both when getting it from the assembly module or when picking it up from the LH6 table without losing it.

4. **Receive Signal (RS):** The LH6 will be notified when the tray has to be replaced.

5. **Send Signal (SS):** The LH6 will notify the assembly module when the tray has been replaced.

Each of these parameters is evaluated with a success/fail. If all of them succeed, the test run is considered successful. If one parameter failed the subsequent parameters is not be measured (i.e. when Drive fails it is not possible to put in the tray). The velocity scaling of the execution was increased over the course of the tests from 40% to 100% though this only affected the velocity of the manipulator and not the mobile platform. Table 8.1 shows the recorded results.
From Table 8.1 it should be noted that after test run 10, two small boxes, marked with green, was placed at the corner, marked with red, where the LH6 has its assembly module location in Figure 8.9. It was believed that the corner was hard for the mobile platform to see and thus resulted in it getting stuck in a loop of path planning and driving (without actually moving). In total the Mobile platform was the point of failure in 33.3% of the test runs (22.2% before adding the two boxes and 11.1% after).

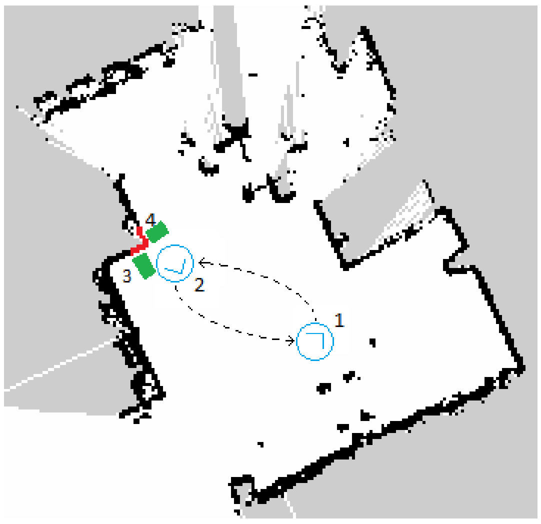


Figure 8.9: Updated test map.

As it also can be seen in Table 8.1 the driver for the PLC communication (both RS and SS) is responsible for a large margin of the errors. In total those two account for 67% of all the failed test runs.
The velocity did not have an impact on any of the parameters in the 25 test runs.

Number	Velocity(%)	RS	Drive	Calibrate	Grasp	SS	Total
1	40	✓	✓	✓	✓	✓	Success
2	40	✓	✓	✓	✓	✓	Success
3	40	✗	-	-	-	-	Fail
4	40	✓	✓	✓	✓	✓	Success
5	40	✓	✓	✓	✓	✓	Success
6	40	✓	✓	✓	✓	✓	Success
7	40	✗	-	-	-	-	Fail
8	40	✓	✗	-	-	-	Fail
9	40	✓	✗	-	-	-	Fail
10	40	✓	✓	✓	✓	✓	Success
11	40	✓	✓	✓	✓	✓	Success
12	70	✓	✓	✓	✓	✓	Success
13	100	✗	-	-	-	-	Fail
14	100	✓	✓	✓	✓	✓	Success
15	100	✓	✓	✓	✓	✓	Success
16	100	✓	✓	✓	✓	✓	Success
17	100	✗	-	-	-	-	Fail
18	100	✓	✓	✓	✓	✓	Success
19	100	✓	✓	✓	✓	✓	Success
20	100	✓	✓	✓	✓	✗	Fail
21	100	✓	✓	✓	✓	✓	Success
22	100	✓	✗	-	-	-	Fail
23	100	✓	✓	✓	✓	✓	Success
24	100	✓	✓	✓	✓	✗	Fail
25	100	✓	✓	✓	✓	✓	Success
Fails	-	4	3	0	0	2	9
Num. of measurements	-	25	21	18	18	18	25
Fail rate(%)	-	16	14.3	0	0	11.1	36

Table 8.1: Results from 25 test runs. Where ✓ is a success, ✗ is a fail and - is not recorded. Note that the fail rate does not include if a parameter prior failed.

