

# Automating a robot cell welding process



---

**Elias Ljunglöv**

Division of Industrial Electrical Engineering and Automation  
Faculty of Engineering, Lund University



# Automating a robot cell welding process

Control and monitoring of an automated stud bolt welding process at  
SWEP International AB



**LUNDS**  
**UNIVERSITET**  
Lunds Tekniska Högskola

**LTH School of Engineering at Campus Helsingborg**  
**Department of Industrial Electrical Engineering and Automation**

Bachelor thesis:  
Elias Ljunglöf

© Copyright Elias Jens Ljunglöf

LTH School of Engineering  
Lund University  
Box 882  
SE-251 08 Helsingborg  
Sweden

LTH Ingenjörshögskolan vid Campus Helsingborg  
Lunds universitet  
Box 882  
251 08 Helsingborg

Printed in Sweden  
Lunds universitet  
Lund 2022

## Abstract

Today, the importance of sustainable and effective energy usage is rapidly growing. SWEP International AB leads the manufacturing of brazed plate heat exchangers that offer effective heating and cooling applications used in a wide range of systems and industries. In their manufacturing process, stud bolts are manually welded onto the heat exchanger surface. A project to fully automate this process using an industrial robot has already begun at SWEP to increase the process productivity, quality, and repeatability. First, the robot uses an attached touch probe to measure points on the surface and sides of a heat exchanger to calculate a center reference point and account for any inclination. Then, the robot uses an attached weld gun to weld stud bolts onto the surface. The purpose of this bachelor thesis was to continue their project of creating a fully automated solution that will fully replace the manual bolt welding station in the future.

The first objective was to study their existing semi-automated solution and perform quality analysis by testing different amounts of current, weld time duration, and shielding gas flow. A bend test, torque test, and visual inspection were then performed to understand how each weld parameter affects the weld result. From this, a suitable set of parameter values were determined.

The positional accuracy of two different methods for calculating the center reference point was then measured. In these accuracy tests, it was seen that the tolerances were not being met. Improvements were made to both methods to increase their accuracy. Lastly, it was concluded which method should be used in the automated welding process.

The welding unit that supplies the current and gas flow was then set up to measure and monitor the welding parameters to create a fail-safe system that stops the process if any parameter is measured outside its tolerance.

A control structure was then implemented to operate the original robot program via PROFINET (industry standard for data communication over Industrial Ethernet). A programmable logic controller (PLC) and Human-Machine-Interface (HMI) were programmed in Siemens TIA Portal, and additions were made to the robot program in ABB RobotStudio. This created an automated solution for the bolt welding process where the HMI can be used to create a work order, create different bolt configuration recipes, control the program flow, and monitor the process.

Finally, ideas were presented on how SWEP can further improve the positional accuracy, as well as ideas for future work to create a fully automated solution that is ready to replace the manual welding process used today in their production process.

**Keywords:** stud bolt welding quality, accuracy in automated welds, welding unit fail-safe, brazed plate heat exchangers, programmable logic controller, human-machine interface.

## Sammanfattning

Idag fortsätter vikten av hållbar och effektiv energianvändning att växa. SWEP International AB leder utvecklingen och tillverkningen av lödda plattvärmeväxlare, som erbjuder en av de mest effektiva lösningarna för värme- och kylapplikationer till en bred mängd av olika system och industrier. I deras tillverkningsprocess svetsas bultar fast manuellt på värmeväxlarens yta och ett projekt med mål att helt automatisera denna process med hjälp av en industrirobot har påbörjats på SWEP, för ökad produktivitet, kvalitet och repeterbarhet. Denna robot använder först en kontaktbaserad mätprob för att mäta punkter på ytan samt sidorna av en värmeväxlare. Detta görs för att beräkna värmeväxlarens mittpunkt med hänsyn till eventuell lutning av värmeväxlaren eller dess underlag. Roboten använder sedan en ansluten svetspistol för att svetsa fast bultar på värmeväxlarens yta. Syftet med detta examensarbete var att fortsätta arbetet i deras projekt med att skapa en helautomatiserad lösning som i framtiden kan ersätta den manuella svetsprocessen.

Det första uppdraget var att studera deras befintliga halvautomatiska lösning samt utföra kvalitetsanalys på denna lösning. Denna analys genomfördes genom att testa olika värden av ström, svetstid och gasflöde. Sedan utfördes ett böjtest, vridmomenttest och visuell inspektion för att förstå hur varje parameter påverkar svetsresultatet. Utifrån detta bestämdes en lämplig uppsättning av parametervärden.

Därefter mättes noggrannheten i positioneringen av bultarna, där det fanns två olika algoritmer som beräknar mittpunkten av värmeväxlaren. I dessa noggrannhetstester konstaterades det att toleranserna inte var uppfyllda. Förbättringar gjordes till de två algoritmerna för att öka deras noggrannhet. Slutligen bestämdes vilken av de två metoderna som bör användas i den automatiserade svetsprocessen.

Svetsenheten som levererar ström och gasflöde till processen konfigurerades sedan till att mäta och övervaka dess svetsparametrar för att skapa ett säkert system som stoppar processen om någon parameter blir uppmätt utanför dess tolerans.

En kontrollstruktur implementerades sedan till att styra det ursprungliga robotprogrammet via PROFINET (branschstandard för datakommunikation över industriell Ethernet). Ett programmerbart styrsystem (PLC) och ett människa-maskin-gränssnitt (HMI) programmerades i Siemens TIA Portal och robotprogrammet i ABB RobotStudio vidareutvecklades. Detta skapade en automatiserad lösning för bultsvetsprocessen där gränssnittet kan användas för att initiera en ny arbetsorder, skapa olika recept med bultkonfigurationer, styra programflödet och övervaka processen.

Slutligen presenterades idéer om hur SWEP ytterligare kan förbättra positionsnoggrannheten, samt idéer för framtida arbete med att skapa en helautomatiserad lösning som är redo att ersätta den manuella svetsprocess som används idag i deras produktion.

**Nyckelord:** kvalitet av bultsvets, noggrannhet i automatiserad svetsning, säker drift av svetsenhet, lödda plattvärmeväxlare, programmerbart styrsystem, människa-maskin-gränssnitt.

## **Acknowledgements**

I want to take the opportunity to thank my supervisors, Anders Robertsson at the Department of Automatic Control at Lund University, and Qian Zhao, Automation Specialist at SWEP, for the excellent assistance, teachings, and motivation that they have given me throughout the course of this project.

I would also like to give a special thanks to Lennie Liegnell, Global Engineering Manager at SWEP, for giving me the opportunity to make this thesis possible. During these 15 weeks, I have learned a tremendous amount about working as an engineer. It has only strengthened my enthusiasm to continue learning and to work in automation and automatic control in the future.





# List of contents

<b>1 Introduction .....</b>	<b>1</b>
<b>1.1 Background .....</b>	<b>1</b>
1.1.1 The company.....	1
1.1.2 Brazed plate heat exchangers.....	2
1.1.3 The existing manual welding process.....	3
<b>1.2 Purpose .....</b>	<b>4</b>
<b>1.3 The robot cell .....</b>	<b>4</b>
1.3.1 Overview of the cell.....	5
1.3.2 The workstation .....	5
1.3.3 Welding gun and measuring probe .....	6
1.3.4 The ABB industrial robot .....	7
1.3.5 IRC5 robot controller.....	7
1.3.6 The FlexPendant interface .....	8
1.3.7 The HBS Visar 1200 welding unit.....	8
1.3.8 Grounding the heat exchanger .....	9
1.3.9 The original welding process algorithms.....	9
<b>1.4 Objectives .....</b>	<b>14</b>
1.4.1 Perform quality analysis of the automated welding process.....	14
1.4.2 Evaluate and improve the program algorithm .....	14
1.4.3 Implement a control structure for a fully automated solution .....	14
1.4.4 Enable a fail-safe system in the HBS welding unit .....	14
1.4.5 Design and implement a Human-Machine-Interface.....	15
<b>1.5 Problem formulation .....</b>	<b>15</b>
<b>1.6 Motivation .....</b>	<b>15</b>
<b>1.7 Boundaries.....</b>	<b>16</b>
<b>2 ABB RobotStudio and Siemens TIA Portal.....</b>	<b>17</b>
<b>2.1 ABB RobotStudio .....</b>	<b>17</b>
2.1.1 The simulation environment .....	17
2.1.2 The RAPID programming language .....	18
<b>2.2 Siemens TIA Portal .....</b>	<b>19</b>
2.2.1 The development environment .....	19
2.2.2 Programming a PLC .....	20
2.2.3 Creating an HMI layout .....	22
<b>3 Method.....</b>	<b>23</b>
<b>3.1 The thesis work .....</b>	<b>23</b>
<b>3.2 Methodology .....</b>	<b>24</b>
<b>4 Analysis.....</b>	<b>26</b>
<b>4.1 Performing quality analysis of the automated process .....</b>	<b>26</b>
4.1.1 Performing the tests .....	26
4.1.2 Visual and physical inspection .....	27

<b>4.2 Extending the original program algorithms .....</b>	<b>28</b>
4.2.1 Studying the original program algorithms .....	28
4.2.2 Detailed flow chart.....	29
4.2.3 Measuring the accuracy of the two methods .....	29
<b>4.3 Creating the control structure.....</b>	<b>30</b>
4.3.1 Using a Siemens S7-1200 PLC as a master controller .....	30
4.3.2 Setting up the PROFINET .....	31
4.3.3 Importing the welding parameters to the PLC.....	31
4.3.4 Giving the PLC complete control of the weld power .....	32
4.3.5 Programming the PLC to control an entire work order .....	32
<b>4.4 Enabling a fail-safe system for the welding unit.....</b>	<b>35</b>
4.4.1 Studying the Visar 1200 manual.....	35
4.4.2 Creating the welding program .....	36
<b>4.5 Designing and programming the HMI .....</b>	<b>36</b>
4.5.1 Defining the needed features .....	36
4.5.2 Designing the HMI .....	37
4.5.3 Programming the HMI in TIA Portal .....	38
4.5.4 Additions made to the RAPID program code.....	39
<b>5 Results.....</b>	<b>42</b>
<b>5.1 Welding parameters for optimal weld quality .....</b>	<b>42</b>
<b>5.2 Accuracy of the two methods of calculating the center.....</b>	<b>43</b>
5.2.1 Accuracy of the two methods .....	43
5.2.2 Accuracy of the two methods, after improvement attempt.....	45
<b>5.3 The final HMI and automated solution .....</b>	<b>46</b>
5.3.1 The "Root screen" .....	46
5.3.2 The "Operator view" .....	47
5.3.3 The "Recipe view" .....	47
5.3.4 The "Settings" screen.....	48
5.3.5 The "Monitoring" screen.....	48
5.3.6 The "Alarms" view .....	48
<b>6 Conclusions .....</b>	<b>50</b>
<b>6.1 The program algorithms and center calculation .....</b>	<b>50</b>
6.1.1 The best method for calculating the center .....	50
6.1.2 Future work.....	50
<b>6.2 Fully automating the bolt welding process.....</b>	<b>50</b>
<b>6.3 An ethical reflection of automation – is it beneficial to society? .....</b>	<b>51</b>
<b>7 Terminology .....</b>	<b>52</b>
<b>8 References .....</b>	<b>53</b>
<b>Appendix A: Flow chart of the RAPID program .....</b>	<b>54</b>
<b>Appendix B: State diagram of the PLC program .....</b>	<b>56</b>
<b>Appendix C: PLC program code.....</b>	<b>57</b>

## List of figures

Figure 1: SWEP brazed plate heat exchangers. Image from [1].	1
Figure 2: The inner parts of a BPHE. Image from [2].	2
Figure 3: A BPHE used as an evaporator. Image from [2].	2
Figure 4: The manual bolt welding workstation	3
Figure 5: The steel template used to position bolts	4
Figure 6: Example of a bolt configuration	4
Figure 7: The manual weld gun used	4
Figure 8: Example of a successful bolt weld.	4
Figure 9: Overview of the robot cell	5
Figure 10: 3D model of the bolt tray	5
Figure 11: 3D model of the heat exchanger fixture	5
Figure 12: Workstation setup with bolt tray and fixture	6
Figure 13: 3D model of the weld gun	6
Figure 14: The touch probe. Image from [4].	6
Figure 15: The industrial robot used in the welding process	7
Figure 16: The IRC5 robot controller cabinet	8
Figure 17: The HBS welding unit. Image from [6].	9
Figure 18: The FlexPendant operator unit.	9
Figure 19: The welding program starting conditions	10
Figure 20: Points measured to calculate the heat exchanger's center reference point and orientation, Method I.	11
Figure 21: A side point being approached by the probe	11
Figure 22: The measured points saved to the controller	12
Figure 23: Calculation of the center point, Method I.	13
Figure 24: Calculation of the center point, Method II.	13
Figure 25: Welding a bolt to the heat exchanger	13
Figure 26: The RobotStudio simulation environment	17
Figure 27: TIA Portal with simulated PLC and HMI.	20
Figure 28: TIA Portal – the four types of program blocks	21
Figure 29: TIA Portal - PLC tags	21
Figure 30: TIA Portal - HMI development	22
Figure 31: The prepared thesis work schedule	23
Figure 32: The weld quality tests, varying the weld current	26
Figure 33: The weld quality tests, varying the weld time	27
Figure 34: The weld quality tests, varying the gas flow	27
Figure 35: Torque wrench used in the physical tests	28
Figure 36: Bend equipment used in the physical tests	28
Figure 37: Inclination of a heat exchanger being accounted for	28
Figure 38: The calculated center point of a larger model size	28
Figure 39: The machine used in the lab to measure the bolt position accuracy	30
Figure 40: The chosen control structure. Images from [6], [7], [8] and [9].	30

Figure 41: The physical devices setup .....	31
Figure 42: RAPID code example, moving control of the weld power to the PLC .....	32
Figure 43: LD code used to generate a delay for cool-off after each weld .....	34
Figure 44: LD code used to measure the weld current duration .....	34
Figure 45: LD code used to measure the sensors analog inputs.....	34
Figure 46: The HBS “learning phase”. Images from the Visar 1200 manual.....	35
Figure 47: Designing an HMI: lead the way by only highlighting allowed actions .....	38
Figure 48: Programming example of an early version of the final HMI design .....	39
Figure 49: RAPID code example: perform a safe return path and ask the operator to flip the unit to F-side.....	40
Figure 50: RAPID code example: the quick-stop .....	40
Figure 51: RAPID code example: importing a recipe.....	41
Figure 52: Graphical reasoning of why it is crucial to separate the short side points as much as possible (exaggerated error d) .....	45
Figure 53: The final HMI - "Operator view" .....	47
Figure 54: The final HMI - "Recipe view" .....	48
Figure 55: The final HMI - "Root screen" .....	48
Figure 56: The final HMI - "Settings" .....	48
Figure 57: The final HMI - "Monitoring" .....	49
Figure 58: The final HMI - "Alarms" .....	49

## List of tables

Table 1: Examples of PROFINET signals created.....	33
Table 2: Bolt quality results from the visual and physical tests, varying the current. ....	43
Table 3: Bolt quality results from the visual and physical tests, varying the welding time.....	43
Table 4: Bolt quality results from the visual and physical tests, varying the gas flow.....	43
Table 5: Resulting position and time taken for the test welds .....	45
Table 6: Resulting position and time taken for the welds, after program improvements .....	46

# 1 Introduction

This chapter contains an overview of the company, its products, and the manufacturing process involved in this thesis project.

## 1.1 Background

### 1.1.1 The company

SWEP International AB<sup>[1]</sup> is a global manufacturing company headquartered in Landskrona, Sweden. It has since 1994 been a part of the Dover multi-industry company. SWEP has more than 1000 employees, production in five countries, and multiple offices worldwide. In a world where sustainable and effective energy use is rapidly growing its importance, SWEP leads the manufacture of brazed plate heat exchangers, BPHEs, that offer effective heating and cooling systems used in a wide range of applications. They offer the broadest product range of BPHEs on the market, in a wide range of different sizes and applications<sup>[2]</sup>. Figure 1 shows three different models of BPHEs from SWEP.



Figure 1: SWEP brazed plate heat exchangers. Image from [1].

### 1.1.2 Brazed plate heat exchangers

Brazed plate heat exchangers, BPHEs, are components that transfer heat in both heating and cooling applications. Their unique design consists of corrugated steel channel plates brazed together with highly diathermal filling material, all between two cover plates, as shown in Figure 2. This design makes BPHEs one of the most energy-effective types of heat exchangers. They can effectively be used as a condenser, evaporator, gas cooler, and more<sup>[2]</sup>. The wide range of suitable applications makes the BPHE a valuable component in residential heating, air conditioning, industrial production, refrigeration, transport, and other industrial areas. Figure 3 shows an example of a BPHE used as an evaporator with its internal flow channels.

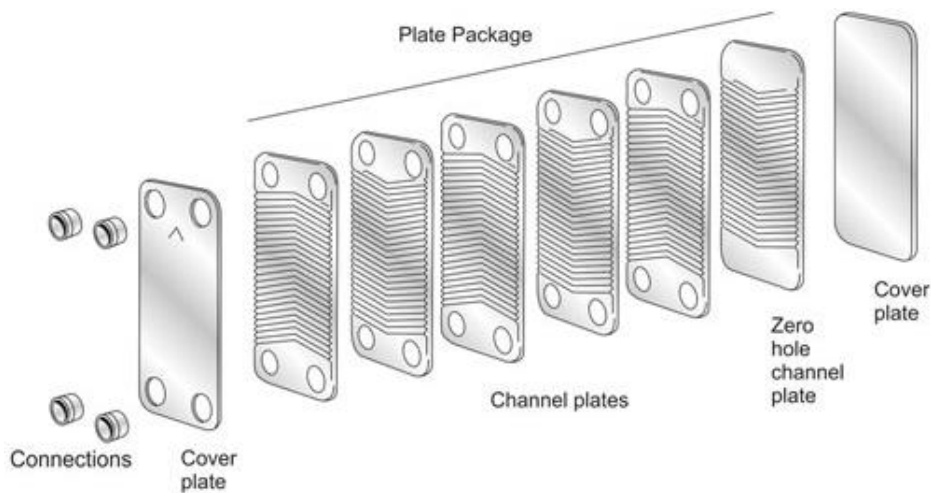


Figure 2: The inner parts of a BPHE. Image from [2].

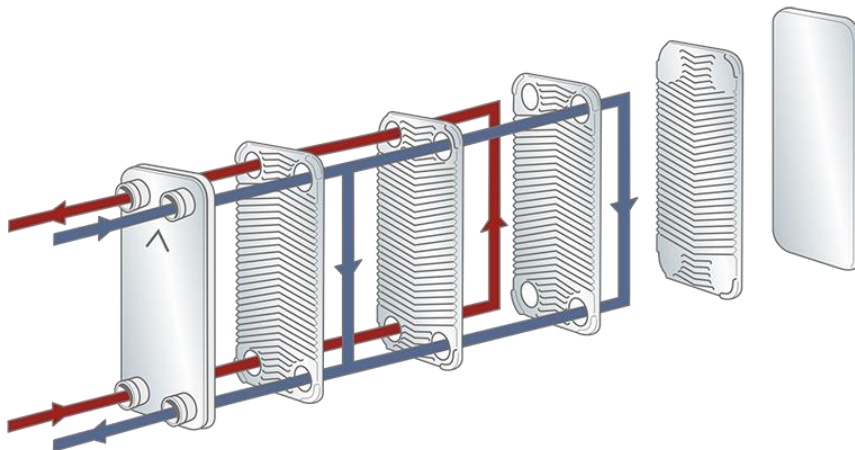


Figure 3: A BPHE used as an evaporator. Image from [2].

BPHEs have several competing benefits compared to other methods of heat transfer. They offer a more compact solution, which saves space and at the same time makes it more adaptable, as they can physically fit in a broader area of applications where other types of high energy-efficient heat exchangers struggle to operate. Their compact design also reduces the needed maintenance, as they do not need any heat exchanger gaskets typically used in heat exchangers to avoid leakage under high strain. By using a remarkably high turbulent flow, BPHEs are also

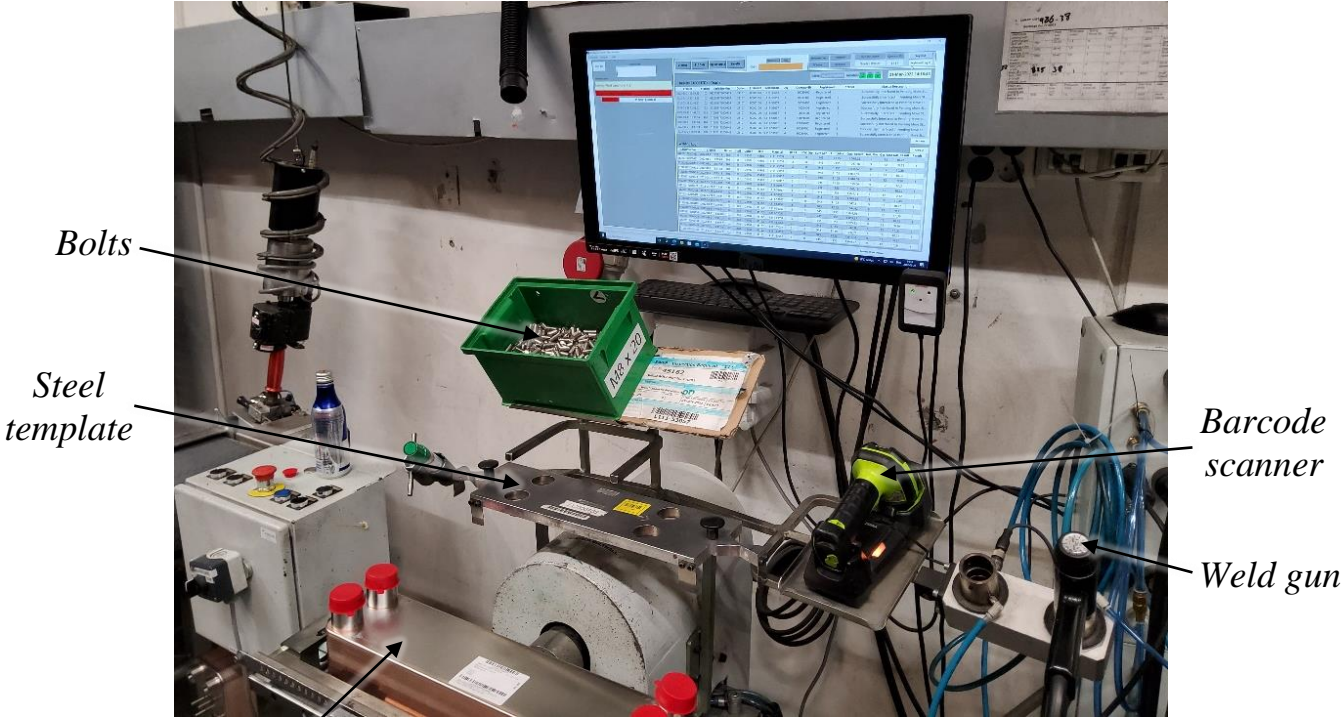
self-cleaning, which further reduces the maintenance needed. As a result of these benefits, BPHEs also becomes a very cost-effective method<sup>[2]</sup>.