Chapter 9

Discussion

To determine stability and consistency of the final solution, the LH6 was tested on its abillity to part feed the assembly module.

The tray exchange scenario was tested with a total of 25 attempts. 16 out of the 25 attempts were successful without any complications, which gives a raw estimated success rate of 64%. However, 5 out of the 9 failed attempts were caused by a discovered bug in the TCP driver, so it failed to pull in the tray if the SBS mission runs for a second time, i.e. the TCP driver and the PLC program needs to be restarted after every execution of the mission. The remaining four failed tests were caused by MiR100 encountering a navigation issue on the corner of the assembly module. Three of those occurred in the first run, after which the two boxes were placed in the environment. The second run was carried out with only one failure of the MiR100's navigation. It can be concluded that MiR100's navigation system is not perfect and placing of its via-points or design of surroundings can directly influence MiR100's success rate of the navigation task. If considering navigation in the mentioned 25 trials, MiR100 was able to successfully reach the destination in 21 of 25 cases (84%). The precision of handling the tray, was a concern, but with the implemented QR-code camera calibration algorithm, the variance of the tool's position was within the position tolerance of $\pm$ 10.8 mm, and there did not occur any error during part feeding in all the attempts were the manipulator got to carry out its task. However, as mentioned before in Section 8.5.1, all the attempts were executed with no PCB's on the tray in order to avoid them being damaged. The possible amount of PCB's on the tray was reduced from 12 to 10, because the gripper needed the space to have a firm grip. This means that the tray needs to be replaced more often if it was integrated in an actual production line.

The reason for the LH6 deployment in the specified use case was to replace manual part feeding of the FESTO CP Factory. During the final test LH6 was able to carry out the exchange in approx. 125 seconds with speed scaling set to 0.3 for most of the manipulator movements. It is not easy to quantify the necessary time to carry out the same task by manual labour, since there are many randomized factors present (initial position, motivation, human factor, etc.), however, considering the task alone, a human is believed to be faster or operate at the same speed even if the velocity of the manipulators movements were increased. But this consideration of the speed does not matter, since human operators have limited working hours and scheduled breaks are needed. With more robust implementation the LH6 would be able to operate consistently over extended amount of time, without any supervision, even overnight, needing only breaks for recharging.

Using SBS for programming tasks in the final test proved that utilizing skills for programming a task can be beneficial for user-friendliness, since it is possible to efficiently program on a higher level (drive to, move, pick, place, etc.), rather than creating the tasks using (low-level is this needed) commands (rotate joint(s)/rotate wheel(s) this much). Although in the final test for controlling the manipulator, only the MoveTo "skill" was used, which did not use SBS's potential to its fullest. It would be preferable to create an application-oriented skill tailored to the FESTO CP Factory use case for increased user-friendliness.

In order to create and utilize the Drive To skill in SBS, the MiR100 driver, described in Section 6.1.1, was used. In the given setup, the SBS and drivers run on a given computer, which runs the ROS core, while MiR100 operates in its dedicated ROS master. That is why the MiR100 driver utilizes Rosbridge and REST API communication to access the topics and services of the MiR100. Later on it has been discovered, that this could have been alternatively done by exporting environment variable of MiR100's ROS master to all terminal windows (similarly as described in camera setup in Appendix B) and thus running SBS and all drivers on MiR100's ROS master. Although having this knowledge when creating the MiR100 driver would not change anything, since this method would have certain disadvantages, like complicated setup and, if exported for all newly opened terminal windows, unavailability to run own ROS core on the computer. It has been utilized though for running the QR-code calibration driver on an extra laptop, placed on the LH6 and connected to the ROS master of the main computer, for convenience, since the computer running the calibration driver needs to be connected to the camera via USB. Moreover, the computer that runs the TCP driver needs to be connected to the PLC by Ethernet and also connected to the ROS master. In order to preserve mobility, those two drivers were ran on two dedicated computers. In an ideal case, all connections in the system would be wireless, which would allow all the drivers and SBS to run on one computer.