### 1.1.3 The existing manual welding process

In one manufacturing step at SWEP, stud bolts are manually welded onto the heat exchanger exterior. Figure 4 shows the existing manual bolt welding station. The number of bolts needed varies between different models of heat exchangers. Bolts may also be welded on either the "P-side" of the heat exchanger, as shown faced up in Figure 4, or on the opposite "F-side".

The procedure of manually performing a weld on the P-side by an operator is as follows. First, the barcode scanner is used to scan the models used of both the heat exchanger and a steel template used to determine the positions of the bolts. The steel template is then moved and securely placed on top of the heat exchanger on the P-side. Figure 5 shows an example of a steel template used. Bolts are then placed onto the steel template, and the weld gun seen in Figure 7 is then used to weld each bolt in place. Figure 8 shows an example of a successful bolt weld.

In this manual welding process, the steel template used by the operator is a critical tool to accomplish the small tolerances needed for the resulting welded bolt position. Figure 6 shows an example of a bolt positioning configuration. The two bolts are welded on the P-side of the heat exchanger and have a tolerance of only 1mm from a fixed reference point.



Heat exchanger (P-side facing up)

Figure 4: The manual bolt welding workstation



Figure 5: The steel template used to position bolts

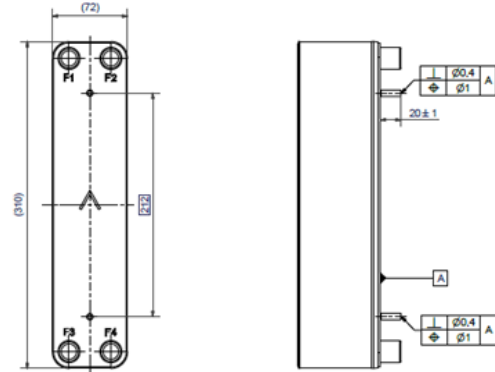


Figure 6: Example of a bolt configuration



Figure 7: The manual weld gun used



Figure 8: Example of a successful bolt weld

## 1.2 Purpose

The manual welding process described is today seen by SWEP as unnecessarily resource and time consuming and therefore is to be automated as much as possible. The work needed to automate this process has already begun at SWEP<sup>[5]</sup>. An ABB industrial robot has been programmed, and a semi-automated solution has been completed, but there are still several remaining tasks before the manual welding process can be entirely replaced by a completely automated solution.

The purpose of this thesis work is to continue the work of creating a fully automated solution that, in the future, can replace the manual process completely. This thesis work will only focus on some of the remaining tasks, and these are described in detail in Subchapter 1.4.

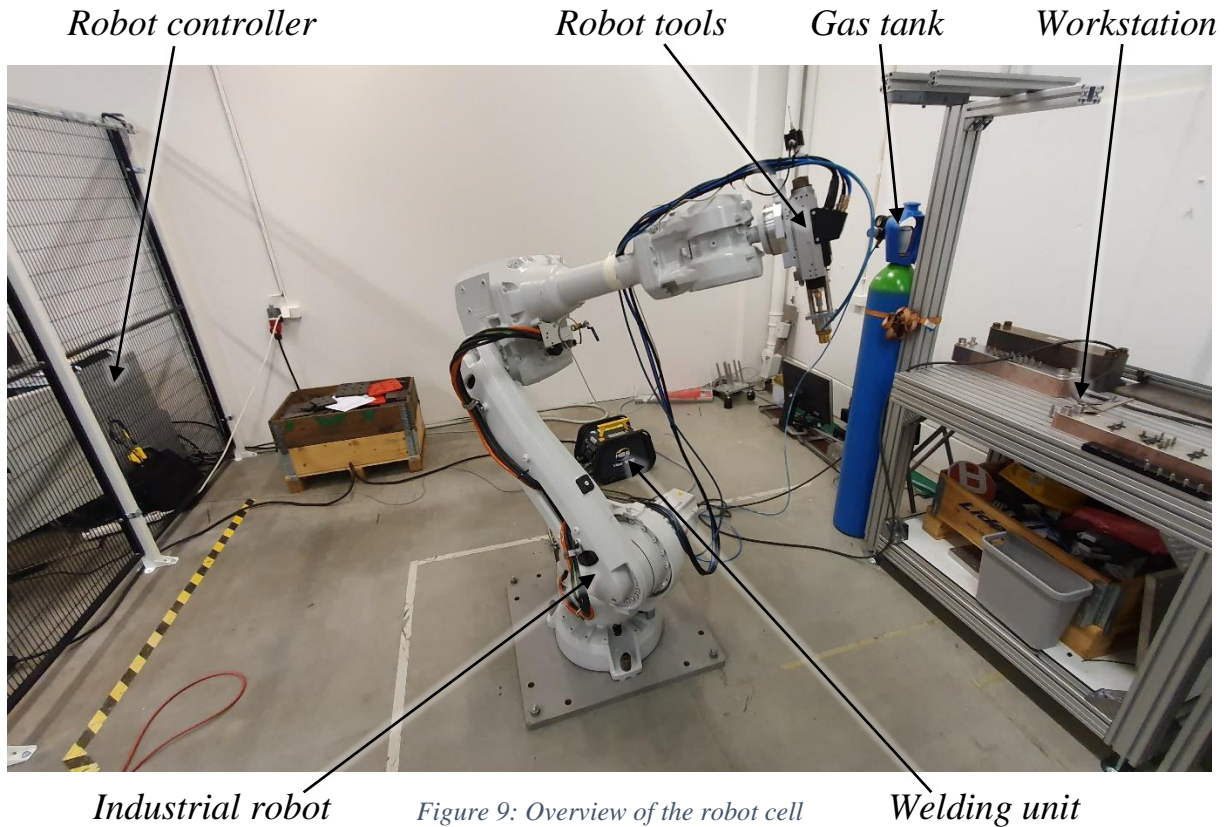
## 1.3 The robot cell

In this section, the original automated state of the robot cell will be introduced and described in detail. An ABB IRB 4600 industrial robot<sup>[3]</sup> uses an attached measuring probe<sup>[4]</sup>, weld gun, and gripper mechanism to execute the welding process. The original robot program<sup>[5]</sup> for the process was created by a consultant from Bravida Prenad AB and will also be described in detail.



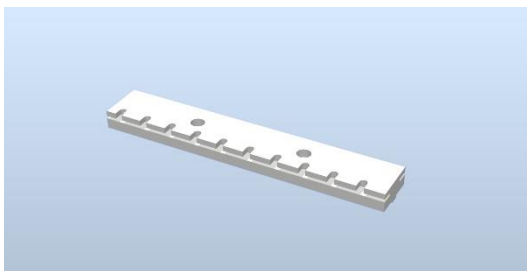
### 1.3.1 Overview of the cell

Figure 9 shows an overview of the robot cell. The robot operates on the heat exchanger placed on the workstation, using its attached tools at the end of the robot wrist. A stud bolt welding unit from HBS<sup>[6]</sup> located on the floor behind the robot provides the weld gun with current and gas flow during the weld. The industrial robot controller<sup>[7]</sup> (IRC) cabinet behind the safety fence contains the active robot program that controls the robot's movements and tools.

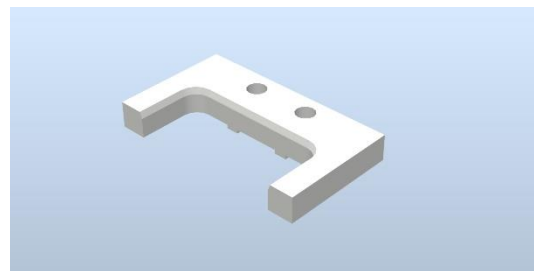


### 1.3.2 The workstation

A bolt tray is used in a fixed position on the workstation to hold the bolts picked up and used by the robot. Figure 10 shows a 3D model of the bolt tray. A heat exchanger fixture is also used to assure that the heat exchanger will be kept still and correctly positioned during the welding process to achieve the required tolerances of the bolt positions. Figure 11 shows a 3D model of the fixture. Figure 12 shows the physical setup of the fixture and a partly filled bolt tray used with a heat exchanger.



*Figure 10: 3D model of the bolt tray*



*Figure 11: 3D model of the heat exchanger fixture*



Figure 12: Workstation setup with bolt tray and fixture

### 1.3.3 Welding gun and measuring probe

Two separate tools are attached at the end of the robot wrist: an automatic welding gun, as seen in Figure 13, and a touch probe, as seen in Figure 14. The automatic weld gun includes a gripper mechanism to pick up the stud bolts. The robot uses the touch probe to find various dimensions and points of the heat exchanger. The metal tip rod of the probe is movable as it is connected to an internal spring. During contact with the heat exchanger surface, the tip gets moved, and the internal sensors detect the touch. The robot controller can then save the point of contact.

To combine these two tools for the robot, a custom-designed tool frame made at SWEP is used to hold both the weld gun and the touch probe with the frame attached to the robot wrist, as seen in Figure 15. The robot can then rotate the tool frame and be able to use both of the tools.

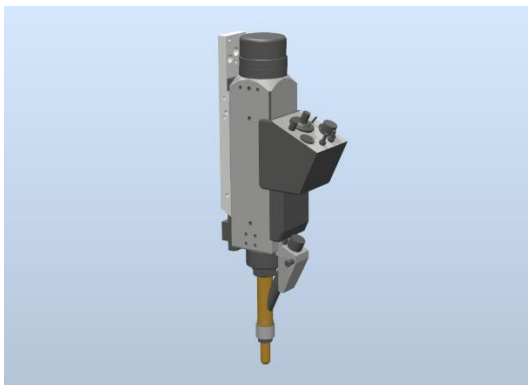


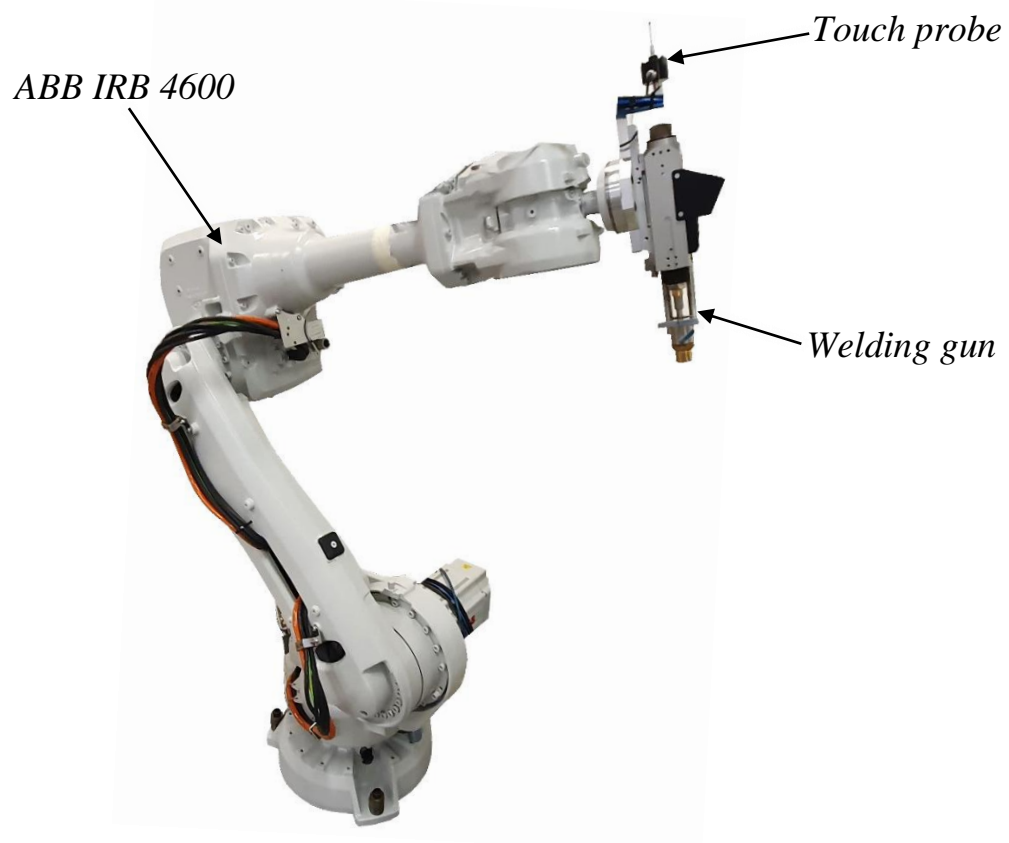
Figure 13: 3D model of the weld gun



Figure 14: The touch probe. Image from [4]

### 1.3.4 The ABB industrial robot

An ABB IRB 4600 industrial robot is used in the bolt welding process to pick up stud bolts, measure the center and orientation of the heat exchanger, and perform the welds. This robot model utilizes a more compact footprint, slim base axis, and a more compact and flexible wrist axis than many other models. The model also allows the highest maximum acceleration and speed within its class. These factors make it a suitable choice of robot in many different automation applications, allowing the robot to operate close to other machines and workstations, saving space and thus expanding productivity. Figure 15 shows the robot and its attached tools.



*Figure 15: The industrial robot used in the welding process*

### 1.3.5 IRC5 robot controller

All movements and speeds of the robot are operated and controlled by an IRC5 industrial robot controller, as seen in Figure 16. This controller provides all the functionality and support that the robot requires, all combined as a single system containing a main processor, power supply, communication systems, and more. This setup provides modular flexibility, centralized safety protection, multi-robot control, external PC support, and automatic diagnostic and quick recovery. The processor performs real-time dynamic modeling of the robot joints and their angular speeds to control the tools' movements and give precise path accuracy. The robot controller is programmed using ABB's high-level programming language, RAPID.

A feature of the robot controller used extensively in the thesis work is the separation of control into two different modes: automatic mode and manual mode. In automatic mode, the

robot controller operates the program independently without the need for an operator in control. This mode should only be used when the program algorithm has been verified and fully trusted not to damage the robot or workstation. In manual mode, the speed is limited, and the program may only continue while an operator is present. This mode should be used while testing the program code, letting the operator quickly stop all robot movements if an unexpected event is seen.



*Figure 16: The IRC5 robot controller cabinet*

### 1.3.6 The FlexPendant interface

The FlexPendant is a handheld control device consisting of software and hardware connected to the robot controller. The operator can interact with the system from the FlexPendant by using a touch screen and joystick to run and debug the RAPID program modules, alter specific instructions within the program, manually jog the robot joints, and more. The FlexPendant is especially used in manual mode. Its enabling switch located on the back acts as a "dead man's switch", i.e., forcing the operator to hold down the switch continuously during program execution. As soon as the switch is released, all robot movements halt.

### 1.3.7 The HBS Visar 1200 welding unit

The weld gun power is monitored and controlled by a welding program uploaded on the HBS Visar 1200 welder unit, as seen in Figure 17. A welding program defining the welding parameters, for example, the level of current, gas, and time duration, can be programmed and uploaded via USB. A digital display allows the operator to view and easily change welding parameters.

This specific model only has internal monitoring, without support to externally export the data monitored. To monitor these parameters in an external controller, they must be measured and imported separately.

By using an HBS welding unit, the welding result is ensured to be of higher quality and more reliably reproducible on all welds, compared to the manual welding process. The welding unit has an internal microcontroller that continuously measures and analyses the welding parameters, and a control algorithm processes this data to regulate the level of current and gas to the defined target values.



Figure 17: The HBS welding unit. Image from [6]



Figure 18: The FlexPendant operator unit

### 1.3.8 Grounding the heat exchanger

During the weld, the heat exchanger must be stationary. Also, it must be electrically grounded to not allow any current flow through the heat exchanger from the weld gun. Since the magnitude of current is large, often around 1200 A, even a slight difference in electrical potential across the heat exchanger creates a leaking current large enough to generate a Lorentz force as of the right-hand rule, pushing on the heat exchanger. To solve this, the heat exchanger is grounded on both ends, as shown in Figure 19, using metal clamps connected to the ground of the HBS welding unit.

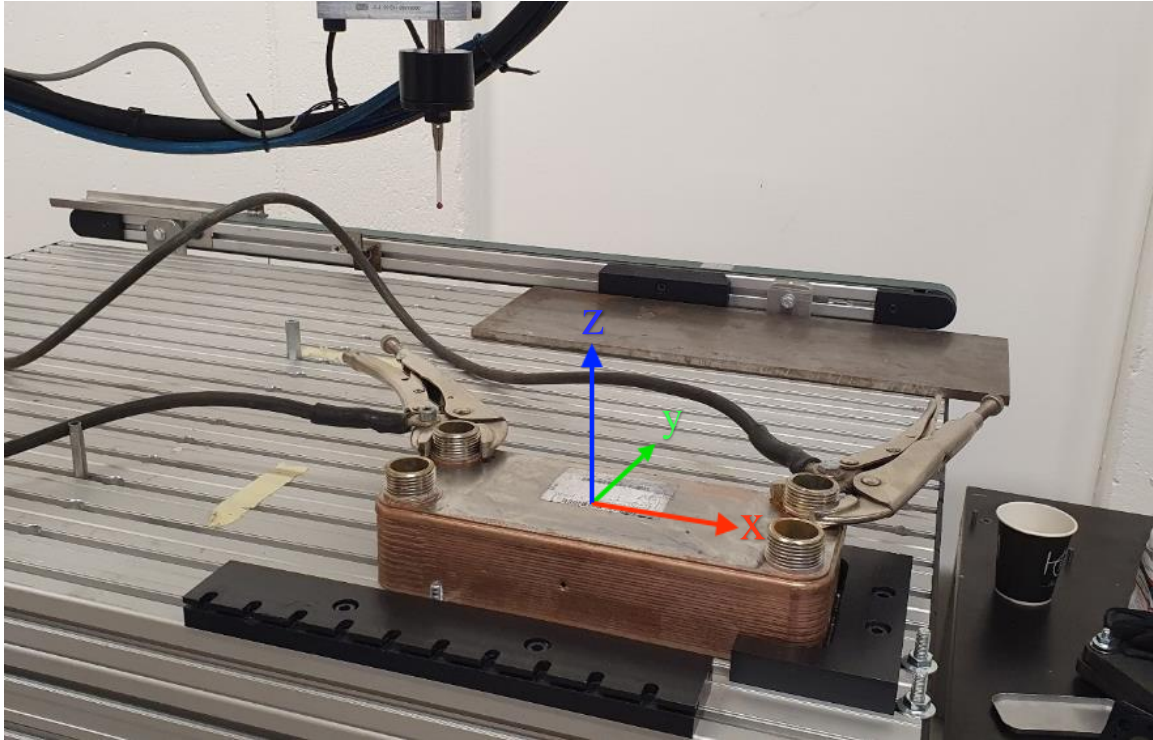
When the bolt welding process has been fully automated and is put to work in the factory, this way of grounding the heat exchanger will, however, not be used. Even though this method is sufficient during the testing phase, it is too unreliable and space-consuming to use in the confined robot cell where the robot will finally operate. Instead, the grounding will be achieved like many other welding processes at SWEP, using wide copper contacts placed on top of the heat exchanger. These contacts create an extensive enough surface contact not to create any sparks, which could happen when a weak contact is used.

### 1.3.9 The original welding process algorithms

This section will introduce all relevant functions of the original welding program. Only the fundamentals will be shown, as seen by the operator. More in-depth details on how the overall algorithm works, how the robot accurately searches for points, and how the center reference and orientation are calculated are described in Appendix A.

## The starting position and initial conditions

Figure 19 shows the initial starting position of the program and the configuration of the workstation. The heat exchanger is mounted on the fixture from Figure 11. The bolt tray from Figure 10 is mounted on the workstation in front of the heat exchanger. Both sides of the heat exchanger are grounded using metal clamps.



*Figure 19: The welding program starting conditions*

### **Method I: Measuring the center reference and orientation using 8 points.**

Figure 20 summarizes the program function for measuring the center reference point and the orientation of the heat exchanger. Firstly, the surface plane of the heat exchanger is measured at three points (a--c) using the touch probe tool mounted on the robot's wrist. Then, with these three points saved by the robot controller, the program calculates the surface plane. By calculating the plane from three points, any inclination of the workstation or heat exchanger around the x- or y-axis can be accounted for. Secondly, all four sides of the heat exchanger are measured (d--h), including an extra point measured on one of the sides (f).

If only one point is measured on each of the four sides, the orientation of the heat exchanger around the z-axis would not be accounted for. To solve this, an extra point is measured on one of the four sides (f), which now explicitly determines the plane's position and orientation. In Figure 22, a summary of all the points saved to the controller using the first method is shown in red.

For each to be measured side point, the touch probe is first moved close to the measured side and then slowly approaches the side until contact is detected, as seen in Figure 21. The search movement speed must be low to ensure that the side point is accurately measured. When

these points have been measured and saved to the robot controller, the orientation and center reference point can be calculated.

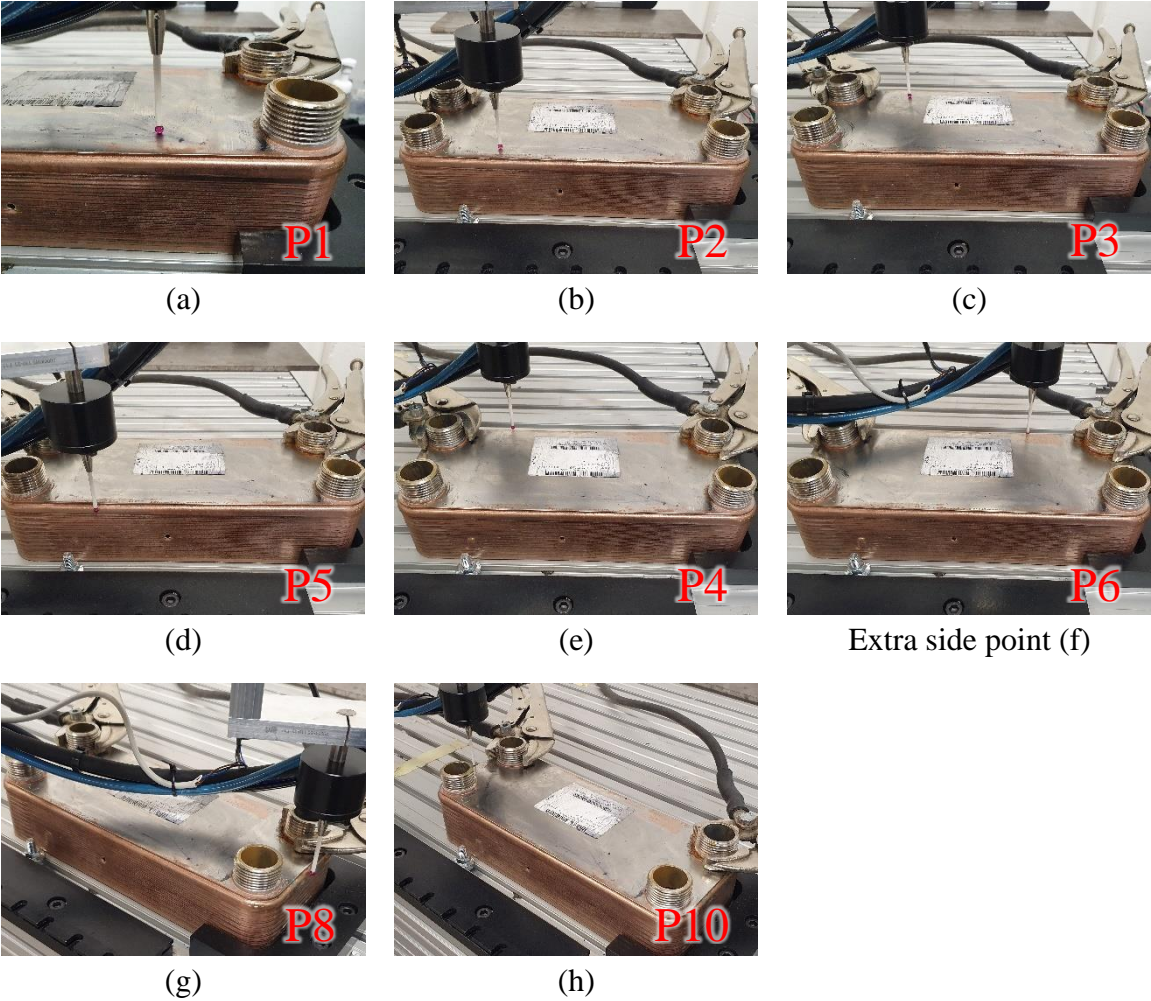


Figure 20: Points measured to calculate the heat exchanger's center reference point and orientation, Method I.

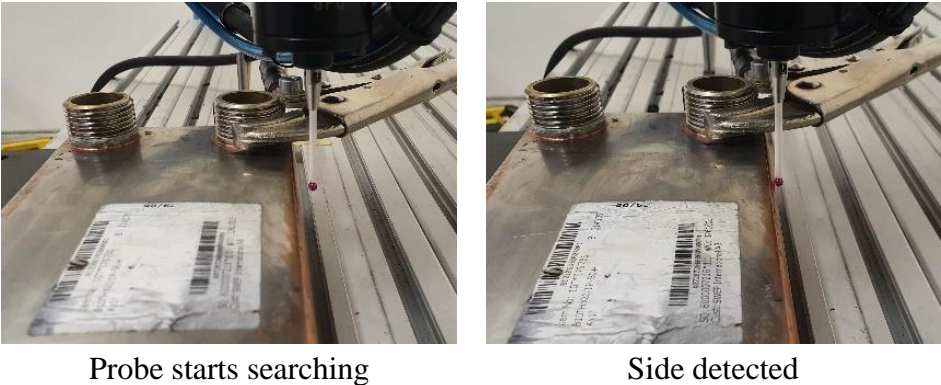


Figure 21: A side point being approached by the probe

To calculate the orientation and z-coordinate of the surface plane, two vectors on the surface plane are defined from the three surface points P1, P2, and P3, as seen in Figure 22. Together, these two vectors unambiguously determine the surface plane equation.

To calculate the center reference point, the following algorithm is used: Calculate the average point of P8 and P10. Save the x-value of this new point. Calculate the average point of P4 and P5. Save the y-value of this new point. Define and save the center reference point as the x- and y-value saved. This method is visualized in Figure 23.

To calculate the orientation of the heat exchanger around the z-axis, the following algorithm is used: calculate the vector from P4 to P6. Calculate the angle between this vector and the x-axis. Finally, rotate the previously calculated surface plane around the z-axis with the calculated angle. If the heat exchanger is not rotated at all, this vector and the workstation x-axis will be perpendicular, and the calculated angle would be zero.



Figure 22: The measured points saved to the controller

\* Points only used by the second method of calculating the center, not needed by Method I.

## Method II: Measuring the center reference and orientation using 11 points.

For the second method of calculating the heat exchanger dimensions, the algorithm for calculating the surface plane is done the same way, using the three surface points P1, P2, and P3, as described in Figure 20 (a--c). However, the algorithm for calculating the center reference point, as well as the number of measurements done on the sides, is different in this method. Each side point is measured in the same manner, but now, an additional three points P7, P9, and P11 are measured, as seen marked in Figure 22 (\*).

To calculate the center reference point, the following algorithm is used: Calculate the line through P8 and P11, as well as the line through P9 and P10. Calculate the intersection point of these two lines. Save the x-value of this intersection point. Calculate the line through P5 and P6, as well as the line through P4 and P7. Calculate the intersection point of these two lines. Save the y-value of this intersection point. Define and save the center reference point as the x- and y-value saved. This method is visualized in Figure 24.



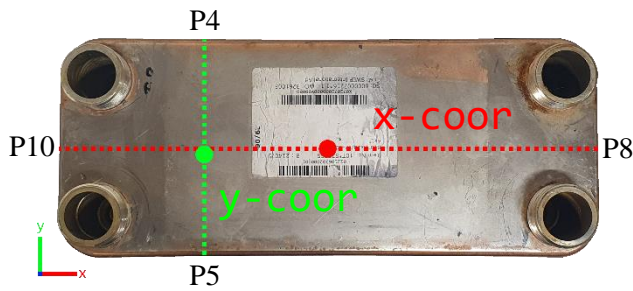


Figure 23: Calculation of the center point, Method I.

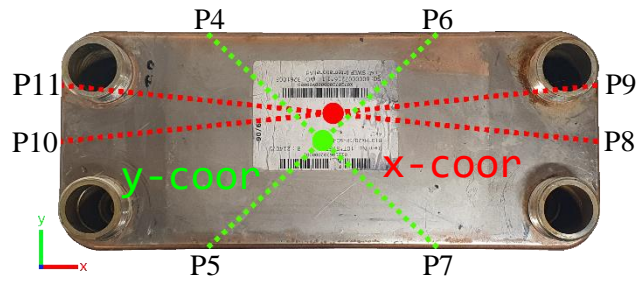


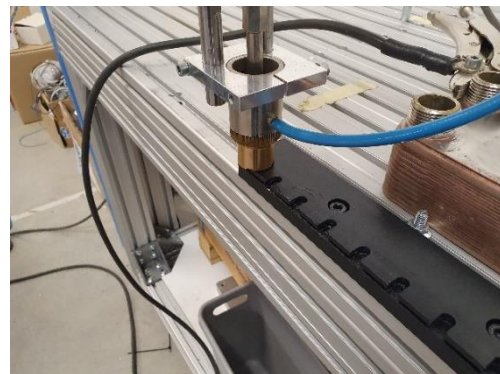
Figure 24: Calculation of the center point, Method II.

## Performing the welds

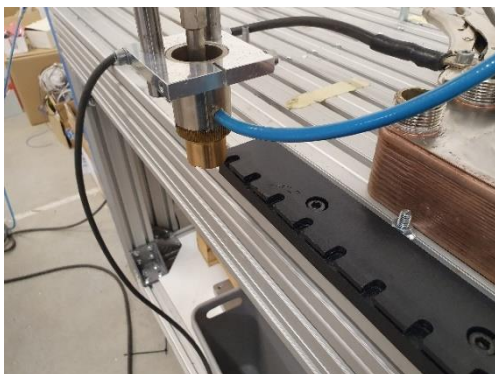
In Figure 25, the program function for welding a bolt onto the heat exchanger is summarized. This function can only be used after the center reference point and orientation measurements have been performed for the used heat exchanger model. First, the weld gun mounted on the opposite side of the touch probe is moved close above the first position of the bolt tray (a). The weld gun is then lowered onto the bolt (b). When the bolt is fully enclosed, a self-locking mechanism will continue to hold the bolt in place when it is slid out of its tray position (c). With the bolt held inside, the weld gun is then moved to the first weld position (d). Here, the robot controller will signal to activate the program of the HBS welding unit, successfully welding the bolt to the heat exchanger. While in this position, the welding gas is continuously flowing into the weld gun chamber.



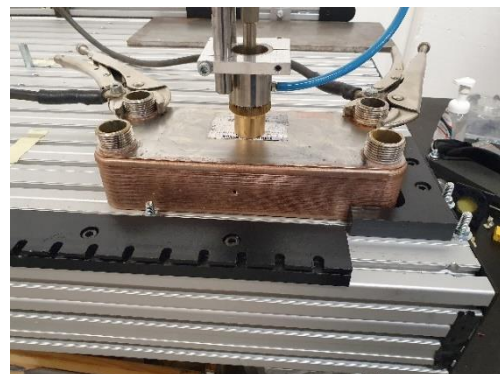
(a) Weld gun above bolt tray position



(b) Weld gun lowered onto the bolt



(c) Bolt has been picked up



(d) Bolt is welded to the surface

Figure 25: Welding a bolt to the heat exchanger

## 1.4 Objectives

This section will describe the necessary tasks within the thesis work of continuing the automation project at SWEP. This includes several tasks, many of which may be implemented and finished independently of each other. Even so, they all serve the same goal of expanding the project of automating the welding process. The work performed on these tasks within this thesis is described in Chapter 4.

### 1.4.1 Perform quality analysis of the automated welding process

The resulting bolt weld quality performed by the current state of the automated solution should be evaluated by testing different weld parameter configurations of the HBS welding unit, i.e., different amounts of weld current, welding time, and gas flow. A series of tests including a bend test, torque test, and visual inspection should also be performed on the bolts.

### 1.4.2 Evaluate and improve the program algorithm

The current robot program algorithms include a calculation of the heat exchanger's center reference point and inclination, and these algorithms should be verified and evaluated. This thesis considers two existing methods of calculating the center point, and these should both be analyzed and have their accuracy measured and compared to know which method should be used in the finished automated process.

Also, an attempt to improve the program algorithms should be made if the evaluation or measured accuracy determines that there is a need for it.

### 1.4.3 Implement a control structure for a fully automated solution

A fully automated control and communications system between the HBS welding unit, IRC, a PLC<sup>[8]</sup>, and a HMI panel<sup>[9]</sup> must be investigated and implemented to fully automate the welding process, with operator control using only the HMI.

After this has been set up, the PLC must be programmed to activate the HBS welding program, import relevant data from the welding unit and robot, and fully automate a "work order" through the HMI, i.e., a chosen recipe of bolt positions, on both the P- and F-side, of a certain model size, and of a selected amount of units.

The relevant welding parameters that should be imported must be defined. All parameters of the HBS welding program of which a certain value or interval may imply a threat or risk to the robot or process should be included.

### 1.4.4 Enable a fail-safe system in the HBS welding unit

After the control and communication system has been set up, the HBS welding unit should be set up to monitor and act upon the welding parameters in real-time during the automated welding process to create a fail-safe feature for the process. With this welding program, the automated welding process will be protected from damage to the robot tool, heat exchanger, and weld quality, for example, when the welding gas intake or current level gets too high or low.

### 1.4.5 Design and implement a Human-Machine-Interface

An assisting Human-Machine-Interface, HMI, must be designed, programmed, and implemented. With this interface, an operator should have quick access to all controls and real-time status needed in the fully finished automated solution. The appropriate features to implement in the HMI must be decided. As a minimum, the HMI must include a start and stop function and a function to receive a work order from the operator with the number of bolts and configuration to be used.

## 1.5 Problem formulation

In this section, the purpose and objectives already described will be presented and reflected as a set of clear and measurable problem formulations and questions. In summary, these are the questions that this thesis work will attempt to answer.

- What does the current semi-automated state of the process look like?
- How can the original program algorithms be improved?
- How should the center reference point of a heat exchanger be calculated?
- How can an analysis of the automated weld quality be performed? Which welding parameter values will assure an optimal weld quality?
- What control structure and communication are needed between the IRC5, PLC, HMI, and HBS welding unit?
- How should the HBS welding unit program be activated?
- How can a fail-safe system be implemented in the HBS welding unit?
- What features are needed in the HMI for a fully automated solution?
- What tasks will be left before the process can be fully automated and replace today's manual process?

## 1.6 Motivation

I applied to this degree project as it fits in well with the knowledge I have gained so far in my studies towards a degree in electrical engineering and automation. My personal goal is that I will have the opportunity to expand this knowledge further and gain a rewarding insight and experience of what it is like to work as an engineer in automation. I see SWEP as a company where I can achieve these personal goals and expectations. SWEP is globally established and a world-leading developer in its field, which strongly captures my interest and motivates me to do a well-executed degree project to conclude my education. I hope this project will provide

SWEP with valuable findings and a functioning addition towards automating their welding process.

## **1.7 Boundaries**

In this section, all boundaries of the thesis work will be defined. These describe the chosen limitations of the project.

- The welding process is only considered to use M8 stud weld bolts.
- A work order recipe can have a maximum number of four bolts on each side of the heat exchanger.
- The HMI program is only considered to be used on a Siemens KTP400 Basic HMI panel.
- The PLC program is only considered to be used on a Siemens SIMATIC S7-1200 PLC.

## 2 ABB RobotStudio and Siemens TIA Portal

This chapter will introduce the two software programs used heavily throughout this project: ABB RobotStudio<sup>[10]</sup> and Siemens TIA Portal<sup>[11]</sup>.

### 2.1 ABB RobotStudio

#### 2.1.1 The simulation environment

RobotStudio is a simulation and programming environment created by ABB. It allows robot programmers and engineers to imitate a real robot cell in a virtual simulation environment. This allows the robot cell to be programmed, tested, or further developed in an environment that is completely isolated from the real robot cell, without the need to disrupt or modify the existing robot station. Furthermore, by using an ABB Virtual Controller, the IRC5 robot controller can be fully emulated to allow development on the robot cell completely remote from both the robot cell and controller. Figure 26 shows an example screenshot of the RobotStudio simulation environment.

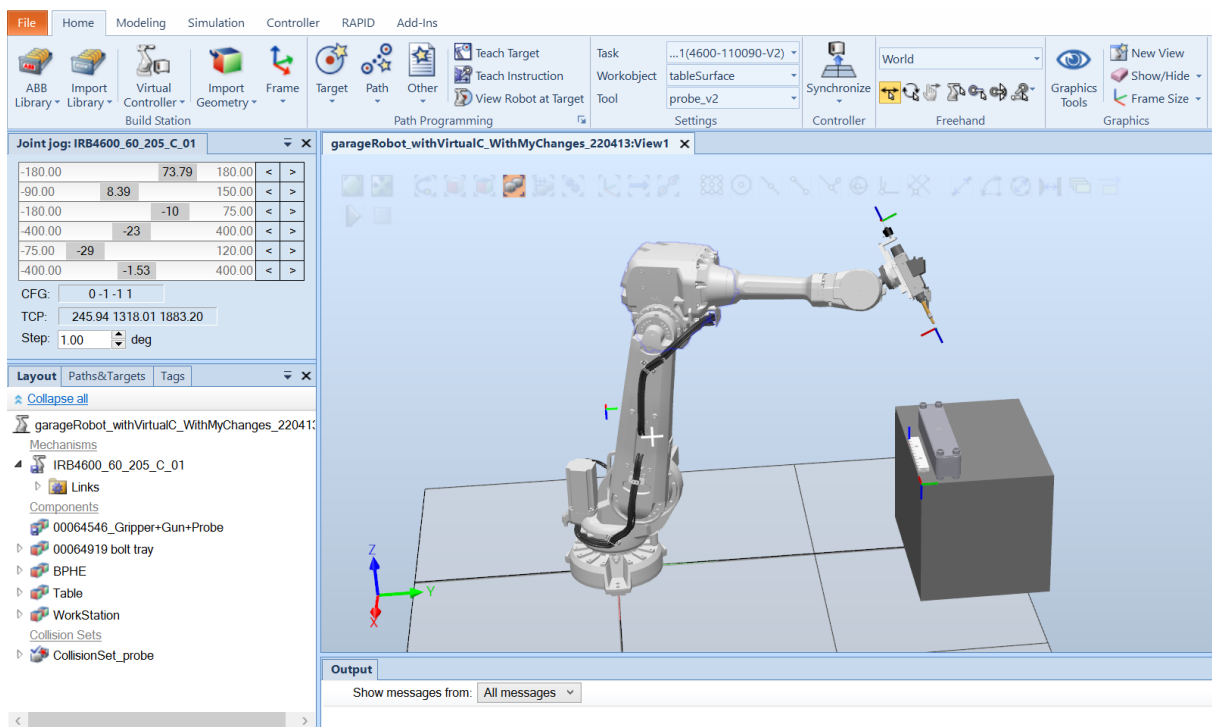


Figure 26: The RobotStudio simulation environment

In the thesis work, RobotStudio is used to simulate the complete robot cell, including the IRB4600 industrial robot, measuring probe, welding gun, bolt tray, workstation table, and heat exchanger. Along with a virtual controller of the used IRC5, the full welding program could be tested and investigated early on and when not working at the “garage” at SWEP where the physical robot cell was located.

## 2.1.2 The RAPID programming language

RAPID is a high-level programming language made by ABB, used to program their industrial robots within RobotStudio. A robot program may contain multiple *modules*, for example a *procedure* (does not return a value), a *function* (does return a value), or a *trap routine* (interrupt routine). In this section, the most relevant variable types and instructions of the RAPID programming language within this project are explained so that the reader can understand the RAPID code examples presented later in the report.

**TCP:** The Tool Center Position coordinate system. This is the center point of the active tool.

**robtarget:** A defined cartesian pose of the robot and tools, consisting of the tool coordinates, tool rotation, and axis configuration. As the robot often has multiple solutions (angle combinations of the six axis) to be at some point, a solution may be chosen as the "axis configuration" used, defining what rotational quadrants the six axis should be in.

**speeddata:** Specifies the velocities used in a move instruction, including the TCP velocity, reorientation velocity, etc.

**zonedata:** Specifies how a movement path should be terminated, i.e., how close the tool must be to an intermediate target point before starting to move towards the next position, "cutting corners".

**workobject:** A defined coordinate system, relative to some other system. For example, the tip of the touch probe. When the probe moves, the workobject is automatically updated.

### **RAPID instructions:**

**MoveL:** The TCP is moved to a target point, at a defined speeddata and zonedata, along a linear path, relative to a robtarget. All robot joints need to collaborate to ensure precise linear movement at constant speed.

**MoveJ:** The TCP is moved to a target point, but a linear path is not enforced. Instead, each robot joint transfers itself to its own end value independent of the other joints. The speed for each joint is set so that all joints reach their target angles at the same time.

**Offs:** Returns the point at a chosen offset from an input target. For example, it can be used to define a point 20mm below some dynamic point, like the touch probe or weld gun tip.

**CRobT:** Reads the current robtarget data of some workobject or tool. For example, the touch probe or weld gun tip.

**SearchL:** The TCP is moved linearly in small increments towards a target point, actively listening to a chosen signal input, and stops immediately when a flank on that signal is detected. This instruction is used with the touch probe to find the dimensions of the heat exchanger, as shown in Figure 21.

**CONNECT:** Initiates an interrupt routine.

**ISignalDI:** Sets an interrupt routine to execute when some digital input signal has a rising or falling edge.

**IDelete:** Cancels an interrupt routine.

**StopMove:** Halts all robot movements temporarily.

**StartMove:** Allow movements again after StopMove.

**ExitCycle:** Breaks the current program cycle, moving the program pointer back to the main routine.

**TPWrite:** Prints a message on the FlexPendant. It is used for testing, debugging code, and to give information to the operator.

**WaitTime:** Pauses program execution a given amount of time.

**WaitDI:** Pauses program execution until a signal has been set or reset.

**SetDO:** Changes the value of a digital output signal.

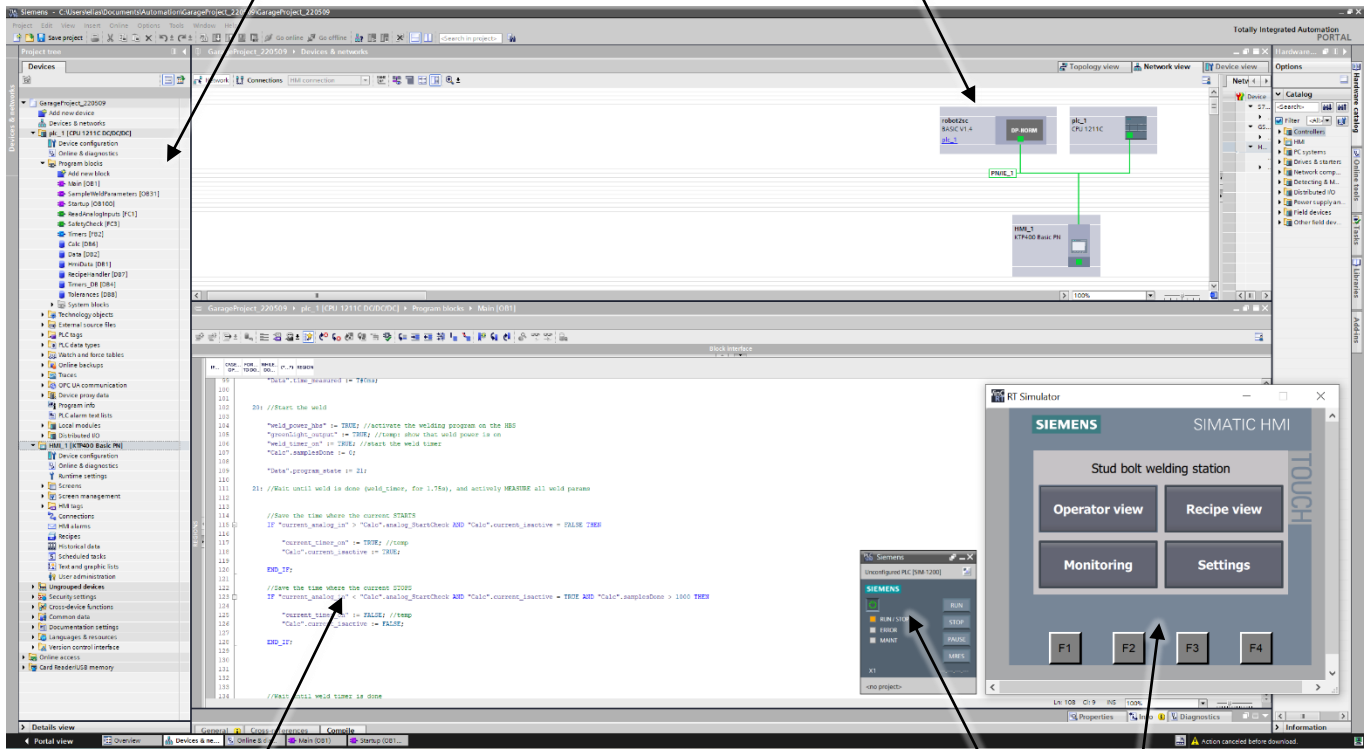
## 2.2 Siemens TIA Portal

### 2.2.1 The development environment

The Totally Integrated Automation software platform from Siemens, TIA Portal<sup>[8]</sup>, offers a solution of centralizing all development work for Siemens PLCs, HMIs, drivers, and PROFINET<sup>[12]</sup> devices on a single platform. TIA Portal can be used in an automation project to deploy the needed devices, software design, program the PLC and HMI, set up a PROFINET network structure, automate machine diagnostics and security, and optimize energy conservation. With these capabilities, TIA Portal provides an efficient way to create a streamlined production process where automation engineers can share and collaborate on the same projects over the cloud, making it a common platform of choice in the process industry. By combining the use of TIA Portal with Siemens S7-PLCSIM<sup>[12]</sup> for simulating a PLC, along with Siemens SIMATIC WinCC<sup>[13]</sup> for simulating and visualizing an HMI, a fully simulated solution was used during the thesis work when not working at the robot cell located in the “garage” at SWEP. An example view of TIA Portal, S7-PLCSIM, and SIMATIC WinCC, as often used in the thesis work, can be seen in Figure 27.