Chapter 10

Future Work

For further development of LH6, there are several things that would be beneficial to look into, which is a goal of this chapter.

This project focused on integration of the LH6 in only one of the three described part feeding challenges at the FESTO CP Factory. For future development a broader view could be looking into part feeding of all three parts. This could lead to several design changes such as the gripper and frame design.

To expand the capability of the LH6, an adjustable frame height was discussed. For this project it was discarded because of extra cost and only one part feeding task was set to be solved, which is why it was decided upon to make a fixed height frame. However, it would be a good feature to include since the concept of the LH's is to be flexible.

Since it is the first time the MiR100 is integrated in SBS, it first needed a driver and implementation in SBS. Thus it was only the MiR100's most important functionalities for this project, that were utilised. In future it could be investigated, if accessing MiR100's ROS master, and thus assuming full control of the MiR100, would be beneficial. This might give SBS the possibility to see how the MiR100 is running while driving and interrupt it if needed. It might also provide the feature of getting the signals from all the sensors and thus make a custom mapping if desired.

It would be beneficial to place a 3D camera just below the tabletop or on top of MiR100 pointing upwards. The reason behind this is, that at the moment, LH6 do not know how tall it is thus it does not see if it is driving under a table and the frame hitting it. Furthermore, an RGB camera could be mounted on the gripper in order to allow the possibility of inspection. This would not be possible in the current setup, since the payload of UR5 would be surpassed.

For LH6 to work autonomously in a real factory environment, it is not a good solution that a person has to plug it in and out of the power outlet. Instead it would be suggested that a charging dock is bought. (MiR-ApS 2016b) This would make LH6 able to autonomously move to and from the charging station and thus require less interference from employees.

If it were to do the part feeding of all parts, it should communicate with the MES instead. This would increase the flexibility by knowing what orders come through. LH6 would also know how many parts it fed and thus could predict when the different parts would run out and could therefore run more efficiently.

The final test was carried out with four tasks each containing several skills. Ideally this would not be the case, instead the four tasks should be application-oriented skills instead. These four skills could be:

1. Pick tray from assembly module.
2. Place tray on LH6.
3. Pick tray from LH6.
4. Place tray in assembly module.

In the case of skills 1 and 4, the movement of the manipulator is related to the placement of the assembly module, thus it needs to contain a QR-code calibration in the beginning of the skill's execution. Likewise, skill 2 and 3 needs to have the base of the manipulator set to default (0,0,0), so the movement is not done according to the QR-code. All of the four skills, should also contain all the movements to and from the tray including grasping and releasing it. Skill number 1 could have built in the feature for listening for the signal from FESTO CP Factory, which, when received, will make LH6 drive to the assembly module. While skill 4 should send a signal to FESTO CP Factory, that it should reel the tray in.
Doing this instead of the four tasks, would comply with the concept of object-centring and bring the programming level for the operator to a higher level of abstraction.

Safety design was not the main focus of this project, however, carrying out a risk assessment in the future would be recommended.

Chapter 11

Conclusion

The focus of this project was to automate the part feeding tasks of FESTO CP Factory specified in Section 2.2 by means of the AIMM called LH6. An analysis was done in order to identify part feeding and LH6 integration challenges, as well as investigate previous version of LH's which provided inspiration for integration challenges. The concept behind the SBS program was described, since it is used as a framework for LH6 and other LH's. The identified challenges and resources resulted in the final problem statement: *How can the part feeding tasks at the FESTO CP Factory line be solved by utilizing an AIMM with SBS integration?*. After identifying the goal in a problem formulation, design requirements could be set up. As a delimitation a specific part feeding challenge was selected to be solved. As a part of the delimitation a list of final requirements was established. This list can be seen in Table 11.1.