*PLC modules, tags,  
PROFINET devices, etc*

*PROFINET  
network view*



*PLC programming*

*Simulated PLC and HMI, in  
separate windows*

Figure 27: TIA Portal with simulated PLC and HMI

## 2.2.2 Programming a PLC

In TIA Portal, the available PLC programming languages are Ladder Diagram (LAD), Function Block Diagram (FBD), and Structured Control Language (SCL). These correspond to LD, FBD, and ST as defined in the commonly used PLC programming language standard of IEC 6113-3<sup>[14]</sup>. A new program module can be created in TIA Portal, referred to as a "program block", programmed in a chosen language, chosen as one of the four program block types as seen in Figure 28.

The "Organization block" is a block that contains more detailed execution conditions. For example, an organization block can be set up to be run continuously as a main program (in TIA Portal called a "Program cycle"), or only when the PLC was started ("Startup"), or at a specific periodic interval ("Cyclic interrupt"), or every time a module is inserted or removed ("Pull or plug of modules"), and more.

The "Function block" can be written in LAD, FBD, or SCL and called from other blocks during program execution. Its values are stored permanently between function calls and not lost after a value is returned. It can, for example, be used to update a TONR timer, conserving the last state and elapsed time between calls.

A "Function" is identical to the "Function block" except it does not conserve any values between calls and these are instead lost after the function returns a value. It can, for example, be used for repeated calculations or processing inputs, where only the returned values are of interest.



The "Data block" is used to declare, manage, and save data. For example, the common data types "Int", "Real", "Byte", and "Date". It can also be used to define start values, set read- and write permissions for the HMI, and more.

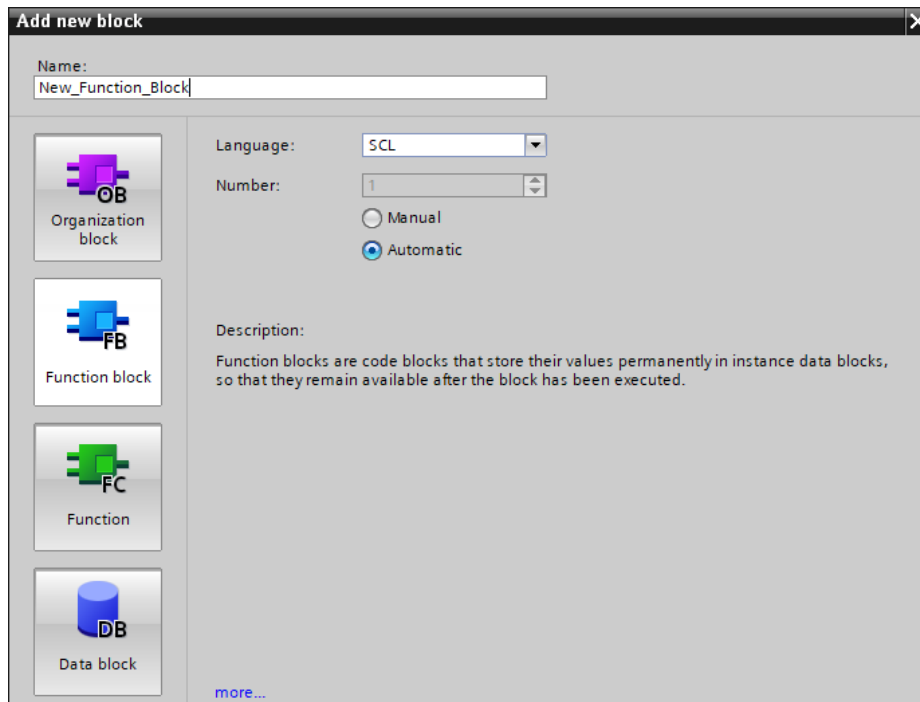


Figure 28: TIA Portal – the four types of program blocks

In TIA Portal, internal program data and variables are saved and used in Data blocks. The physical PLC inputs, outputs, and other specific memory bits are managed separately and referred to as PLC "tags" in TIA Portal. Each tag is assigned a name, a data type, and is related to a chosen memory address, as seen in Figure 29. The address prefix %I refers to inputs, %Q to outputs, and %M to specific memory bits. Data types other than "Bool" have an extra prefix for defining the tag type. For example, a "Word" (2 bytes) have the prefixes %IW, %QW and %MW. Following the prefix, the bit or bit's addresses are defined. For example, %I68.1 referring to the first bit on the sixty-ninth byte in memory. These memory addresses can then be easily read and written to from the program blocks by using the tag names. For example, if the physical PLC output pin 0.4 is connected to the control input of the HBS welding unit, the welding program can be started in code using the tag name weld\_power\_hbs, with no further need to remember the exact bits that are used.

Weld Control tags								
	Name	Data type	Address	Retain	Acces...	Writa...	Visibl...	Comment
1	irc_want_weld	Bool	%I68.1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
2	weld_power_hbs	Bool	%Q0.4	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
3	weld_done	Bool	%Q2.0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
4	weld_timer_done	Bool	%M0.2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
5	weld_timer_on	Bool	%M0.3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

Figure 29: TIA Portal - PLC tags

## 2.2.3 Creating an HMI layout

In TIA Portal, an HMI layout can be designed by using the provided HMI elements. Simple objects such as lines, shapes, text, and images can be dragged onto an HMI screen from the "HMI toolbox" as seen in Figure 30. In the "Screen view", the active screen layout with elements is displayed.

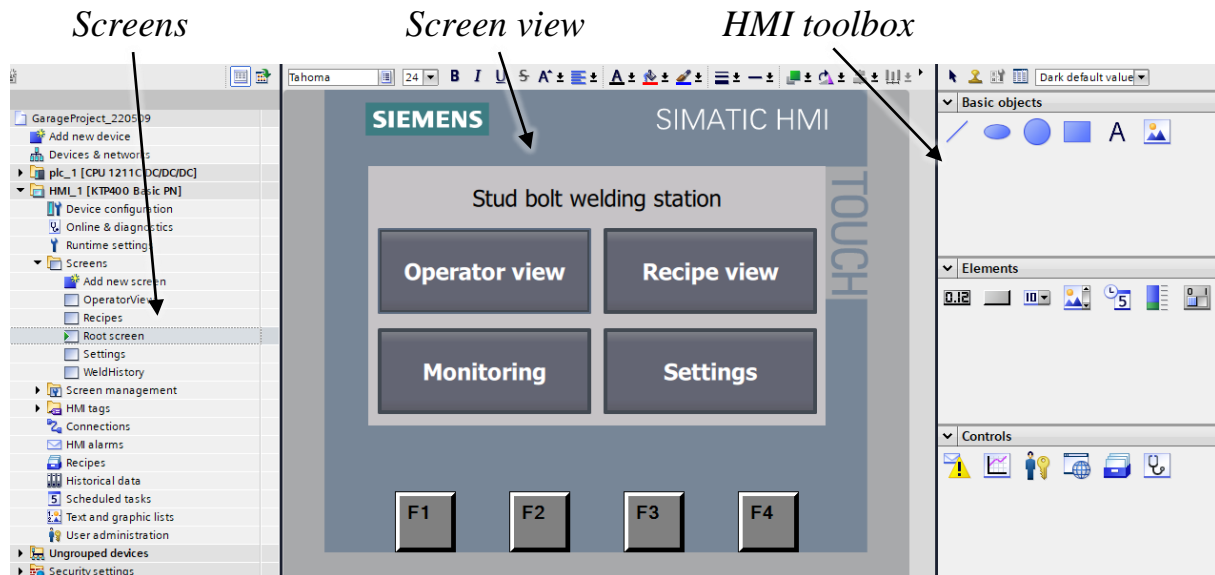


Figure 30: TIA Portal - HMI development

Other than the basic objects, TIA Portal offers more advanced HMI elements that are often linked with PLC tags and variables. Within the thesis work, the following HMI elements were used:



**I/O field:** displays a text related to a PLC tag or variable. It can be used in "Output mode" to only display its value on the HMI or in "Input mode" to let the operator change its value.



**Button:** a pushbutton for the operator to use. Different "events" can be set up with instructions to execute when the button is released, pressed, etc. For example, to change the value of a PLC tag or open a new screen.



**Symbolic I/O field:** identical to the normal I/O field, except that it uses a discrete set of values. For example, letting the operator choose one of five options from a drop-down list or displaying a text that changes for different values of some PLC tag or variable.



**Recipe view:** a default graphical view to use with recipes. It includes a drop-down list that can be linked to configured recipes, a table to show recipe elements and data, and buttons for saving, deleting, and modifying existing recipe records.



**Switch:** a flip switch for the operator to use. It can be used in both output and input mode, connected to a boolean PLC tag.

# 3 Method

This chapter presents the followed schedule of the thesis work and the chosen methods to complete the defined objectives of the thesis work.

## 3.1 The thesis work

Before the 15-week thesis work began, an initial project plan was created and approved by the supervisor at Lund University, the supervisor at SWEP, and the examiner. The project plan contained a preliminary description of the thesis work proposed by SWEP and the objectives of the project. An initial project schedule was also created, as seen in Figure 31.

During the first two weeks, the focus was put on initial report writing and preparations. Early on, a detailed list of tasks clarifying the thesis work was created with the help of the supervisor at SWEP. Then, as there was no prior personal experience in neither RobotStudio nor TIA Portal, more knowledge and experience were needed before fully committing to the thesis work. Therefore, a similarly detailed list of research and learning objectives was created. The preparatory tasks put on this list were fully completed during the first two weeks, which provided the needed experience and skills within RobotStudio, RAPID programming, Siemens TIA Portal, the manual welding process, operating the robot cell safely, the operating manuals for the touch probe and HBS Welding Unit, and more.

The robot cell and all needed equipment were located and provided by SWEP at their development “garage” of their main office located in Landskrona, Sweden. A majority of the time was spent at the garage, and the rest at the office of their separate production facility when not in need of the physical robot cell. Meetings and discussions with the supervisor at SWEP were planned continuously and often done daily to quickly address any problems or questions formed during the work.

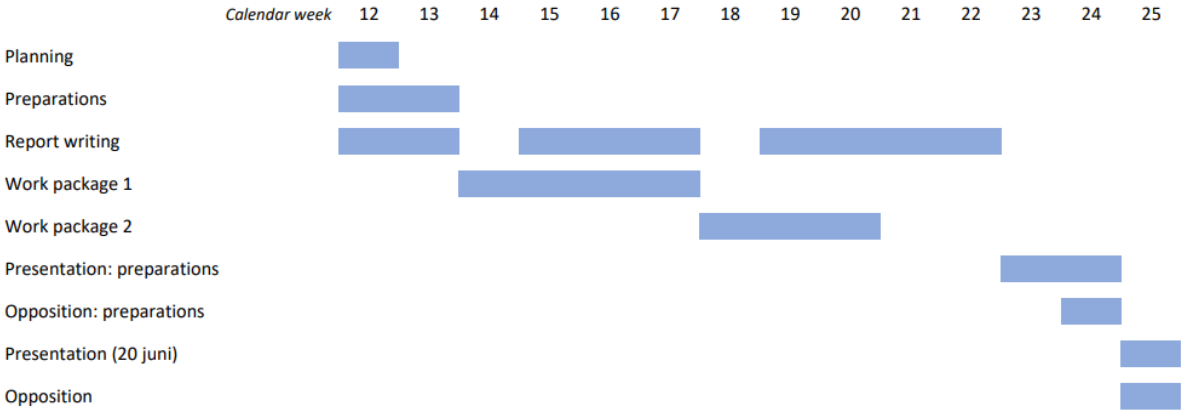


Figure 31: The prepared thesis work schedule

## 3.2 Methodology

This section describes the chosen methods for completing the objectives defined in Subchapter 1.4.

### Performing quality analysis of the automated welding process

A series of welding tests will be performed to assess the weld quality when using an automated solution and to find a suitable set of welding parameters to use within the HBS welding unit. In these tests, the welding parameters (current, welding time, gas flow) will be changed manually on the HBS welding unit. To understand how and why each weld parameter affects the welding quality, every parameter will be tested in its normal range as well as for lower and higher values outside the normal range. For example, to investigate the result when using a deficient gas flow or far too much current. When the weld tests have been completed, visual and physical strength tests will determine the weld quality.

### Evaluating and improving the original program algorithm

The original state of the program algorithm will be evaluated by testing the program code with different sizes and models of heat exchangers as the feature to support various sizes is presented in the code. After this has been assessed, the code will also be tested with an inclination of the heat exchanger, as the code has a feature to measure this inclination and correct for it in all robot movements.

To compare the two original methods of calculating the center reference point, both methods will be used to perform a series of welds, after which the accuracy and program time duration for these series of welds are analyzed in the measuring lab at SWEP.

After the original program algorithm is thoroughly evaluated and understood and the accuracies have been measured, an attempt to improve any part of this algorithm will then be made.

A detailed flow chart of the original algorithms will also be created. After the full control structure between the PLC, IRC, HMI, and HBS welding unit has been completed, the RAPID program flow chart will be extended with the additions made in the thesis work.

### Controlling the automated welding process

To control the welding process in a fully automated solution, a Siemens SIMATIC S7-1200 programmable logic controller (PLC) will be used, together with a Siemens KTP400 Basic Human-Machine-Interface (HMI). The PLC and HMI will be programmed in Siemens TIA Portal and connected to the robot controller using the PROFINET communication standard to ensure fast data delivery. From the HBS welding unit, three welding parameters will be exported: weld current, weld duration, and gas flow. Since the used HBS model (Visar 1200) does not have any support to transmit these parameters to an external controller, separate sensors will be used to measure the current and gas flow continuously. A complete state machine diagram of the finished PLC program will also be made.

## Enabling a fail-safe system

To implement a fail-safe system for the welding unit, the HBS Visar 1200 model operating manual will be studied. The welding unit is known to have an internal monitoring system that can be used to abort a weld if any parameter is outside the chosen tolerances.

## Designing and implementing the HMI

The HMI will be designed and programmed in TIA Portal. Discussions will be held with an automation specialist at SWEP to gain feedback on the chosen features and controls to implement in the HMI, until a satisfactory level of control for the automated solution of a work order is reached.

# 4 Analysis

This chapter describes the thesis work performed, how it was performed, and why it was performed in that way, together with all decisions taken. This includes completed analyses, tests, assessments, programming, and program design.

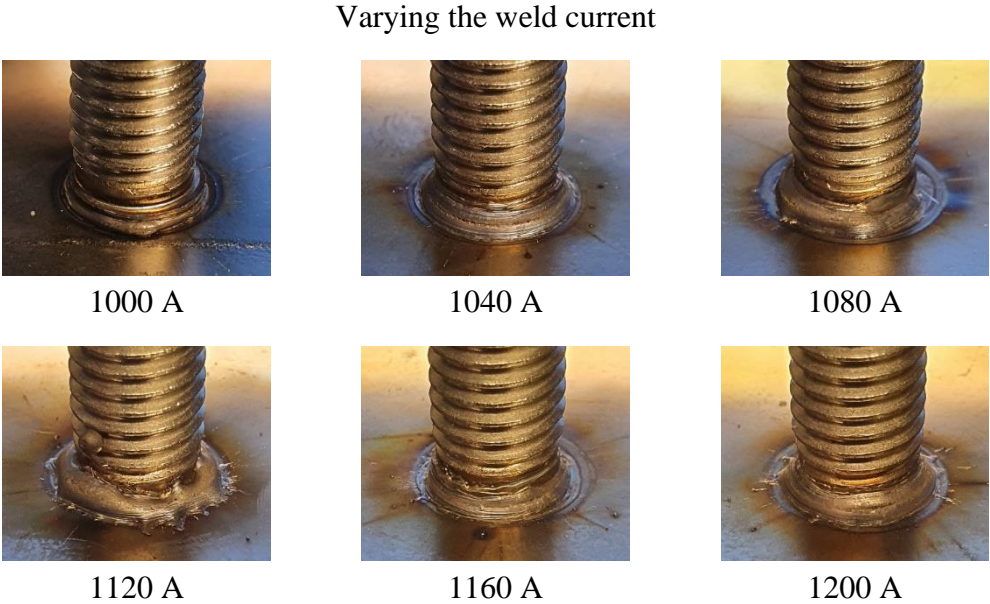
## 4.1 Performing quality analysis of the automated process

To understand how and why the different welding parameters each affect the quality of a weld, a series of tests were performed for each parameter. In this section, the completed tests will be described in detail.

### 4.1.1 Performing the tests

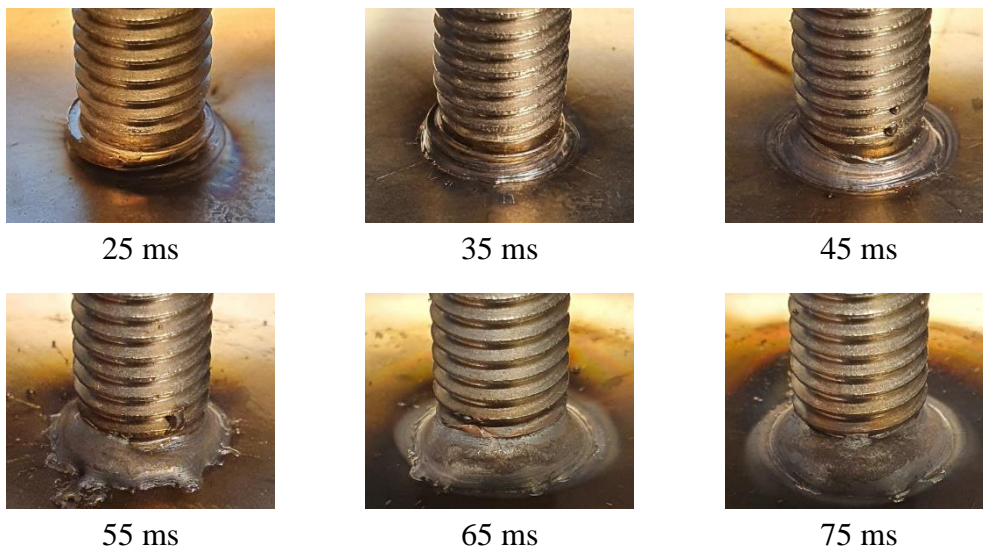
In the manual welding process used today in production, the amount of current typically used during a weld is around 1200 A during 50 ms. After a dialogue with one of the production engineers at SWEP when planning the welding tests, a suitable parameter range to test was determined to be from 900 A to 1200 A of current and a weld time duration from 25 ms to 75 ms. The gas flow is safe to test on the full available range of the gas tank, from no flow up to 25 liters/min.

To get a reasonable degree of resolution, six discrete values were tested for the current and the welding time, and three values were tested for the gas flow. When varying one parameter, the other two were kept static at the typical values mentioned that are used in production. Figure 32, Figure 33, and Figure 34 show the parameter values chosen and their resulting welds.



*Figure 32: The weld quality tests, varying the weld current*

### Varying the weld time



*Figure 33: The weld quality tests, varying the weld time*

### Varying the gas flow



*Figure 34: The weld quality tests, varying the gas flow*

## 4.1.2 Visual and physical inspection

To examine the resulting welds, a document was provided by SWEP describing their standard operating procedure<sup>[15]</sup>, SOP, performed by the operators of the manual welding process. First, a visual test was conducted for all performed tests by comparing them with a list of reference pictures of typical weld appearances. The list demonstrated what a successful weld looks like as well as faulty welds, including a partial weld, irregular weld collar, too large weld collar, pores in the collar, lack of penetration, etc. With each type of appearance, an assessment and recommended corrective action was given for the kind of fault. After the visual test was completed, a physical bending test and torque test were performed on each welded bolt.

The torque test was performed using a torque wrench, as seen in Figure 35. When this tool is used to apply torque to a bolt, the wrench stops applying a force when the rated torque of 9 Nm is reached. This maneuver was applied ten times on each bolt, in two perpendicular directions, to account for any asymmetrical partial welds that might be robust only in a certain direction. If the bolt is visually seen to have been bent after the torque test, the bolt is said to have failed the test. If no change is seen, the bolt passed the torque test.

If a bolt passed the torque test, a bending test was also performed using a bend test equipment, as seen in Figure 36. This tool was used to forcibly bend the welded bolts to around 30 degrees relative to the surface. If the bend test did not cause a fracture, tearing of the heat exchanger surface, or completely detached the bolt, the bolt is said to have passed the bend test.



Figure 35: Torque wrench used in the physical tests



Figure 36: Bend equipment used in the physical tests

## 4.2 Extending the original program algorithms

### 4.2.1 Studying the original program algorithms

In order to work with and later continue the development of the original robot program, its code was studied early on until the algorithms were fully understood. Then, before operating the program, the necessary skills for operating the robot were first learned. This included safety around the robot cell, how to use the FlexPendant as a valuable debugging tool in manual mode, controlling the gas tank, and general good practices of operating the robot. The program was then executed and thoroughly tested in the physical robot cell, including tests with different sized heat exchangers as well as with heat exchangers placed with a significant incline. In Figure 37, the inclination of the heat exchanger can be seen to be accounted for by the robot. In Figure 38, the calculated center point of a larger model is shown by the touch probe.

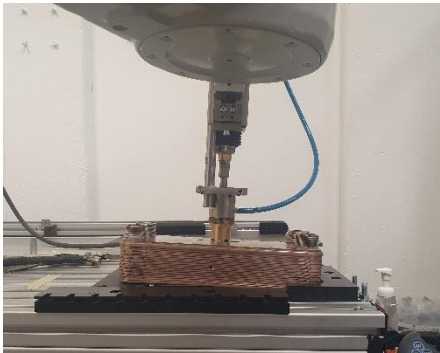


Figure 37: Inclination of a heat exchanger being accounted for

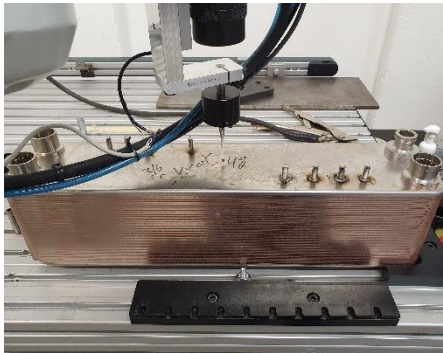


Figure 38: The calculated center point of a larger model size



## 4.2.2 Detailed flow chart

A detailed flow chart was created for the original code to further study and document the original program algorithms of the semi-automatic solution, including the two different methods for calculating the center reference point. In this flow chart, the main program flow and the essential RAPID procedures are explained in a level of depth and complexity chosen to best assist future programmers in using or further developing the program code. After the automated solution from the PLC and HMI was completed, the flow chart was extended with the changes and additions made in the thesis work. Appendix A shows a printout of the finished flow chart.

## 4.2.3 Measuring the accuracy of the two methods

To compare the two methods of calculating the center reference point and orientation, three bolt welds were performed using each method. The full measurement procedure and calculation of the center reference, as described in Subchapter 1.3.9, were executed again before each weld. If the center had been calculated only one time and used to position all bolts, they would all be affected by the same eventual positional error of the center position. After each weld, the observed measuring time was read on the FlexPendant and logged. The reason for performing three similar welds for each method was to ensure that the accuracy and measuring time of each method were consistent.

The six welded bolts were then brought to the measuring lab at SWEP. There, a unique measuring machine, as seen in Figure 39, was used to measure the actual positions of the bolts with the incredibly high accuracy of one micron, i.e., within 0.001 mm. This high level of accuracy is achieved by using a touch probe much like the one seen in Figure 14 but with a much finer probe diameter and sensitivity. Starting from the bottom of the bolt, the probe touches and steps up and around the actual bolt threads towards the top of the bolt, resulting in the very fine tolerance of the machine. Any inclination of the bolt will then be accounted for, which would not be the case if, for instance, a top-down photo of the bolts was used to measure the positions, as the top of the bolt would be above the surface in a different position than the bolt base position that is sought after.

The theoretical positions chosen in the robot program of the six performed welds can be seen in Table 5, along with their actual positions precisely measured in the lab at SWEP. For each bolt, the difference between the theoretical and actual position on the x- and y-axis is calculated as the "deviation" of the bolt. When the separate accuracy of the x- and y-axis is of interest, as when investigating methods to improve the overall accuracy, these deviations were used. The final target tolerance at SWEP is 1 mm, i.e., all points must be within a circle of diameter 1mm with its origin in the nominal weld position. In Table 5, a position being well within tolerance is marked in the color green, a position very close to being outside the tolerance as blue, and a point well outside tolerance in red.



Figure 39: The machine used in the lab to measure the bolt position accuracy

### 4.3 Creating the control structure

#### 4.3.1 Using a Siemens S7-1200 PLC as a master controller

While the RAPID welding program is running and is in need of a weld, the robot controller must signal to activate the HBS welding unit program to perform a stud bolt weld. To implement this crucial step of communication and control of the weld gun power, the PLC will be used as a master controller, i.e., have complete control of the program flow and single-handedly control the welding unit. Figure 40 shows the complete control structure.

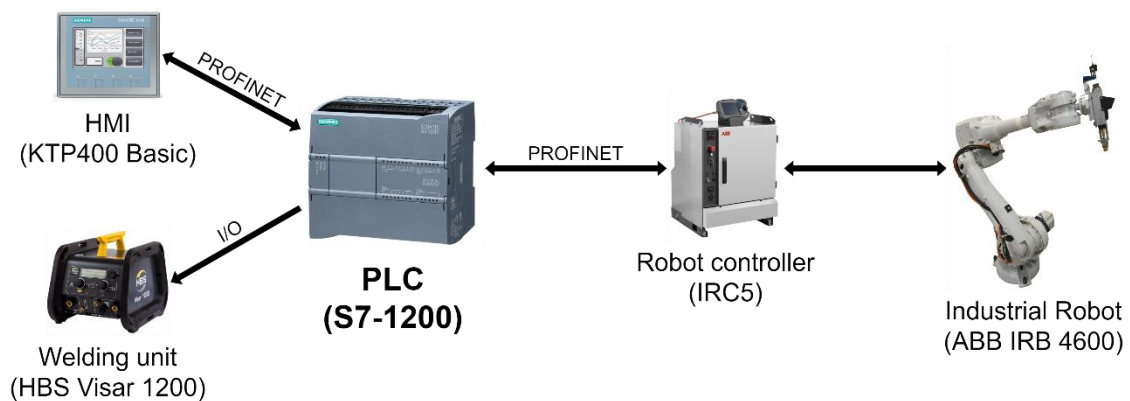
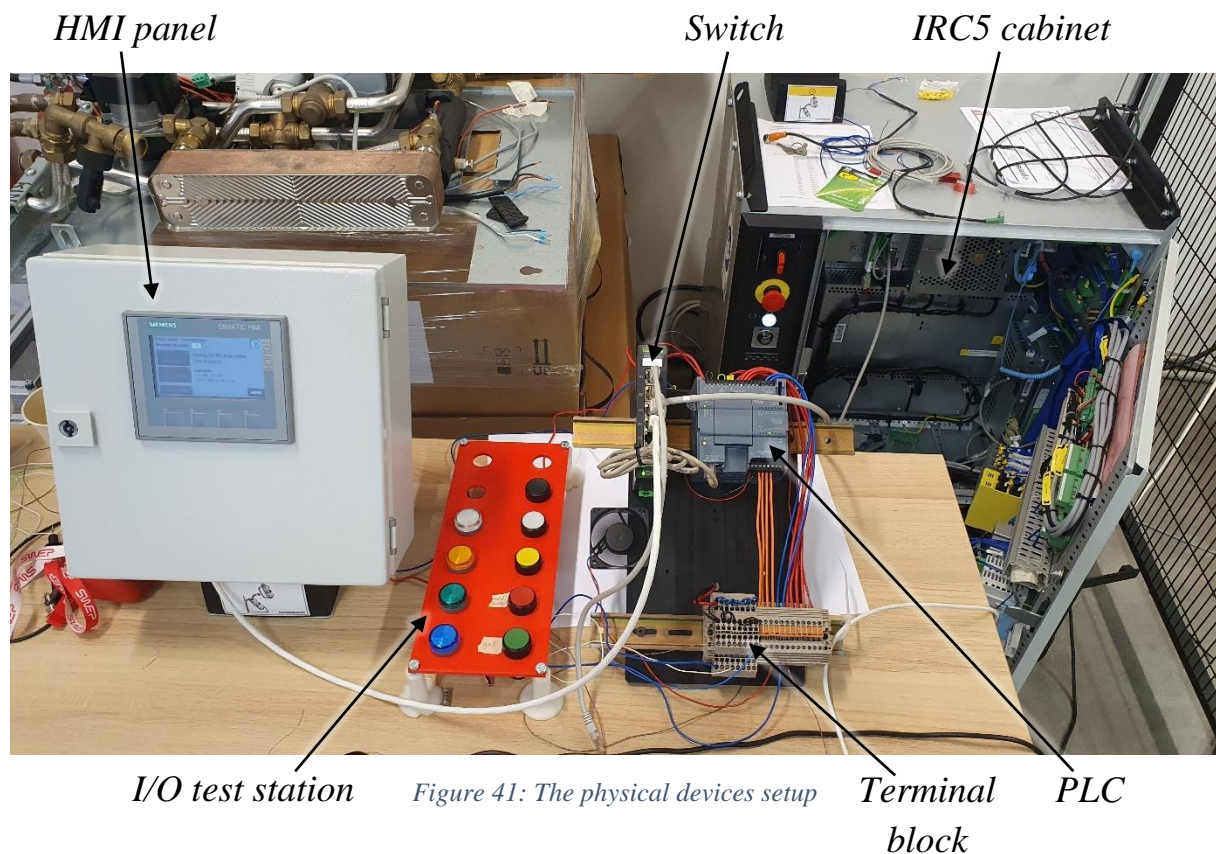


Figure 40: The chosen control structure. Images from [6], [7], [8] and [9].

### 4.3.2 Setting up the PROFINET

Figure 41 shows the physical devices and setup next to the robot cell. The PLC, HMI panel, and IRC are connected to a network switch via Industrial Ethernet, IE. The digital inputs and outputs of the PLC are connected to a terminal block, where external I/O communication with the PLC can be made.

For the connection between the PLC and IRC5 robot controller, a PROFINET link was set up in TIA Portal. PROFINET is a commonly used communication standard in the automation and control industry. It uses Industrial Ethernet on the OSI data link layer and is designed for collecting and delivering industry device data faster than alternative standards. To use PROFINET, an XML file coded in GSDML<sup>[14]</sup> (General Station Description Markup Language) containing the device information of the IRC must be exported from RobotStudio and imported to TIA Portal when programming the PLC. The GSDML file describes the robot controller's network capabilities, robot parameters, diagnosis data, and other information needed in TIA Portal to set up and use the PROFINET link.



### 4.3.3 Importing the welding parameters to the PLC

As stated before, the welding parameters (current, weld time, gas flow) of each weld in the finished automated solution must be exported to the PLC to be monitored. During program execution, the actual values of these welding parameters must be compared to the target values chosen in the HBS welding unit program. This comparison will act as a safety check, instead of blindly trusting the HBS welding unit. If any parameter is found to be outside its allowed

range, the program execution must be stopped until the HBS welding unit has been inspected and the values have returned to their allowed ranges.

Initially, the plan was to change from using the Visar 1200 to a model with internal support for exporting the welding parameters. However, after some discussion, it was chosen to use an alternative method of using separate sensors to measure these parameters instead. Sensors were ordered by SWEP, but it was quickly realized that they would not arrive in time for the thesis work. Therefore, the sensor's analog signals were simulated using the I/O testing station.

#### 4.3.4 Giving the PLC complete control of the weld power

In the initial program code, the IRC controlled the welding power directly. Now, when the IRC needs a weld, it will instead signal this to the PLC using a signal "want\_weld" on the PROFINET, and then wait for the PLC. The PLC will then perform the weld and alert the IRC of its completion using another signal "weld\_done" and, it too, will wait until the IRC acknowledges the completion by lowering its initial signal "want\_weld". This "handshake" will ensure that both the IRC and PLC program will terminate together if any one of them declares an error. In Figure 42, the original and modified RAPID code that is executed when a weld is needed are shown to illustrate this handshake from the IRC perspective.

```
SetDO Local_IO_0_D08,1; !activate the HBS welding program
WaitTime(1.75); !wait for program to finished, and let bolt cool off
SetDO Local_IO_0_D08,0; !deactivate the HBS welding program
```

original code: IRC operates the weld directly.

```
SetDO PN_do1_1, 1; !ask the PLC to perform a weld
WaitDI PN_di8_1, 1; !wait until PLC is done
SetDO PN_do1_1, 0; !send back ACK of completion to PLC
```

modified code: let the PLC control the weld

*Figure 42: RAPID code example, moving control of the weld power to the PLC*

#### 4.3.5 Programming the PLC to control an entire work order

After the control structure had been set up and the PLC had been given control of the welding power, the PLC was first programmed to operate the original welding program, i.e., using the "handshake" to control the weld power when requested by the IRC.

Then, as the HMI was being designed to control a work order, the PLC program was extended to handle multiple recipes chosen from the HMI, sending the model dimensions and recipe to the IRC, and performing the recipe welds on both the P- and F-side. Needed functions for the HMI operator were also implemented, such as a cycle-stop, quick-stop, and a work order reset. To illustrate the finished PLC program state machine, a state diagram was created. In Appendix B, the finished state diagram of the PLC program is presented. Some examples of the signals created on the PROFINET are presented and described in Table 1 as seen by the PLC, with outputs going to the IRC robot controller.

Table 1: Examples of PROFINET signals created

Signal name	Signal type	Description
irc_is_online	Digital input	Notifies status of the manual mode safe guard.
plc_state	Word output	Informs the IRC of the present program state. Used to synchronize the two programs.
want_weld	Digital input	Informs that a new weld is needed.
weld_done	Digital output	Notifies the completion of a weld.
in_start_pos	Digital input	Notifies that the robot has returned to the start position.
hmi_cycleControl	Digital output	Main control of program flow. Start and resume production, cycle-stop during program execution, stop when loading a new unit, and when flipping unit to F-side.
hmi_quickStop	Digital output	Quick-stops the program: immediately halts all robot movement, continues when operator resumes production.
hmi_newOrder	Digital output	Notifies the robot to return to the start position and wait for a new recipe import. A new work order is initiated.
hmi_pos_sent	Digital output	When sending the recipe, informs that another bolt position has been sent.
sent_pos_x	Word output	x-coordinate of the sent bolt position.
sent_pos_y	Word output	y-coordinate of the sent bolt position.
irc_pos_received	Digital input	IRC acknowledges that the bolt has been saved, and is ready to receive the next bolt position.
recipe_send_done	Digital output	Informs that the recipe has been fully sent.
hmi_need_flip	Digital output	Informs that the unit needs to be flipped before the next weld.

The waiting period in the PLC program state 21, as seen in the state diagram in Appendix B, reached when the weld current is gone, is realized using an on-delay timer (TON) in ladder code, as seen in Figure 43. The timer is started using the PLC tag "weld\_timer\_on". When the defined time "weld\_wait\_ms" has elapsed, the output tag "weld\_timer\_done" alerts the main program.

The measurement in state 21 of the weld current duration is realized using an on-delay retentive timer (TONR) in ladder code, as seen in Figure 44. The timer is started using the tag "current\_timer\_on" and reset with "current\_timer\_reset". The elapsed time is saved to the tag "time\_measured".

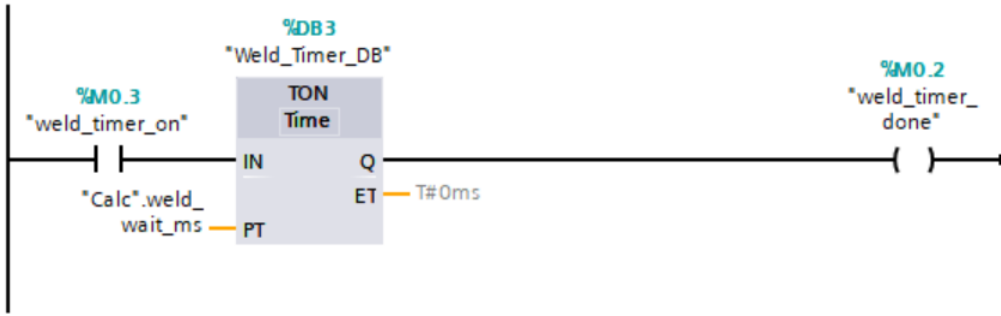


Figure 43: LD code used to generate a delay for cool-off after each weld

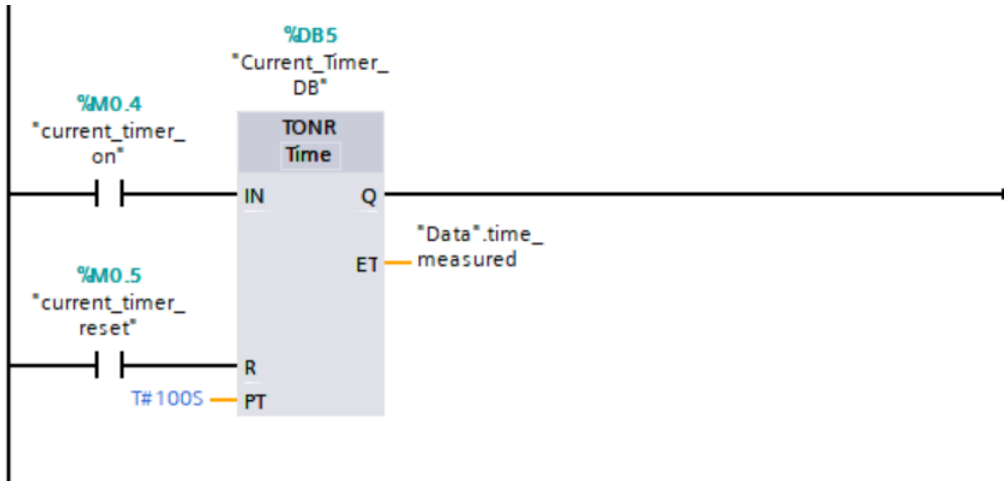


Figure 44: LD code used to measure the weld current duration

The measurements in state 21 of the analog inputs from the simulated sensors are realized using a function in ladder code, as seen in Figure 45. For a Siemens PLC, the maximum value of an analog input is 27648, given by the maximum allowed input of 10 V. In order for the PLC program to support a different value range output by the current or gas sensor, the analog value is first normalized and then scaled to match the sensor value range. For example, if a sensor outputs a range of 0 to 5V representing 0 to 1500A of current, the analog signal must be scaled by a factor of two in order to match the PLC range of 0 to 10V, as to make use of the full resolution.

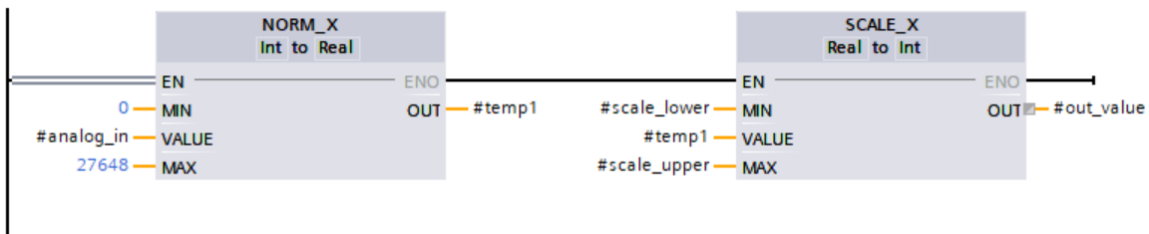


Figure 45: LD code used to measure the sensors analog inputs

# 4.4 Enabling a fail-safe system for the welding unit

## 4.4.1 Studying the Visar 1200 manual

As the Visar 1200 model had never been used before in the production at SWEF, it was not fully known how to create new welding programs, define tolerances, or export these programs to other welding units. For this reason, the manual was studied to learn more. The obtained knowledge is presented in the following paragraphs. As suggested by HBS in the manual, any automation of a welding process should undergo two phases: an initial "learning and validation phase" to prepare the welding unit program, and later an "application phase" for when it is put to use in production.

In the learning phase, the "SYNERGY" mode should be used to create a new welding program. In this mode, numerous properties of various welding components may be selected. For example, the stud bolt type, bolt material and diameter, but also more specific properties such as bolt insertion depth, spring force, workpiece thickness, shielding gas composition, parameter tolerances, and more. When this is done, five reference welds that reflect the welding task must be performed. After all properties have been chosen, and five reference welds have been completed, the HBS welding unit will then automatically calculate and choose welding parameter values that it sees as ideal for the welding process. Figure 46 shows the SYNERGY mode as seen by the operator on the display.

In the application phase, the welding program created in the SYNERGY mode may be used in production. The welding unit will then automatically compare the actual values during each weld to the target values and tolerances automatically chosen in the learning phase. If any parameter is seen outside of its tolerance, the program will be stopped immediately. The program must then be manually reset using a button on the welding gun.

The newly created welding program can then be saved to a USB drive in MENU → USB MENU. Here, previously saved programs can also be uploaded.

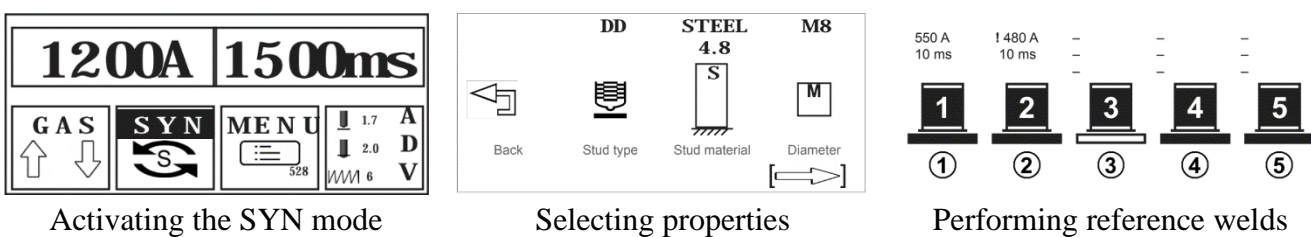


Figure 46: The HBS "learning phase". Images from the Visar 1200 manual.

Another known feature of the Visar 1200 model was the possibility to log and export data after each performed weld, including a quality assessment, average parameter values (current, voltage, time, etc) as well as each parameter plotted over time. As learned from the manual, this is configured by enabling any of the three "USB commands" in the USB MENU: "Documentation" for exporting average values, "Oscillogram" for exporting the values plotted over time, and "Lock" for preventing further program changes being made on the unit. Using these features, the analog values sent from the current and gas sensor to the PLC could be compared and verified to be accurate. However, it was realized that these features were not

provided by HBS by default and had to be separately requested and purchased. As there was not enough time left in the thesis project to request and wait for this, the sensors could not be properly verified to be accurate. In any case, as already described, the sensors were ordered by SWEP but did not arrive during the thesis work anyway. Therefore, after the sensors have arrived and the additional software has been requested from HBS, this comparison and verification of the sensors should be performed by SWEP.

#### 4.4.2 Creating the welding program

As target values for the welding parameters had already been decided from the quality analysis within the thesis work, the SYNERGY mode could not be used as it would change these values to its own calculated ideal values. These "ideal" values were often a much lower current, around 500 A, but with an increased welding time around 150 ms, and so might offer a similar weld as the typical values used at SWEP of around 1200 A during 50 ms. However, it was decided not to use the automatically chosen target values from the SYNERGY mode. Therefore, the welding program created in the SYNERGY mode was later edited: the target values were manually edited using two turning knobs underneath the display, and the "advanced menu" was used to edit the tolerances.

### 4.5 Designing and programming the HMI

This chapter describes the completed work of designing, evaluating, and programming the HMI, as well as the PLC and RAPID programs used with it. Chapter 5.3 presents the finished HMI design and features.

#### 4.5.1 Defining the needed features

After the obtained knowledge of how the original robot program algorithms operate, how the different welding parameters can affect the weld quality, and what type of bolt configurations are needed at SWEP, the following list of required features to implement in the HMI was created:

##### **Controlling the welding program and work order:**

- Creating a new work order containing a bolt configuration recipe and the number of units in the order.
- Cycle-stopping the program execution at any time, i.e., finishing the active cycle, for example picking up a bolt, then pausing the program, until the cycle-stop is ended.
- Quick-stopping the program execution at any time, i.e., immediately halt all robot movements until the quick-stop is ended.
- Stopping and waiting for the next unit to be loaded, or stopping when a unit needs to be flipped to the F-side.

##### **Supporting different bolt configurations:**

- Creating new and choosing between saved bolt configurations ("recipes") and automatically sending the recipe data to the RAPID program.



- Automatically sending the expected model dimensions to the RAPID program. The measuring algorithms need these to know where to start searching for the sides using the touch probe.

#### **Monitoring and logging:**

- Displaying the last measured weld data and parameters, i.e., the x- and y-coordinates, as well as the current, gas, and weld time measured by the sensors.
- Displaying the present state of the master PLC state machine. For example, "Waiting for production start..", "Program running..", "Welding..", "Program has cycle-stopped..", "Unit needs flip to F-side", etc.
- Displaying the active bolt configuration recipe, number of units in the work order, and units finished so far.

#### **Optimizing weld quality:**

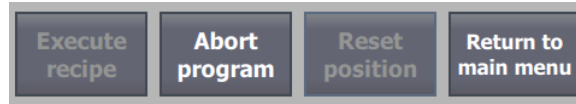
- Selecting tolerances of the welding parameters, i.e., current, weld duration, and gas flow.
- Automatically alert the operator if any weld parameter is measured outside of the chosen tolerances.