Requirement number 1 specifies the battery life of the LH6 design. This was not met since the final prototype was not built. Requirements number 2 was inspired by the other LH's, although it was not tested. Number 3 was set because of the limitations of the given hardware. It was vital to ensure that the created construction for LH6 and manipulated objects would comply these limits. The feeding point at the chosen FESTO CP module has certain dimensional limitations, which was identified as a challenge in case of imprecise positioning. This fact was the reason for establishing requirement number 4. It was concluded that doing the calibration test significantly improves the general precision, which then fulfils the set precision requirement. Furthermore, the accuracy of tray handling in the final test never caused a trial failure. Requirement number 5 was partly established by given hardware, but also in order for LH6 not to waste its payload on a compressor for a pneumatic gripper. Since LH6 did not receive a battery, requirement number 6 was not met as well. LH6 framework was chosen to be SBS, since it was

Item	Description	Value	Unit	Success
1	Battery life	3.5 - 1 - 3.5	Hours	✗
2	LH6 payload	20	kg	?
3	Manipulator payload	5	kg	✓
4	Manipulator precision	$\pm$10.8	mm	✓
5	Electric gripper	-	-	✓
6	Common power source	-	-	✗
7	Programmable in SBS	-	-	✓
8	Drive Functionality	-	-	✓
9	Common emergency buttons	-	-	✗
10	Restrict manipulator movement when mobile platform moves	-	-	✓
11	Restrict mobile platform movement when manipulator moves	-	-	✓
12	Communication with FESTO CP module PLC	-	-	✓

Table 11.1: Review of final requirements.

successfully integrated with previous LH's. For the mobile platform a Drive functionality was needed to be integrated since no such functionality was developed for SBS, thus requirement number 8 was met. For safety purposes, requirement 9 was established, since it is desirable to have a common emergency button which cuts power to all components included in LH6. In relation to requirements number 1 and 6, the final prototype was not built, thus the electric circuit was not made. The mobile platform should not move while the manipulator is moving and vice versa. This gave requirements number 10 and 11, which were met. Lastly to carry out integration of LH6 with FESTO CP Factory a communication was needed to be established so LH6 knows when it is necessary to carry out the part feeding task. This has been specified as requirement number 12 and has been achieved by creating a driver which allows exchange of TCP/IP signals.
For overview of the accomplished requirements, see Table 11.1.

With the developed prototype, it was possible to carry out the final test, described in Section 8.5 with a fail rate of 36%, which was contributed to by navigation errors and a communication bug. In relation to the final problem statement, results of the test show that LH6 had difficulties navigating in the environment it was integrated in, however, the test showed that LH6 could potentially be used as AIMM dedicated to part feeding in an Industry 4.0 environment.

Bibliography

Andersen, Rasmus S. et al. (2013). "Fast Calibration of Industrial Mobile Robots to Workstations using QR Codes". In:

Bøgh, Simon et al. (2008). "Fremtidens produktionsmedarbejder - udvikling af mobilrobotten; "Lille Hjælper"". MA thesis. Aalborg Univiversity.

Bøgh, Simon et al. (2012). *Does your Robot have Skills?* Tech. rep. Aalborg University. URL: `http://vbn.aau.dk/files/69945674/ISR_2012_Paper_ID_1158_FINAL.pdf`.

Bonev, Ilian (2014). *Should we fence the arms of Universal Robots?* Control and Robotics Laboratory. URL: `http://coro.etsmtl.ca/blog/?p=299`.

Carøe, Christian, Mikkel Hvilshøj, and Casper Schou (2012). "Intuitive Programming of AIMM Robot". MA thesis. Aalborg University.

Cisco (2016). *Quick Start Guide*. URL: `http://www.cisco.com/c/dam/en/us/td/docs/switches/lan/csbus/sd208/quick_start/guide/SD208QuickStart.pdf`.

Corke, Peter (2013). *Robotics, Vison and Control*. Second. Springer.

Etzerodt, Rune, Morten Palmelund-Jensen, and Casper Abildgaard Pedersen (2013). "Creation of an Autonomous Industrial Mobile Manipulator". MA thesis. Aalborg University.

Kalpakjian, Serope and Steven R. Schmid (2014). *Manufacturing Engineering and Technology*. Ed. by Esther Yap. 7th. Pearson.