### 4.5.2 Designing the HMI

After all needed features of the HMI had been decided, a fitting design that implements all features together in a practical and efficient way was developed before starting any programming in TIA Portal. This was started by first doing some research to learn common good practices and general tips for designing a good quality HMI. With this, a series of guidelines to follow when designing the HMI was decided:

- Show only the necessary information that the operator needs to see to operate the work order. If presenting too much or unnecessary information, it might distract the operator from monitoring the values that matter.
- The present state of the PLC state machine must be apparent at all times, in order to provide a situational awareness to the operator. For example, it might be too vague to only distinguish between "program running" and "program not running" and better to be more precise on what exact state the program is currently in.
- When needed, draw extra attention of the operator by using suitable graphics, symbols, sizes, colors, and blinking animations. For example, while the weld gun is active or when any weld parameter has fallen out of tolerance.
- "Lead the way" to the operator by clearly showing what actions are allowed in the moment. For example, if the program has been aborted and is resetting, the program may not be started again until the robot has reached the start position. Therefore, do

not only disable the "Execute recipe" button while waiting but also make it transparent to let the operator know that it may not be used yet. As for the example seen in Figure 47 of an early version of the final HMI program, it becomes easily understood what actions are allowed in the moment:



*Figure 47: Designing an HMI: lead the way by only highlighting allowed actions*

- Use a reasonable number of different views but minimize the number of clicks needed by the operator to reach any function. While controlling a work order, only a single view should be needed by the operator.

### 4.5.3 Programming the HMI in TIA Portal

Many of the more simple or standard features often needed in an HMI can be implemented directly in TIA Portal without using external scripts. To show an example within the thesis work, consider the simulated HMI view as seen on the left side in Figure 48, showing an early version of the final HMI design. Here, the goal was to use the symbolic I/O field element to indicate to the operator the present state of the PLC state machine and IRC program. When in an idle state, the text "Waiting for operator" was presented, and when in any of the measuring or welding states, it was presented as "Program running", etc. The text color and nearby state icon were also programmed to change with the program states.

To implement this feature in an HMI, the existing tools and settings in TIA Portal are enough. First, the I/O field is marked and navigated to the "Animations" tab of its properties. Here, a new "Appearance" animation can be created, and the PLC tag "program\_state" within the Data block "Data" was connected to the animation. Now, different range values of the connected variable can be defined as "animation states" to have the text color automatically change for different program states, as desired. To also change the actual text, a TIA Portal "Text list" can be created where, similarly, different text data can be connected to different value ranges of the same PLC tag.

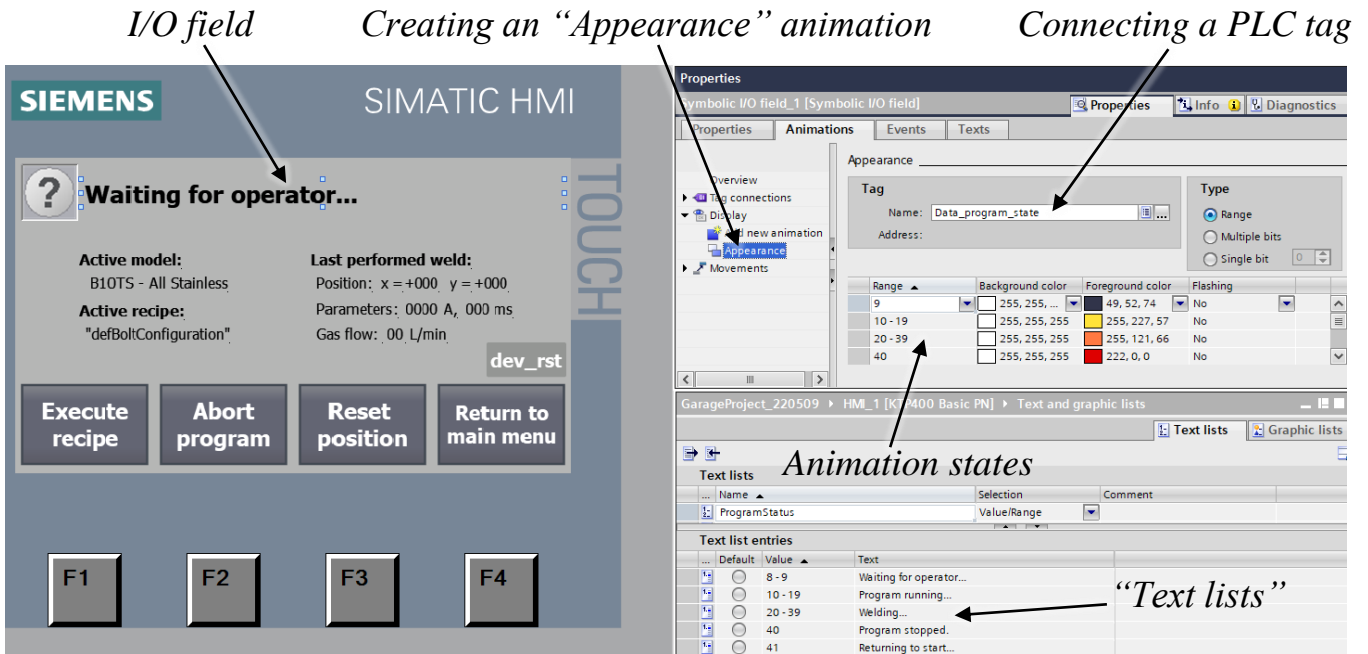


Figure 48: Programming example of an early version of the final HMI design

#### 4.5.4 Additions made to the RAPID program code

This section presents a few examples of all additions made to the RAPID program within the thesis work, extending the original semi-automated solution, to implement the PLC and HMI control for a fully automated solution. The created PROFINET signals were described in Subchapter 4.3.5.

#### Cycle control

A single digital signal on the PROFINET, "hmi\_cycleControl", is used to stop the RAPID program execution when the operator orders a "cycle stop", when a new unit needs to be loaded, or when a unit needs to be flipped. This is implemented by inserting a type of "checkpoints" before each new defined "cycle" of the welding program, where the program execution pauses and waits for this signal to be reset before continuing. This was done by using the RAPID instruction WaitDI. Some of the chosen "cycles" of the welding program were: the measuring of the three surface points, the measuring of the side points, picking up a bolt, welding a bolt, etc. For example, if the operator orders a cycle stop during the measurements of the surface, the program will finish that cycle and, only then, stop.

#### Flipping the unit

After each new bolt is welded at the specified positions of the imported recipe, the RAPID program checks if it is needed to flip the unit before the next bolt, notified by the PLC using the signal "hmi\_need\_flip". The added RAPID program code for this can be seen in Figure 49.

```

IF PN_di17_1 THEN
  !PLC has signaled to FLIP the HE, i.e. switch between P- and F-side. PLC currently waiting for robot to reach start pos

  safePath := Offs(CRobT(\Tool:=auto_gun_v2, \WObj:=wHEPlane),0,0,150); !create safe path
  MoveL safePath,vSlow,z10,auto_gun_v2,\WObj:=wHEPlane; !move weldgun safely away from surface

  MoveJ pMHome,vMedium,z100,tProbe\WObj:=wobj0; !return to start position
  SetDO PN_do5_1, 1; !alert PLC that start position was reached
  WaitTime 0.5;
  SetDO PN_do5_1, 0;

  WaitDI PN_di12_1, 1; !cycle checkpoint, as to ask operator to flip the unit before continuing

  !the HE has been flipped, and so needs to be measured again
  IF NOT Measure() THEN
    TPWrite("Measuring after HE flip: failed");
    ExitCycle; !abort
  ENDF

  !new side was measured successfully, continue welds
  MoveJ pWeldHome,vFast,z200,auto_gun_v2\WObj:=wobj0;

ENDIF

```

Figure 49: RAPID code example: perform a safe return path and ask the operator to flip the unit to F-side

## Quick-stop

To immediately halt all robot movements when the operator orders a quick-stop, an interrupt routine (TRAP routine) was created using the CONNECT instruction and defined to execute when the signal “hmi\_quickStop” had a rising edge by using the ISignalDI instruction. The added RAPID program code for this interrupt routine can be seen in Figure 50.

```

TRAP quickStopInterrupt
  !quick-stop was activated by HMI operator

  StopMove; !stop all movements (without ClearPath, so path may be continued after quickstop ended)
  WaitDI PN_di13_1, 0; !wait for operator to resume
  StartMove; !allow movements again

ENDTRAP

```

Figure 50: RAPID code example: the quick-stop

## Importing a new recipe

To send a recipe to the RAPID program, two possible solutions were considered. The initial idea was to place each PLC recipe tag on its own PROFINET address, as available addresses were plentiful and each recipe only has about twenty data points. In this way, the recipe data can be kept updated and read at any time by the IRC. However, changing the active recipe during a work order is not permitted, so this method would unnecessarily occupy addresses with static data. Instead, a method of only using two analog output addresses on the PROFINET was used. When the operator initiates a new work order on the HMI, the PLC sends each recipe position one by one to the IRC, using the two addresses for each x- and y-position.

A type of “handshake” is used to ensure that each recipe position is received by the IRC. The PLC writes the first bolt position on the two addresses on the PROFINET, signals this with “hmi\_pos\_sent”, then waits. After the PROFINET has had time to update, the IRC notices and saves the first position. Then, the IRC acknowledges this to the PLC on “irc\_pos\_received”. The PLC may then send the next bolt position. The IRC will continue to wait for another bolt

until “recipe\_send\_done” is set, marking the end of the recipe. The RAPID procedure programmed for receiving a recipe can be seen in Figure 51.

```
PROC ImportRecipe()

  SetDO PN_do6_1, 0; !nothing has been received yet
  recipeCount := 0; !reset counter

  WaitTime 0.5;

  SetDO PN_do6_1, 0;
  WaitGI PN_GI_6, 90; !Synchronize with PLC program flow

  !Import the full recipe from the PLC and HMI
  WHILE NOT PN_di16_1 DO

    WaitTime 0.5; !wait for PN

    WaitDI PN_di15_1, 1; !wait for new bolt

    !new bolt received. read and save it.
    weldingPositions{recipeCount+1}.x := PN_GI_1 - analogSignOffs;
    weldingPositions{recipeCount+1}.y := PN_GI_2 - analogSignOffs;

    SetDO PN_do6_1, 1; !ACK that the pos was received
    WaitDI PN_di15_1, 0; !wait for PLC ACK
    SetDO PN_do6_1, 0;

    recipeCount := recipeCount + 1;

  ENDWHILE

  SetDO PN_do6_1, 0; !reset before next recipe

ENDPROC
```

*Figure 51: RAPID code example: importing a recipe*

## 5 Results

This chapter describes all results from performed analyses and tests, the finished PLC program, the final HMI design, and the collective result of the automated solution.

### 5.1 Welding parameters for optimal weld quality

The resulting bolt quality from the visual and physical tests is presented in Table 2, Table 3, and Table 4. Even though a wide range of values were chosen for each parameter, it was seen that almost all bolts passed the torque and bending tests. This was brought up to a production manager, who confirmed it to be an expected result. The visual test results were seen to vary significantly between the welds, and so became the primary source to draw conclusions from.

When varying the current, as seen in Table 2, it was seen that using a too low value resulted in only a partial weld even though the typical weld period of 50 ms was used. When the current was too high, it was seen to result in a too large collar and or weld splatter. This test concluded that a suitable level of current to use during the default time of 50 ms is around 1040 to 1080 A.

When varying the welding time, as seen in Table 3, it was similarly seen that using a too low value resulted in a decrease in the total weld energy, resulting in only a partial weld. When using a too long welding time, it was seen to result in a too large collar with lots of oxidation. This test concluded that a suitable welding time to use with the default current level of 1200 A is around 45 ms.

When varying the gas flow, the goal is only to remove all oxygen near the weld spot. If any oxygen remains, oxidation on the heat exchanger surface will occur. For this reason, it was seen to be enough to test only three levels of gas. As predicted, the main difference between using no gas, and 25 liters/min, was the very apparent oxidation level. When using 15 liters/min, only some oxidation was seen. Even though the current and weld time were held at their typical values of 1200 A during 50 ms, the weld collar was also seen to be affected by the level of gas. This test concluded that a suitable level of gas flow to remove all oxidation is around 25 liters/min.

Within the thesis work, the main objective was to understand how each weld parameter affects the weld results, and from this, decide on a suitable set of parameters to upload to the HBS welding unit. A good set of parameters was estimated to be around 1100 A during 45 ms, with a gas flow of at least 25 liters/min. However, since only one parameter was varied within a test, to only six different values, each test only performed once, the weld results should only be used as a guidance for understanding how the parameters affect the weld. This is because any set of welding parameters has been seen to have some variation in the weld result. To perform a more in-depth analysis, a wider range of discrete values should be tested, and most importantly, they should be tested more than once until consistent results are observed.

Table 2: Bolt quality results from the visual and physical tests, varying the current.

Current	Visual test result	Assessment	Torque test	Bend test
1000 A	Partial weld.	Too low weld energy.	Passed	Passed
1040 A	Regular collar. No visual defects.	Acceptable.	Passed	Passed
1080 A	Regular collar. No visual defects.	Acceptable.	Passed	Passed
1120 A	Large irregular collar.	Too high weld energy.	Passed	Passed
1160 A	Regular collar. Some weld splatter.	Too high weld energy, causing splatter.	Passed	Passed
1200 A	Regular collar. Weld splatter.	Too high weld energy, causing splatter.	Passed	Passed

Table 3: Bolt quality results from the visual and physical tests, varying the welding time.

Time	Visual test result	Assessment	Torque test	Bend test
25 ms	Partial weld. Collar off-center.	Too low weld energy.	Failed	-
35 ms	Partial weld.	Too low weld energy.	Passed	Passed
45 ms	Regular collar.	Acceptable.	Passed	Passed
55 ms	Irregular collar. Splatter.	Too high weld energy, causing splatter.	Passed	Passed
65 ms	Too large collar. Some oxidation.	Too high weld energy. Not enough gas flow, causing oxidation.	Passed	Passed
75 ms	Too large collar. Lots of oxidation.	Too high weld energy. Not enough gas flow, causing oxidation.	Passed	Passed

Table 4: Bolt quality results from the visual and physical tests, varying the gas flow.

Gas flow	Visual test result	Assessment
No gas	Large irregular collar. Full of oxidation.	No removal of oxygen, causing oxidation.
15 l/min	Irregular collar. Some oxidation.	Some oxygen was removed, decreasing oxidation.
25 l/min	Regular collar. No oxidation.	Full removal of oxygen, removing oxidation completely.

## 5.2 Accuracy of the two methods of calculating the center

### 5.2.1 Accuracy of the two methods

Table 5 shows the theoretical positions and actual positions, as well as the time taken for the six test welds using the two different methods. The measuring time is seen to be consistent in both methods, with the second method taking roughly 35% longer to measure the heat exchanger. This is expected, as the second method measures an additional three points compared to the five points measured by the first method.

The "True Position", TP, is defined from the deviation in the x- and y-axis as double the distance from the measured point to the theoretical point. This is the value that is checked against the defined tolerance. For example, a True Position of 0.5 mm means that the points are spread out within a circle of diameter 0.5 mm with its center in the theoretical position. At SWEP, the True Position tolerance is 1 mm, as seen visualized in Table 5 and Table 6.

As the second method uses more points, it was initially guessed to have higher accuracy than the first method, as more points perhaps could mean a lower impact on the result from each single inaccurate point. This hypothesis was seen to be incorrect as the first method, the one measuring fewer points, was shown to be just within the TP tolerance of 1 mm, as seen visualized in blue color, versus the second method, which was well outside the TP tolerance, as seen visualized in red.

With the initial guess that the second method should at least not be so inaccurate, this idea was brought up and discussed with a measuring engineer at the lab, describing the two methods used to calculate the center. The first method's algorithm was told to have an expected and realistic accuracy, but that the second method's algorithm should have performed better. Details regarding the algorithm of the second method were discussed, and some possible reasons for its inaccuracy were found. It was pointed out that when measuring two points on each side, as done in the second method, it is crucial that the two points are as far away from each other as possible. If not, if a short side point is incorrectly placed by some error "d", it will have a significant effect on the calculated lines and the resulting intersection point, as seen illustrated in Figure 52 in the color red. The same error "d" on a long side point would, however, not change the intersection as much, as seen illustrated in Figure 52 in green, as the error becomes relatively smaller on the long sides. On the long sides of the heat exchanger, the used distance was determined to be enough. On the short sides, however, a distance of only around two centimeters was used on some of the smaller heat exchanger models.

Therefore, in an attempt to improve the robot program, a new test of six bolts was performed, with the short side distance increased for the second method. The first method, however, is not affected by the illustrated problem as it only measures one point on each side. From the first set of lab results, it was also noticed that a consistent positional offset was present in both methods, most likely from some divergence errors of the robot joints, weld gun, and measuring probe. Therefore, an offset was also introduced in the program code to counteract this. The search speed of the probe was also reduced.



Table 5: Resulting position and time taken for the test welds

Method used	Bolt number	Measuring time (s)	Theoretical position (mm)	Actual Position (mm)	Deviation (mm)	
Method I (8 point)	1	50.56	x: -60 y: 10	x: -59.627 y: 10.324	x: 0.373 y: 0.324 TP: 0.988	
	2	50.28	x: -30 y: 10	x: -29.957 y: 10.401	x: 0.043 y: 0.401 TP: 0.806	
	3	50.54	x: 0 y: 10	x: 0.242 y: 10.326	x: 0.242 y: 0.326 TP: 0.813	
Method II (11 points)	4	68.27	x: 60 y: -15	x: 60.504 y: -14.619	x: 0.545 y: 0.694 TP: 1.765	
	5	68.17	x: 30 y: -15	x: 30.570 y: -14.456	x: 0.570 y: 0.544 TP: 1.575	
	6	68.30	x: 0 y: -15	x: 0.545 y: -14.306	x: 0.504 y: 0.381 TP: 1.264	

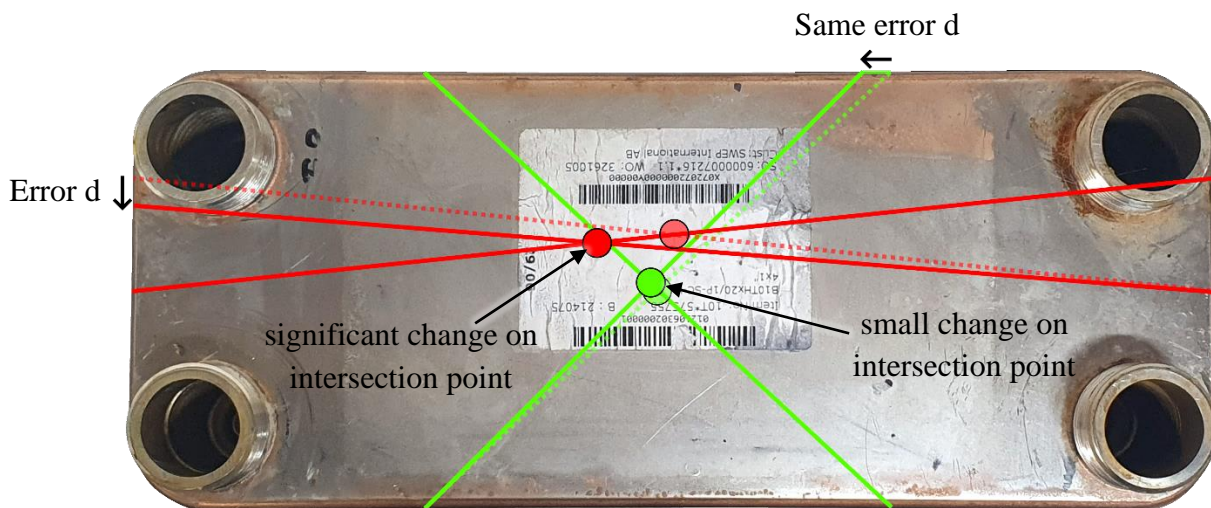


Figure 52: Graphical reasoning of why it is crucial to separate the short side points as much as possible (exaggerated error d)

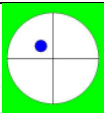
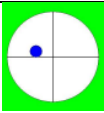
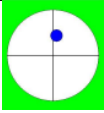
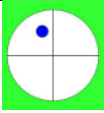
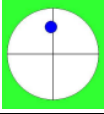
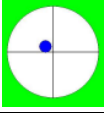
### 5.2.2 Accuracy of the two methods, after improvement attempt

After implementing the improvements into the RAPID program code, six new bolt welds were performed as before, with three bolts for each method. These new welded bolts were then taken to the lab and measured. Table 6 shows the resulting positions, times taken, and the new resulting accuracy of the two methods.

It was seen that the measuring time has increased for both methods. This was because of the decrease in search speed. Nonetheless, the measuring time is still increased by about 35% as before because of the three additional points measured by the second method.

With the improvements added, the accuracy of both methods was seen to have been greatly improved, as seen in Table 6. The two methods were seen to be consistently within the tolerance of 1 mm. As most of the positions are within the second quadrant, another slight change of the offset might improve the accuracy even further.

Table 6: Resulting position and time taken for the welds, after program improvements

Method used	Bolt number	Measuring time (s)	Theoretical position (mm)	Actual Position (mm)	Deviation (mm)	
Method I (8 point)	7	105.9	x: -60 y: 0	x: -60.133 y: 0.135	x: -0.133 y: 0.135 TP: 0.379	
	8	105.9	x: -30 y: 0	x: -30.186 y: 0.061	x: -0.186 y: 0.061 TP: 0.391	
	9	105.9	x: 0 y: 0	x: -0.081 y: 0.059	x: -0.081 y: 0.059 TP: 0.201	
Method II (11 point)	10	133.9	x: 30 y: 0	x: 30.039 y: 0.217	x: 0.039 y: 0.217 TP: 0.441	
	11	134.4	x: 60 y: 0	x: 59.883 y: 0.263	x: -0.117 y: 0.263 TP: 0.575	
	12	134.2	x: 90 y: 0	x: 89.979 y: 0.293	x: -0.021 y: 0.293 TP: 0.587	

### 5.3 The final HMI and automated solution

To implement all features of the HMI decided in Subchapter 4.5.1, it was decided that six HMI views were needed: a "Root screen", "Operator view", "Recipe view", "Settings", "Monitoring", and "Alarms". In this section, each view will be presented separately.

#### 5.3.1 The "Root screen"

This screen is opened by default when the HMI panel is powered on or reset. It lets the operator navigate to the other screens. On all other screens, a button labeled "home" is used to return to this root screen. In Figure 55, the "Root screen" is presented.

### 5.3.2 The "Operator view"

In Figure 53, the "Operator view" is presented. On this screen, the operator controls the automated work order process. At the top, the active recipe is presented, together with an input box where the operator may choose the number of units that the work order consists of. Initially, the top button is used to "Start production" and is later used to control the cycle-stop features, as well as letting the operator resume the program after flipping over a unit, or after loading the next unit. With the "Quick stop" button, the operator can immediately stop all robot movements until the quick stop is ended. The "New order" button is used to initiate a new work order by sending the active recipe to the IRC and resetting the PLC program. This button may be used during program execution as well, terminating the active work order and resetting the robot to its start position in a programmed safe path.