Madsen, Ole (2016). "AAU Smart Production". In:

MiR-ApS (2016a). *MiR100: Data sheet*. URL: `http://mobile-industrial-robots.com/wp-content/uploads/2016/05/2016-11-11_MiR100-EN-11-20161.pdf`.

— (2016b). *MiR100 Docking Station – Data sheet*. webpage. URL: `http://mobile-industrial-robots.com/wp-content/uploads/2016/05/MiR100-Docking-Station-EN-05-2016.pdf`.

— (2016c). *Technical Discription*. Webpage. URL: `http://mobile-industrial-robots.com/da/mir100/teknisk-materiale/`.

Pedersen, Charlotte (2015). *Fornem pris til fynsk robot*. webpage. URL: `http://www.fyens.dk/erhverv/Fornem-pris-til-fynsk-robot/artikel/2791820`.

Pedersen, Mikkel Rath (2011). "Integration of the KUKA Light Weight Robot in a mobile manipulator". MA thesis. Aalborg University.

Pedersen, Mikkel Rath et al. (2016). "Robot skills for manufacturing: From concept to industrial deployment". In: *Robotics and Computer-Integrated Manufacturing* 37, pp. 282–291.

Robotics and Automation Group (2016). *Home Page*. Aalborg University. URL: `http://robotics-automation.aau.dk/`.

Robotiq (2016a). *ADAPTIVE ROBOT GRIPPER*. Webpage. URL: `http://robotiq.com/products/industrial-robot-hand/`.

— (2016b). *Robotiq 3-Finger Adaptive Gripper Specifications*. URL: `http://robotiq.com/wp-content/uploads/2014/08/Robotiq-3-Finger-Adaptive-Gripper-Specifications-ES.pdf`.

Rüssmann, Micheal et al. (2015). "Industrial 4.0 - The Future of Productivity and Growth in Manufacturing Industries". In: *BCG*. URL: `http://www.zvw.de/media.media.72e472fb-1698-4a15-8858-344351c8902f.original.pdf`.

Universal-Robots-A/S (2016a). *Ethernet Socket Communication via URScript*. URL: `https://www.universal-robots.com/how-tos-and-faqs/how-to/ur-how-tos/ethernet-socket-communication-via-urscript-15678/`.

— (2016b). *UR5*. URL: `https://www.universal-robots.com/media/1003684/ur5-a-highly-flexible-robot-arm-big.png`.

— (2016c). *UR5 - Technical details*. URL: `https://www.universal-robots.com/media/1514597/101081_199901_ur5_technical_details_web_a4_art03_rls_eng.pdf`.

— (2016d). *UR5 Technical specifications*. URL: `https://www.universal-robots.com/media/50588/ur5_en.pdf`.

Appendix A

Budget

This appendix contains the detailed budget list for LH6. All prices are excluding VAT (besides shipment fees) and is the price for the university. Prices marked with a * is an estimated price.

Table A.1 shows the price paid for the frame including alloy profiles and other parts. The alloy plates where machined at Aalborg University and thus is an estimation.

Table A.2 lists the expenses for parts inside LH6. This includes an extra battery, emergency button etc. It should be noted, that parts for the electric circuit did not make it in time, thus the price is not available.

Table A.3, shows the price for the robots and the gripper. The price for the UR5 and Robotiq 3-Finger gripper is from (Etzerodt, Palmelund-Jensen, and Pedersen 2013). The University got at a discount on the MiR100; the price of the MiR100 is normally in the range of 180 000 DKK. (Pedersen 2015)

The full estimated budget list of LH6 is shown in Table A.4.