After each new weld, its position and measured weld parameter values are presented under "Last weld". These values are then compared to the chosen tolerances, aborting the work order if any parameter is measured outside its tolerances.

To inform the operator of the present program state and when some action is needed, a text box and assisting image are used. For example, when the PLC program is in the first state, the text "Waiting for IRC to go online" is shown. In states 7 or 8, the text "Ready for production start". In state 9, the text "Load new unit then resume", and so on, together with a suitable image.

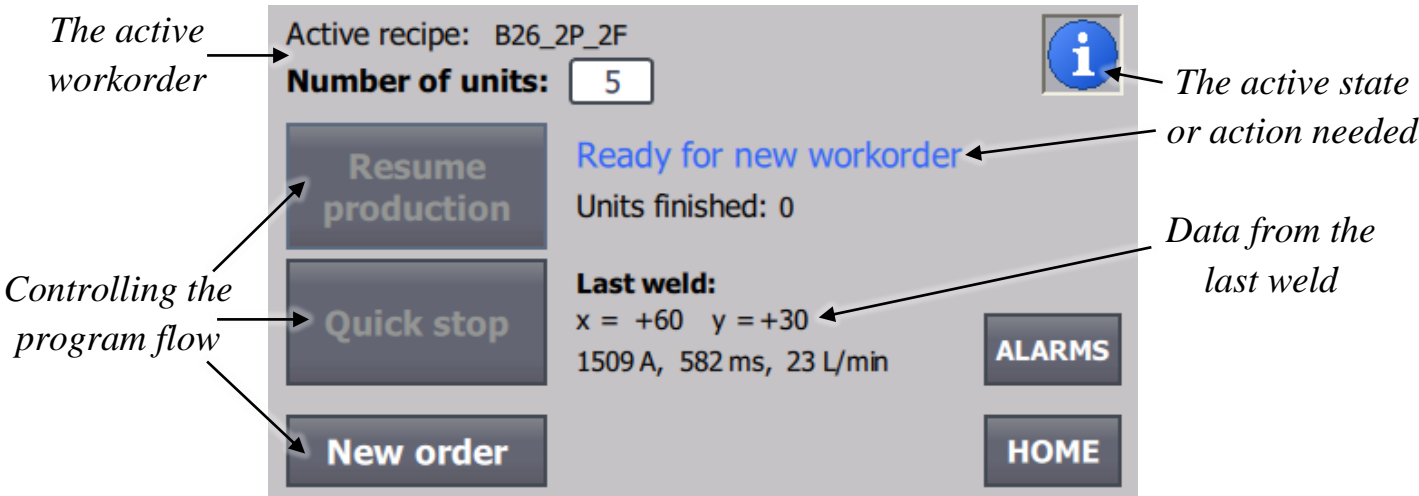


Figure 53: The final HMI - "Operator view"

### 5.3.3 The "Recipe view"

In Figure 54, the "Recipe view" is presented. In this screen, the operator can create, save, delete, and modify recipes. After a new recipe has been created or a saved one has been selected, it can be chosen as the active recipe of the work order with the upload button.

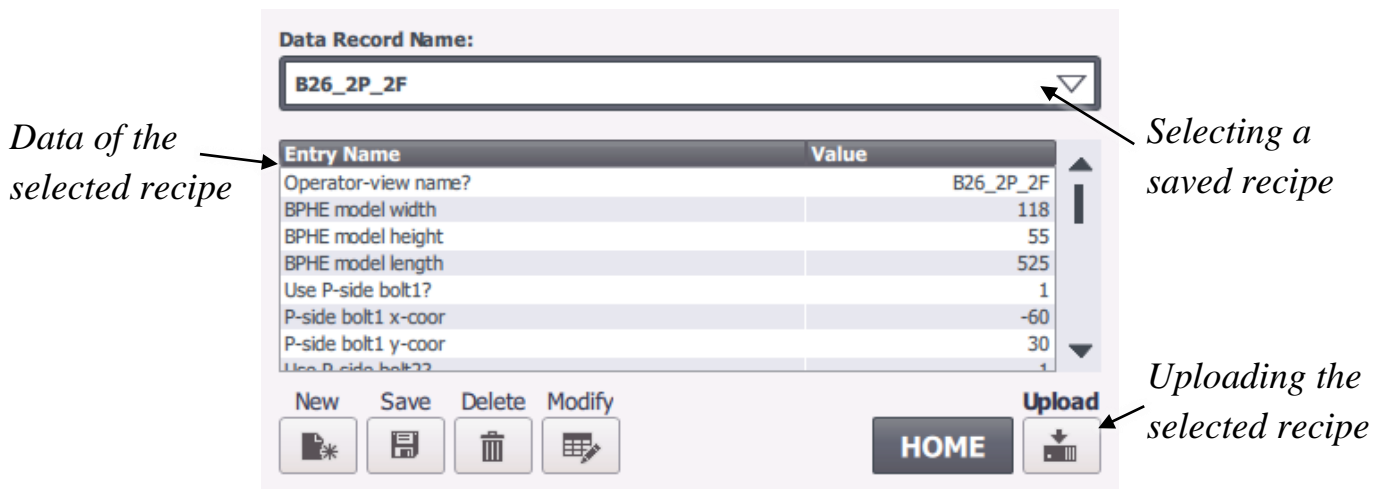


Figure 54: The final HMI - "Recipe view"

### 5.3.4 The "Settings" screen

In Figure 56, the "Settings" screen is presented. In this screen, the operator may change various settings about the process: tolerances of the welding parameters, enabling a "debug mode", enabling an abort of the work order if any parameter is measured outside its tolerance, and if the operator may edit saved recipes. The tolerances are set using a drop-down list with a distinct set of defined tolerance values.

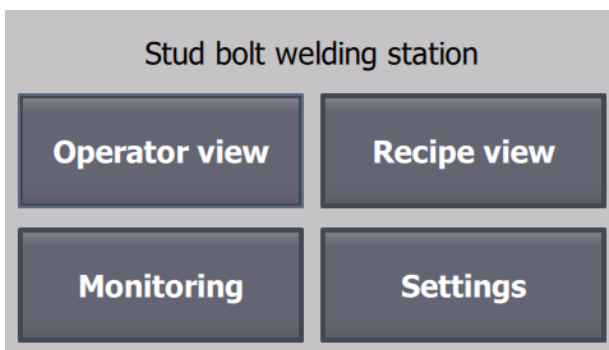


Figure 55: The final HMI - "Root screen"

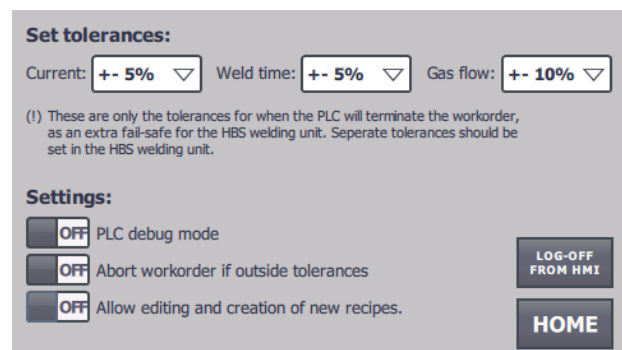


Figure 56: The final HMI - "Settings"

### 5.3.5 The "Monitoring" screen

In Figure 57, the "Monitoring" screen is presented. In this screen, live values of a majority of all PLC tags and other program data are presented. This offers the operator a single screen where all values may be monitored during the active process. It also becomes a valuable debugging tool.

### 5.3.6 The "Alarms" view

In Figure 58, the "Alarms" view is presented. In this screen, various alarms are shown to the operator. For example, if a recipe is incorrectly set up or if the chosen work order "number of units" has an incorrect value. Alarms are also shown after each weld if the measured current, weld time, or gas flow is outside the tolerances chosen in "Settings".

Program flow		Sensors		Active workorder	
program_state	10	current_now	0	numberOfUnits	5
irc_is_online	1	current_an_in	0	unitsDone	0
irc_in_start_pos	0	gaslevel_now	0		
cycleControl	1	gaslevel_an_in	0		
quickStop	0				
resetOrder	0				
want_flip	0				
Active recipe		Active unit		Welding	
recipe_name	B26_2...	finishedBolts_P	2	irc_wants_weld	0
model_width	118	finishedBolts_F	1	weld_power_hbs	0
model_height	55	currentlyOnPside	0	weld_done	0
model_length	525			weld_timer_on	0
bolts_P_side	2	Calculations		lastWeld_x	-60
bolts_F_side	2	current_avg	1521	lastWeld_y	-30
irc_received	1	gaslevel_avg	15	weld_coolof_ms	3000
		current_time_ms	730		
		weld_tmrs_elapsed	0		
		samplesDone	0		

**HOME**

Figure 57: The final HMI - "Monitoring"

No.	Time	Date	Text
A 8	1:54:59 PM	5/31/2022	Bad workorder. Check so that nu...
A 7	1:54:59 PM	5/31/2022	Bad recipe. Check that it is not empty.
A 4	1:54:59 PM	5/31/2022	Gas flow is too low, please look ...
A 1	1:54:59 PM	5/31/2022	Weld current is too high, please ...

**GO BACK**

Figure 58: The final HMI - "Alarms"

# 6 Conclusions

This chapter presents conclusions drawn from the results, discusses future work, and ends this thesis by giving an ethical reflection on the rise of automation.

## 6.1 The program algorithms and center calculation

### 6.1.1 The best method for calculating the center

After the improvements had been implemented and tested, and it was seen that the two methods of calculating the center now had equally good tolerance, both well within the target tolerance of 1 mm, the drawn conclusion was thus that the first method should be used in the welding process as it needs significantly less time for measurements. The three extra points used by the second method are not seen to improve the accuracy, as initially predicted. The eight points used by the first method are well sufficient to achieve the tolerances required at SWEP.

### 6.1.2 Future work

In between performing the two presented accuracy tests as seen in Table 5 and Table 6, more accuracy tests and tuning of the improvements were made until satisfactory results were seen. During these tests, an additional observation was made. It was noticed that bolts further away from the calculated center point were consistently less accurate. After some discussions with a measuring engineer at the lab, this was predicted to result from the calculated angle around the z-axis (as presented in Subchapter 1.3.9, and in more detail in Appendix A). Even a small error in this angle results in a growing inaccuracy as a bolt is positioned further away from the center. For the two tests presented in Subchapter 5.2, an error of only 0.1 degrees of this calculated angle would result in an additional error of 0.26 mm in the y-axis for bolts positioned at the edge of the heat exchanger, as envisioned by a right triangle with an angle of 0.1 degrees, the adjacent side as the distance from the heat exchanger center, and the opposite side length as the additional inaccuracy in the y-axis. This error would increase linearly to 0.52 mm for a heat exchanger with twice the length of the one used in the tests.

Therefore, to keep the accuracy within tolerances for larger models, future work should include an improvement in the calculation of this angle around the z-axis.

## 6.2 Fully automating the bolt welding process

This thesis work has further developed the already begun project at SWEP of creating a fully automated solution to use in the factory. However, there are several steps left until this can be completed. Even though most of the process is now automated from the PLC and HMI, there still needs to be an operator present to flip over some units, fill the bolt tray with bolts, as well as loading and unloading each new unit. To automate the process even further, these tasks could be automated as well.

As the automated solution created in the HMI and PLC program were developed from the bottom up, additional investigating work should be performed to further enhance and secure

the solution, as there are several needed considerations that were outside the scope of the thesis work. For example, if an incorrect model of a heat exchanger is placed on the workstation, whose dimensions differ from the chosen recipe, the touch probe could get damaged, or the measuring algorithm could fail to find the surface or sides. A possible solution to this would be to not include the model dimensions in the recipe, but instead use separate distance sensors mounted around the workstation to automatically measure and confirm that the correct model is being used.

### **6.3 An ethical reflection of automation – is it beneficial to society?**

During the last century, the rise of automation has had a significant impact on the manufacturing industry and has overall given technological progress to society. During this rise, it is therefore important to reflect on how it affects society and people. The main concern is often whether or not it will result in mass replacement of factory jobs, where robots are put to replace many workers. As with many major changes in society, there are often both positive and less positive consequences. In the case of automation, I would argue that it coincides with other industrial revolutions of mechanization, electrification and digitalization as yet another revolution that brings society and quality of life forward.

Automation brings an increase in productivity, quality, and repeatability, as automated robots often operate faster, with more accuracy, and with fewer errors than human workers. This is the main cause of why automation often replaces existing factory jobs and workers. However, I think it is important to also realize that it creates a safer industry. Automated robots can replace many repetitive or dangerous jobs which otherwise could result in great harm to the worker. Automation also results in creating entirely new jobs in robot programming, development, and administration. It often also creates a more efficient use of material and less waste, being a great contribution to stop any future energy crisis resulting from global warming.

It has been clear that many jobs have been lost to automation. However, just as after many other major historical changes to society, I think it will bring society forward by creating more opportunities and reasons to focus on innovation and research.

## 7 Terminology

<b>“Dead man's switch”</b>	A switch designed to be deactivated if the human operator becomes incapacitated or leaves the operator area.
<b>GSDML</b>	A readable text file containing device-specific identification and communication specifications.
<b>HMI</b>	Human-Machine-Interface, a user interface that connects an operator to a device or system.
<b>IE</b>	Industrial Ethernet, a common standard for using Ethernet in an industrial environment.
<b>IRC</b>	An Industrial Robot Controller made by ABB.
<b>PLC</b>	Programmable Logic Controller, an industrial computer designed for process control in harsh factory environments.
<b>PROFINET</b>	An industry standard for data communication over Industrial Ethernet.
<b>SOP</b>	Standard Operating Procedure, a defined set of instructions made by a company or organization, used by its workers.
<b>“State machine”</b>	A Finite-state machine, an abstract computational model that can be in exactly one of several states at any time, transitioning between states in response to inputs.
<b>TCP</b>	The Tool Center Point coordinate system for industrial robots in ABB RobotStudio, the center point of the active tool.
<b>TP</b>	True Position, of a measurement, is twice the distance between the theoretical point and the actual point.
<b>XML</b>	Extensible Markup Language, a file format for storing arbitrary data in a defined set of rules and encoding.



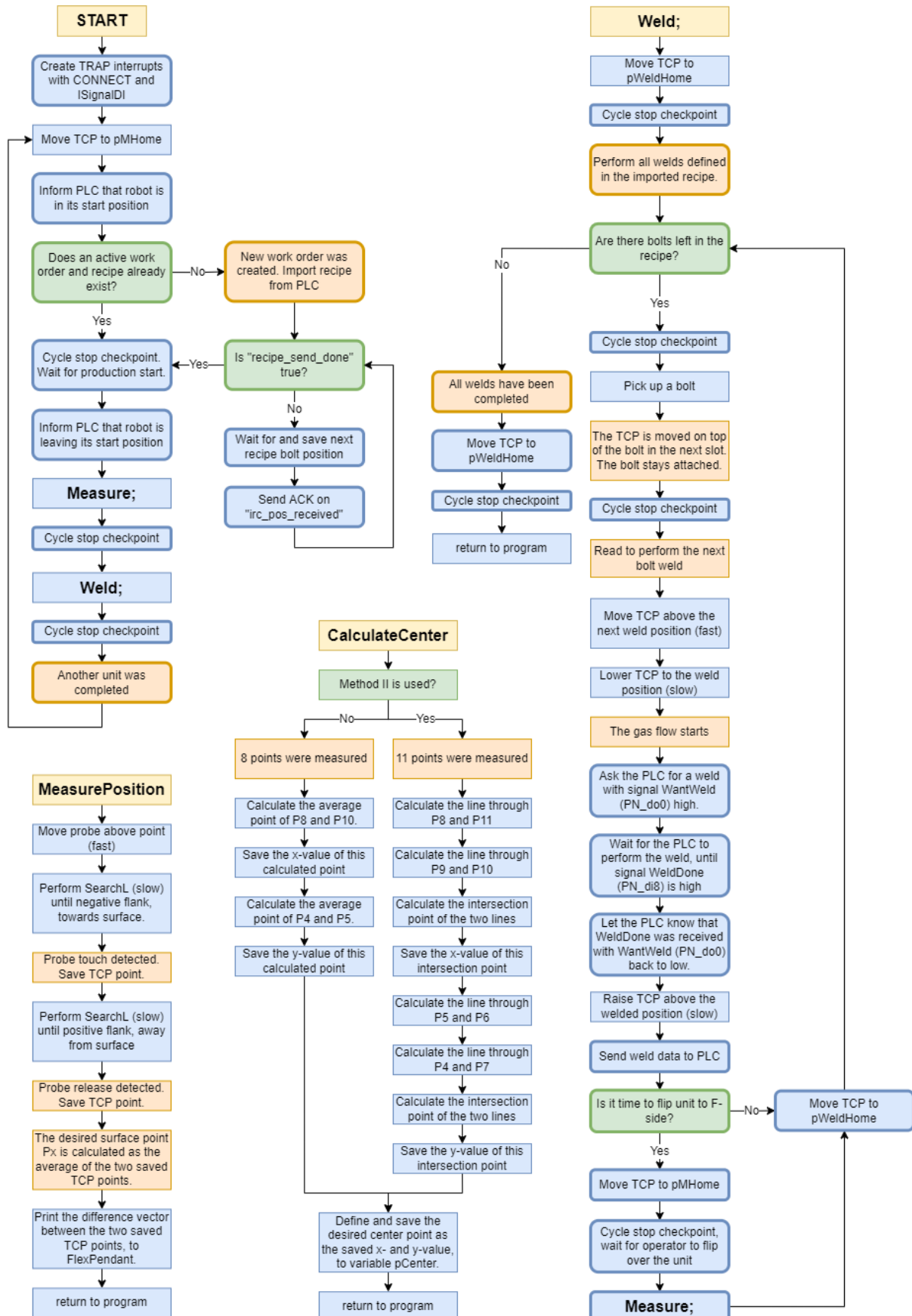
## 8 References

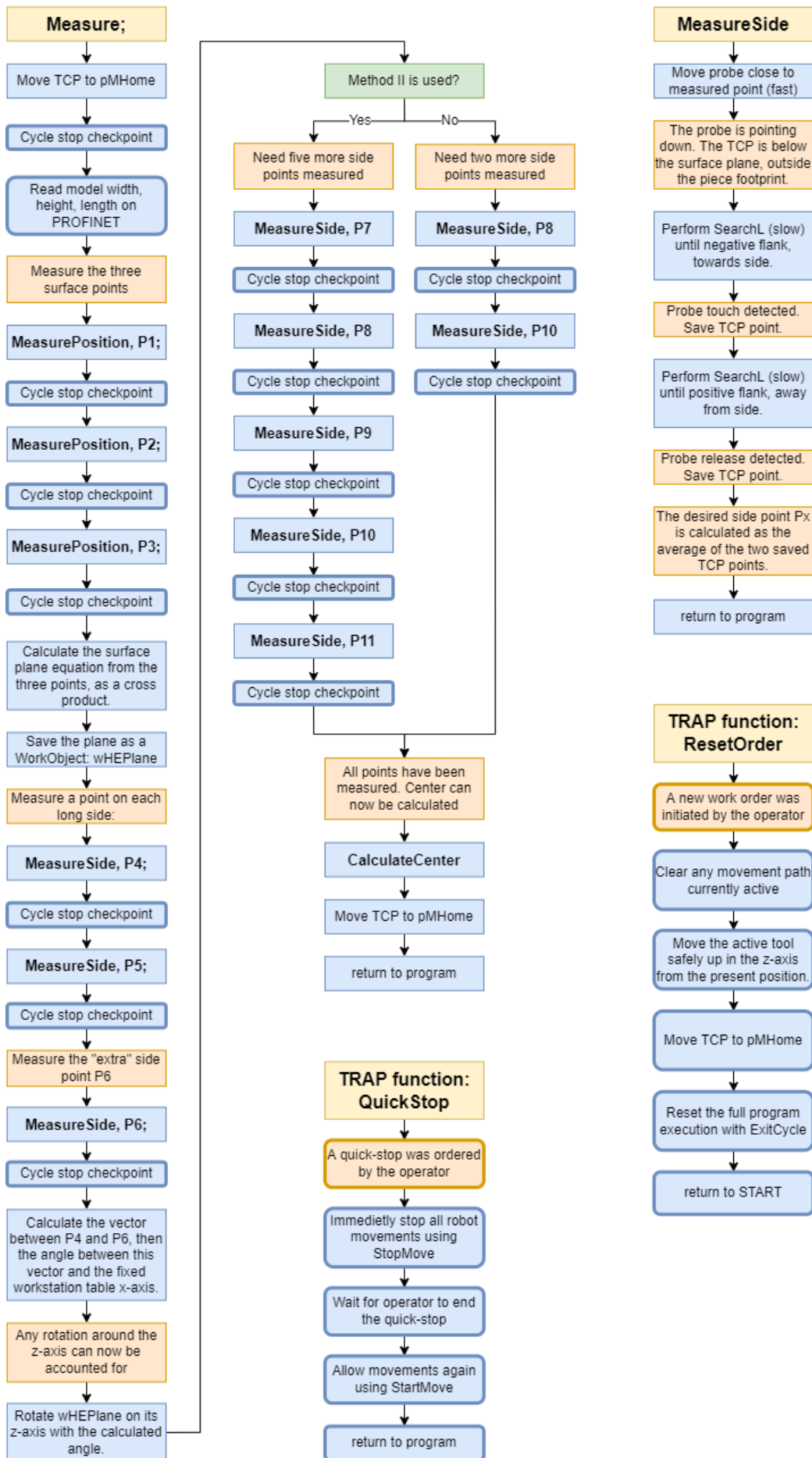
- [1] SWEP International AB. *About SWEP*. URL: <https://www.swep.net/company/company/> [visited on 2022-03-24]
- [2] SWEP International AB. *BPHE technology*. URL: <https://www.swep.net/technology/> [visited on 2022-03-24]
- [3] ABB Robotics. *IRB 4600*. URL: <https://new.abb.com/products/robotics/industrial-robots/irb-4600> [visited on 2022-03-31]
- [4] Tormach. *Passive probe technical document*. URL: [https://tormach.com/media/asset/t/d/td10088\\_passive\\_probe\\_install\\_0515a.pdf](https://tormach.com/media/asset/t/d/td10088_passive_probe_install_0515a.pdf) [visited on 2022-03-31]
- [5] SWEP International AB. The initial RAPID program. *Measures and welds mounting points to a heat exchanger*. Code version 1.9.
- [6] HBS. *Visar 1200*. URL: <https://hbs-info.com/products/welding-units/visar-1200> [visited on 2022-04-04]
- [7] ABB Robotics. *IRC5 Industrial Robot Controllers*. URL: <https://new.abb.com/products/robotics/controllers/irc5-overview> [visited on 22-05-23]
- [8] Siemens. *SIMATIC S7-1200*. URL: <https://new.siemens.com/global/en/products/automation/systems/industrial/plc/s7-1200.html> [visited on 2022-04-23]
- [9] Siemens. *SIMATIC HMI KTP400 Basic*. URL: <https://support.industry.siemens.com/cs/products/6av2123-2db03-0ax0/simatic-hmi-ktp400-basic?pid=379924&mlfb=6AV2123-2DB03-0AX0&mfn=ps&lc=en-WW> [visited on 22-05-27]
- [10] ABB Robotics. *RobotStudio*. URL: <https://new.abb.com/products/robotics/robotstudio> [visited on 22-03-31]
- [11] Siemens. *Totally Integrated Automation Portal*. URL: <https://new.siemens.com/global/en/products/automation/industry-software/automation-software/tia-portal.html> [visited on 2022-05-09]
- [12] PROFIBUS. *GSDML Specification for PROFINET*. URL: <https://www.profibus.com/download/gsdml-specification-for-profinet> [visited on 22-05-23]
- [13] Siemens. *SIMATIC S7-PLCSIM*. URL: <https://mall.industry.siemens.com/mall/sv/se/Catalog/Products/5000483> [visited on 2022-05-09]
- [14] Siemens. *SIMATIC WinCC*. URL: <https://new.siemens.com/global/en/products/automation/industry-software/automation-software/tia-portal/software/simatic-wincc-tia-portal.html> [visited on 2022-05-09]
- [15] International Electrotechnical Commission. *IEC 61131-3*. URL: <https://webstore.iec.ch/publication/4552> [visited on 2022-05-27]
- [16] SWEP International AB. *Quality control of stud bolts*. Doc-Id: SOP-2405-7.

Reading instructions: begin at START. Whenever a box appears with a **bold name**, it alerts you to jump to the separate function with that name.

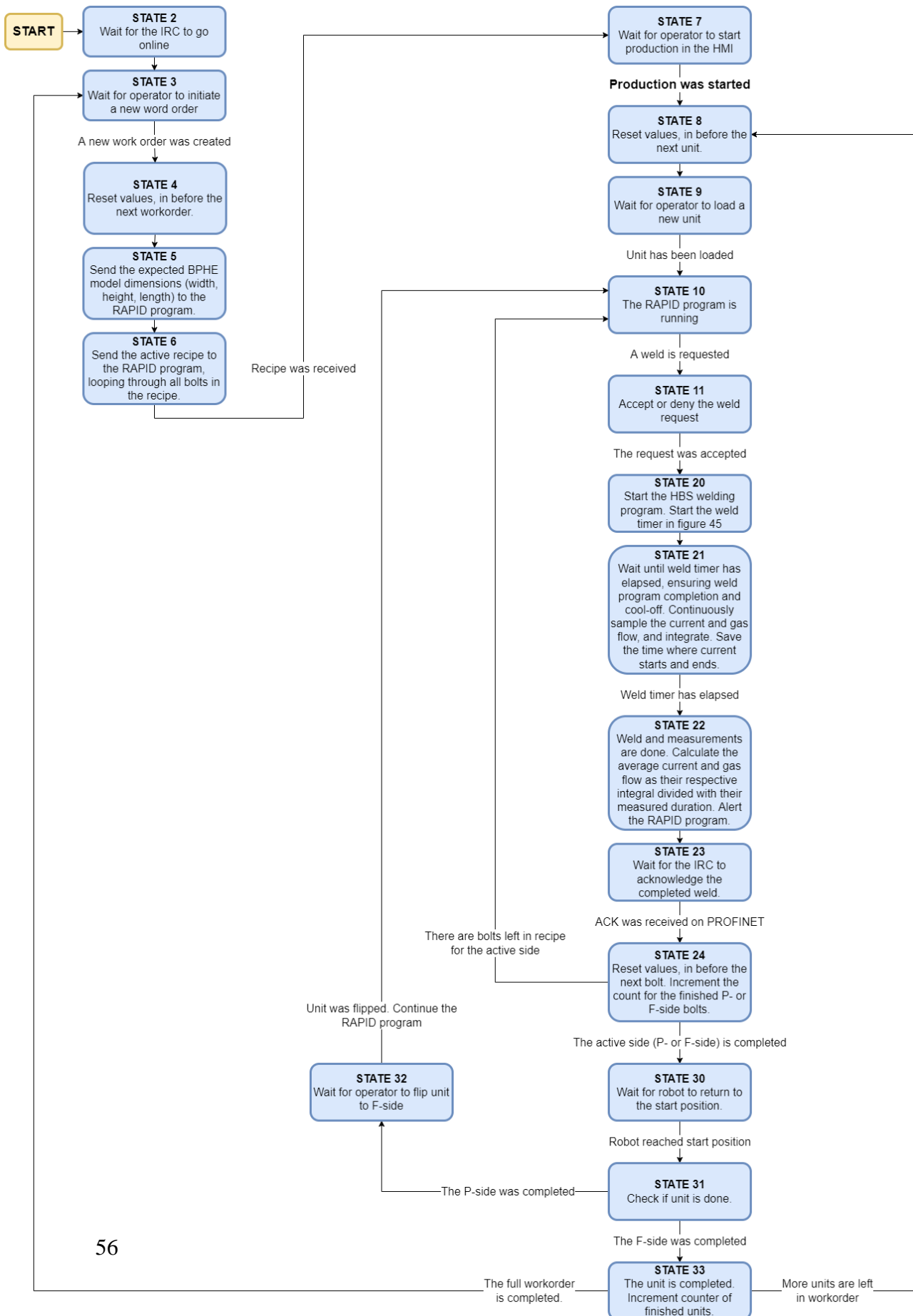
Rectangular parts: the original RAPID program.  
Rounded off parts with thicker borders: the additions made in this thesis.

## Appendix A: Flow chart of the RAPID program





## Appendix B: State diagram of the PLC program



## Appendix C: PLC program code

```
1 //This is the main PLC program. It will run continuously.
2
3 IF "btnPress_cycle" OR "btnPress_quick" OR "btnPress_reset" THEN
4     "InputManager"(); //a button was pressed. Check if it may perform its action
5 END_IF;
6
7 "Timers_DB"(); //update timers
8
9 //Measure current
10 IF "ReadAnalogInputs"(analog_in := "current_analog_in",
11     scale_lower := 0, //current value when sensor signal MIN
12     scale_upper := 1500, //current value when sensor signal MAX
13     out_value => "Data".current_now);
14
15 //Measure gas flow
16 IF "ReadAnalogInputs"(analog_in := "gas_analog_in",
17     scale_lower := 0,
18     scale_upper := 30,
19     out_value => "Data".gaslevel_now);
20
21 //Read the analog inputs from PN. Must SWAP bytes in each WORD.
22 "Data"."lastWeld_x" := WORD_TO_UINT(SWAP_WORD("lastWeld_x_analog")) - "Calc".word_signOffset;
23 "Data"."lastWeld_y" := WORD_TO_UINT(SWAP_WORD("lastWeld_y_analog")) - "Calc".word_signOffset;
24
25 "state_to_irc" := SWAP_WORD(INT_TO_WORD("Data".program_state)); //Send program state to IRC
26
27 //Control the custom HMI text of the "Production button", as it is dep on BOTH hmi_cycleControl AND program_state
28 IF "hmi_cycleControl" THEN
29     "HmiData".cycleBtnTextControl := 10; //use text "Cycle stop"
30 ELSE
31     IF "Data".program_state <= 8 OR "Data".program_state = 90 THEN
32         "HmiData".cycleBtnTextControl := 0; //use text "Start production"
33     ELSIF "Data".program_state = 32 OR "Data".program_state = 9 THEN
34         "HmiData".cycleBtnTextControl := 1; //use text "Resume production", is waiting for flip
35     ELSE
36         "HmiData".cycleBtnTextControl := 11; //use text "Leave cycle stop"
37     END_IF;
38 END_IF;
39
40 //the finite-state-machine
41 CASE "Data".program_state OF
42
43     2: //Wait for IRC to start
44
45     IF "irc_is_online" THEN
46         "hmi_resetOrder" := FALSE;
47         "HmiData".warningActive := FALSE; //remove when has dropped warning thing cleanly
48         "Data".program_state := 3;
49     END_IF;
50
51     3: //Wait for "Reset Order"
52     ;
53     4: //Reset before each WORKORDER
54
55     "hmi_recipe_sent" := FALSE;
56     "hmi_cycleControl" := FALSE;
57     "hmi_quickStop" := FALSE;
58     "ActiveWorkOrder".unitsDone := 0;
59
60     "Calc".weld_wait_ms := T#3000ms; //Weld execute+cooloff time
61     "Calc".analog_StartCheck := 20000; //After what analog value (0 to 27648) the current will count as live
62     "Calc".word_signOffset := 10000;
63     "HmiData".cycleBtnTextControl := 0;
64     "ActiveRecipe".currentlyOnPside := TRUE;
```

```

65
66 //Calculate the amount of P- and F-side bolts in the active recipe:
67 "ActiveRecipe".activeBolts_Pside := BOOL_TO_INT("ActiveRecipe".P_bolt1_active) + BOOL_TO_INT("ActiveRecipe".P_bol
68 "ActiveRecipe".activeBolts_Fside := BOOL_TO_INT("ActiveRecipe".F_bolt1_active) + BOOL_TO_INT("ActiveRecipe".F_bol
69
70 IF "robot_inStartPos" THEN
71     "Data".program_state := 5;
72 END_IF;
73
74 5: //Update the model dimensions to the IRC (is needed as "expected dimensions" when measuring later)
75
76     "hmi_model_width" := SWAP_WORD(INT_TO_WORD("ActiveRecipe".modelDim_width));
77     "hmi_model_height" := SWAP_WORD(INT_TO_WORD("ActiveRecipe".modelDim_height));
78     "hmi_model_length" := SWAP_WORD(INT_TO_WORD("ActiveRecipe".modelDim_length));
79
80 //Reset before recipe send
81 "hmi_recipe_sent" := FALSE;
82 "hmi_pos_sent" := FALSE;
83 "ActiveRecipe".sendCount := 0;
84
85 "Data".program_state := 6;
86
87 6: //Send the full recipe to the IRC
88
89 IF "ActiveRecipe".sendCount < 8 THEN
90     "ActiveRecipe".sendCount := "ActiveRecipe".sendCount + 1;
91     "SendBoltToIRC"(boltNbr := "ActiveRecipe".sendCount,
92         retVal => "boltActive");
93     IF "boltActive" THEN
94         "hmi_pos_sent" := TRUE;
95         "Data".program_state := 90; //go to recipe side-handler
96     END_IF;
97 ELSE
98     "hmi_recipe_sent" := TRUE;
99     "hmi_resetOrder" := FALSE;
100    "Data".program_state := 7;
101 END_IF;
102
103 7: //Wait for PRODUCTION START...
104
105 IF "hmi_cycleControl" THEN
106     "Data".program_state := 8;
107 END_IF;
108
109 8: //Reset before each new UNIT
110
111 IF "ActiveWorkOrder".unitsDone = 0 THEN
112     "hmi_cycleControl" := 0; //wait for the first unit
113 END_IF;
114
115 //Decide if should start on P- or F-side
116 IF "ActiveRecipe".activeBolts_Pside > 0 THEN
117     //We have at least one bolt on P-side, so start on P-side
118     "ActiveRecipe".currentlyOnPside := TRUE;
119 ELSE
120     //We have no bolts on P-side, so this recipe is F-side only, so start there.
121     "ActiveRecipe".currentlyOnPside := FALSE;
122 END_IF;
123
124 "ActiveRecipe".finishedBolts_Pside := 0;
125 "ActiveRecipe".finishedBolts_Fside := 0;
126 "hmi_want_flip" := FALSE;
127 "hmi_cycleControl" := FALSE;
128

```

```

129     "Data".program_state := 9;
130
131 9: //Wait for new BPHE to be loaded/placed
132
133 IF "hmi_cycleControl" THEN
134     "Data".program_state := 10;
135 END_IF;
136
137 10: //Weld program running. PLC waiting for weld request from IRC.
138
139 IF "irc_want_weld" THEN
140     //irc is in welding position, gas is flowing. IRC wants a weld.
141     "Data".program_state := 11;
142 END_IF;
143
144 11: //Weld request received
145
146     "Data".program_state := 20; //accept the weld request
147     "current_timer_reset" := FALSE;
148     "Data".current_avg := 0;
149     "Data".gas_avg := 0;
150     "Data".time_measured := T#0ms;
151
152 20: //Start the weld
153
154     "HmiData".AlarmTriggerBits := 0;
155     "weld_power_hbs" := TRUE; //activate the welding program on the HBS
156     "weld_timer_on" := TRUE; //start the weld timer
157     "Calc".samplesDone := 0;
158     "Data".program_state := 21;
159
160 21: //Wait until weld is done (weld+coolof), and actively MEASURE all weld params
161
162 IF "current_analog_in" > "Calc".analog_StartCheck AND "Calc".current_isactive = FALSE THEN
163     "current_timer_on" := TRUE; //Current has begun, so start timer
164     "Calc".current_isactive := TRUE;
165 END_IF;
166
167 IF "current_analog_in" < "Calc".analog_StartCheck AND "Calc".current_isactive = TRUE THEN
168     "current_timer_on" := FALSE; //Current has stopped, end timer
169     "Calc".current_isactive := FALSE;
170 END_IF;
171
172 //Wait until weld timer is done
173 IF "weld_timer_done" THEN
174     "weld_timer_on" := FALSE; //reset timer
175     "weld_power_hbs" := FALSE; //turn off the welding program
176     "Data".program_state := 22;
177 END_IF;
178
179 22: //Weld is done. Parameters has been measured.
180
181 //Calculate the average current and gas, from the integrals
182 "Calc".time_measured_int := TIME_TO_INT("Data".time_measured); //dont merge with below. HMI needs int.
183
184 IF "Calc".time_measured_int > 0 THEN
185     "Data".current_avg := "Calc".current_integral / "Calc".time_measured_int;
186 ELSE
187     "Data".current_avg := 0;
188 END_IF;
189
190 "Data".gas_avg := "Calc".gas_integral / "Calc".samplesDone;
191 "weld_done" := TRUE; //alert the IRC
192 "Data".program_state := 23;

```

```

193
194 23: //Wait for IRC acknowledgement
195
196 □ IF "irc_want_weld" = FALSE THEN
197     "weld_done" := FALSE;
198     "Data".program_state := 24;
199     END_IF;
200
201 24: //Weld is done! Reset between welds.
202
203     "current_timer_on" := FALSE;
204     "current_timer_reset" := TRUE;
205     "weld_timer_on" := FALSE;
206     "weld_power_hbs" := FALSE;
207     "weld_done" := FALSE;
208
209     "Calc".weld_wait_ms := T#3000ms; //Weld execute+cooloff time
210     "Calc".current_isactive := FALSE;
211     "Calc".samplesDone := 0;
212     "Calc".current_integral := 0;
213     "Calc".gas_integral := 0;
214     "Data".current_now := 0;
215     "Data".gaslevel_now := 0;
216
217     //Put out alarm if any weld parameter was outside the tolerances
218 □ IF "HmiData".abortIfOutsideTol THEN
219 □     IF "Data".current_avg > "Tol".currentTarget * (1 + "Tol".currentTol / 100) THEN
220         "HmiData".AlarmTriggerBits := "HmiData".AlarmTriggerBits OR LREAL_TO_UINT(2 ** 0); //Current TOO HIGH
221     ELSIF "Data".current_avg < "Tol".currentTarget * (1 - "Tol".currentTol / 100) THEN
222         "HmiData".AlarmTriggerBits := "HmiData".AlarmTriggerBits OR LREAL_TO_UINT(2 ** 1); //Current TOO LOW
223     END_IF;
224
225 □     IF "Data".gas_avg > "Tol".gasTarget * (1 + "Tol".gasTol / 100) THEN
226         "HmiData".AlarmTriggerBits := "HmiData".AlarmTriggerBits OR LREAL_TO_UINT(2 ** 2); //Gas TOO HIGH
227     ELSIF "Data".gas_avg < "Tol".gasTarget * (1 - "Tol".gasTol / 100) THEN
228         "HmiData".AlarmTriggerBits := "HmiData".AlarmTriggerBits OR LREAL_TO_UINT(2 ** 3); //Gas TOO LOW
229     END_IF;
230
231 □     IF "Calc".time_measured_int > "Tol".timeTarget * (1 + "Tol".timeTol / 100) THEN
232         "HmiData".AlarmTriggerBits := "HmiData".AlarmTriggerBits OR LREAL_TO_UINT(2 ** 4); //Time TOO LONG
233     ELSIF "Calc".time_measured_int < "Tol".timeTarget * (1 - "Tol".timeTol / 100) THEN
234         "HmiData".AlarmTriggerBits := "HmiData".AlarmTriggerBits OR LREAL_TO_UINT(2 ** 5); //Time TOO SHORT
235     END_IF;
236     END_IF;
237
238     //Increment finished bolt count
239 □ IF "ActiveRecipe".currentlyOnPside THEN
240     "ActiveRecipe".finishedBolts_Pside := "ActiveRecipe".finishedBolts_Pside + 1;
241 ELSE
242     "ActiveRecipe".finishedBolts_Fside := "ActiveRecipe".finishedBolts_Fside + 1;
243 END_IF;
244
245     //Check if P side is done
246 □ IF "ActiveRecipe".currentlyOnPside AND "ActiveRecipe".finishedBolts_Pside = "ActiveRecipe".activeBolts_Pside THEN
247     //This was the last bolt on the P side
248 □     IF "ActiveRecipe".currentlyOnPside AND "ActiveRecipe".activeBolts_Fside > 0 THEN
249         //there are bolts on F-side too, so need FLIP
250         "hmi_want_flip" := TRUE;
251     END_IF;
252     "Data".program_state := 30;
253 ELSIF "ActiveRecipe".currentlyOnPside = FALSE AND "ActiveRecipe".finishedBolts_Fside = "ActiveRecipe".activeBolts_
254     //This was the last bolt on the F side
255     "Data".program_state := 30;
256     "hmi_want_flip" := FALSE;

```



```

257     ELSE
258         //There are more bolts on the active side
259         "Data".program_state := 10;
260     END_IF;
261
262 30: //Wait until robot has returned to the start position
263
264     IF "robot_inStartPos" THEN
265         "hmi_want_flip" := FALSE;
266         "Data".program_state := 31;
267     END_IF;
268
269 31: //The current side (P or F) was completed
270
271     IF "ActiveRecipe".currentlyOnPside THEN
272         //the P side was done, wait for FLIP, then perform F side, if F bolts exist
273     IF "ActiveRecipe".activeBolts_Fside > 0 THEN
274         "hmi_cycleControl" := FALSE; //USING THE CYCLE-STOP FOR THE FLIP-WAIT TOO
275         "ActiveRecipe".currentlyOnPside := FALSE;
276         "Data".program_state := 32;
277     ELSE
278         //P side was done, and there are NO F-side bolts. Unit is completed.
279         "Data".program_state := 33;
280     END_IF;
281     ELSE
282         //The F side was done, so ALL IS DONE!
283         "Data".program_state := 33;
284     END_IF;
285
286 32: //P-side was completed, so wait for OP to flip the HE
287
288     IF "hmi_cycleControl" = TRUE THEN
289         "Data".program_state := 10; //"Resume" was pressed
290     END_IF;
291
292 33: //UNIT IS FULLY COMPLETED. Check active workorder, if more units are needed.
293
294     "ActiveWorkOrder".unitsDone := "ActiveWorkOrder".unitsDone + 1;
295     IF "ActiveWorkOrder".unitsDone = "ActiveWorkOrder".numberOfUnits THEN
296         //The last unit was completed. The full work order is completed!
297         "hmi_cycleControl" := FALSE;
298         "hmi_resetOrder" := FALSE;
299         "Data".program_state := 3;
300     ELSE
301         //There are units left in the workorder
302         "Data".program_state := 8;
303     END_IF;
304
305 90:
306     //RECIPE SEND: wait for IRC ACK
307     IF "irc_pos_received" THEN
308         "hmi_pos_sent" := FALSE;
309         "Data".program_state := 91;
310     END_IF;
311
312 91:
313     //RECIPE SEND: wait for IRC reset before next bolt
314     IF "irc_pos_received" = FALSE THEN
315         "Data".program_state := 6;
316     END_IF;
317     ELSE
318         ;
319 END_CASE;
320

```

## Separate function: **SampleWeldParameters**

```
1 //This will be called every lms during the welding unit program
2
3 IF "Data".program_state = 21 THEN
4     //Is in state 21, i.e weld is ongoing. Integrate the current and gas parameters!
5     "Calc".current_integral := "Calc".current_integral + 1 * "Data".current_now;
6     "Calc".gas_integral := "Calc".gas_integral + 1 * "Data".gaslevel_now;
7     "Calc".samplesDone := "Calc".samplesDone + 1;
8 END_IF;
```

## Separate function: **InputManager**

```
1 //To allow for a complex yet dynamic input system from the HMI (multiple actions used for a single button, etc) the hmi
2 //buttons only activate dummy-signals btnPress_x, that in turn cause this function to run.
3 //Here, the hmi input is checked towards the active program state, to allow or ignore the input. For example,
4 //so that "Start production" may not be triggered if the chosen recipe is empty, or workorder unit count is zero.
5
6 "state" := "Data".program_state;
7
8 IF "btnPress_cycle" THEN
9     //check if it may do its action in the current state
10    IF (("state" = 7 AND "ActiveRecipe".activeBolts_Pside + "ActiveRecipe".activeBolts_Fside > 0 AND "ActiveWorkOrder".nu
11        OR ("state" >= 9 AND "state" <= 30) OR "state" = 32) AND "hmi_quickStop" = FALSE THEN
12        //allow
13        "hmi_cycleControl" := NOT ("hmi_cycleControl");
14    END_IF;
15
16    IF "state" = 7 AND "ActiveRecipe".activeBolts_Pside + "ActiveRecipe".activeBolts_Fside <= 0 THEN
17        "HmiData".AlarmTriggerBits := "HmiData".AlarmTriggerBits OR LREAL_TO_UINT(2 ** 6); //faulty or empty recipe input
18    END_IF;
19
20    IF "state" = 7 AND "ActiveWorkOrder".numberOfUnits <= 0 THEN
21        "HmiData".AlarmTriggerBits := "HmiData".AlarmTriggerBits OR LREAL_TO_UINT(2 ** 7); //faulty or empty workorder in
22    END_IF;
23
24    ELSIF "btnPress_quick" THEN
25        //check if it may do its action in the current state
26        IF (("state" = 10 OR "state" = 11 OR "state" = 30) AND "hmi_cycleControl") OR "hmi_quickStop" THEN
27            //allow
28            "hmi_quickStop" := NOT ("hmi_quickStop");
29        END_IF;
30
31    ELSIF "btnPress_reset" THEN
32        //check if it may do its action in the current state
33        IF "irc_is_online" AND (("state" = 3 AND "robot_inStartPos") OR "state" = 7 OR "state" = 9 OR "state" = 10 OR "state"
34            OR "state" = 32 OR "state" = 90 OR "state" = 91) THEN
35            //allow
36            "hmi_resetOrder" := TRUE;
37            "Data".program_state := 4;
38        END_IF;
39
40    ELSE
41        ;
42    END_IF;
43
44 //Reset dummy tags (dont allow simultan presses, so reset them all)
45 "btnPress_cycle" := FALSE;
46 "btnPress_quick" := FALSE;
47 "btnPress_reset" := FALSE;
```