Item	**Describtion**	**QTY**	**Price**
Profile 40x40 natur	Alloy profile	2.10 m	355.40 DKK
Profile 40x80 natur	Alloy profile	2.47 m	744.89 DKK
Profile 40x40 R90 natur	Alloy profile rounded	2.51 m	398.66 DKK
Machining	Machining of central fasteners	28	713.16 DKK
Cuts	Cuts of alloy profiles	17	338.77 DKK
Drilling	Drilling of mounting holes to MiR100	4	117.24 DKK
Mounting angle 40x20x15	Mountings angle for acrylic glass	8	291.30 DKK
Central Fastener S		28	724.29 DKK
Nut stone M5	Nut stone for mounting	32	193.20 DKK
Shipment	Shipment by GLS	1	189 DKK
Acrylic Glass	Acrylic glass in total 8 plates with different cuts	1	945 DKK
Shipment	Shipment by GLS	1	69 DKK
Alloy Plate	Alloy plate 500x800x10	2	800 DKK*
Alloy Plate	Alloy plate 200x200x10	1	400 DKK*
SUM			**6 279.91DKK**

Table A.1: Budget list for the frame.

Item	**Describtion**	**QTY**	**Price**
Emergency Button	Emergency button from Eaton	2	1 259 DKK
WICA Li-NMC 39Ah Battery	Extra battery for MiR100	1	4 000 DKK
Kinect Camera	Kinect Camera from ASUS	1	900 DKK*
Electrician Components	Converters, circuit breaker ect.	N/A	N/A
SUM			**6 159 DKK**

Table A.2: Budget list for internal components.

Item	**Describtion**	**QTY**	**Price**
MiR100	Mobile platform from MiR	1	180 000 DKK
UR5	Manipulator from UR	1	170 291 DKK
Robotiq 3-Finger Gripper	Gripper from Robotiq	1	128 471 DKK
SUM			**478 762 DKK**

Table A.3: Budget list for robots and gripper.

Item	**Price**
Frame	6 279.91 DKK
Internal Components	6 159 DKK
Robots	478 762 DKK
SUM	**491 200.91 DKK**

Table A.4: Summary budget list of LH6.

Appendix B

Setup Guide for Running SBS and LH6

Before running SBS together with LH6 there are several programs and drivers which needs to be started. This appendix consists of a guide explaining which programs and drivers are needed in order to run SBS, together with LH6 and how to start these drivers. To follow the guide, the folder found in the DropBox link (`https://www.dropbox.com/sh/easiqhiqohkqfq7/AABjUeWcLpskfj-iLzxyMaX-a?dl=0` or in Figure B.1) and ROS indigo are expected to be installed.

In order to run SBS with LH6 the following software dependencies are required:

Figure B.1: Link to project repository.

1. Openni2_launch
2. Zbar
3. SBS

The following list, describe the different needed to set up LH6:

1. Turn on MiR100, UR5 and Robotiq 3-Finger

2. **Connect to the MiR100 WiFi and start roscore:**
   Connect the computer running SBS to the WiFi of MiR100.

3. **Run the drivers for MiR100, UR5 and Robotiq 3-Finger:**
   To start the drivers execute the following commands in separate terminals:

   - To execute the driver for MiR100 run:
     rosrun mir mir_driver.py
   - To execute the driver for UR5 run:
     roslaunch ur_bringup ur5 _and_proxy_bringup.launch
   - To execute the driver for Roboticq 3-Finger run the following into separate terminals:
     rosrun robotiq_s_model_control SModelTcpNode.py 192.168.12.123
     rosrun rq3_proxy rq3_proxy_v3.py

4. **Connect and setup the camera on the computer placed on LH6:**

   - Open a terminal and write the following(Here the IP address of the computer running SBS is needed):
     export ROS_MASTER_URI=http://192.168.12.xx:11311
   - Connect to the camera and run the following in the same terminal:
     rosrun openni2_launch openni2.launch
   - Open a new teminal and run the following again:
     export ROS_MASTER_URI=http://192.168.12.xx:11311
   - In the same terminal run:
     rosrun QR_pose QR_pose.py

5. **Establish communication with the assembly module** :

   - Connect to the computer running SBS with the assembly module through Ethernet, while still being connected to MiR100 WiFi.
   - Run the communication TCP driver:
     rosrun tcp_driver plc_communicate.py
   - Run the PLC program on the assembly module.

6. **Run SBS** :
   Run the following:
   rosrun Skill_based_system SBS -r LH6

To execute the SBS without running the drivers run the following:
rosrun Skill_based_system SBS -r SIM