CODEN:LUTEDX/(TEIE-5532)/1-103/(2025)

Industrial Electrical Engineering and Automation

Automation and Simulation of a Robot Cell for Stud Bolt Welding

Elias Ljunglöv

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

Automation and simulation of a robot cell for stud bolt welding

Elias Ljunglöv

A thesis presented for the degree of
Master of Science in Electrical Engineering

LUNDS UNIVERSITET

LTH, Faculty of Engineering, Lund University, March 2025

Supervisor: Jörgen Svensson
Examiner: Ulf Jeppsson

Abstract

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 various systems and industries. Stud bolts are manually welded onto the heat exchanger surface in their manufacturing process. A project to fully automate this process using an industrial robot has already begun at SWEP to increase the process productivity, quality, and repeatability. The idea is to use a robot with an attached touch probe to measure points on the surface and sides of a heat exchanger in high precision to calculate a center reference point and account for any slight rotational offset. Then, the same robot uses an attached weld gun to weld stud bolts onto the surface. This thesis aimed to continue their project of creating a fully automated solution to eventually replace the manual bolt welding station with an automated robot cell solution.

The first objective was to study their pre-existing robot program used for initial testing 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 for two different methods for calculating the center reference point was then measured. Improvements and calibration 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 pre-existing robot program via PROFINET (industry standard for data communication over Industrial Ethernet) on a programmable logic controller (PLC) and Human-Machine-Interface (HMI) both programmed in Siemens TIA Portal. Then, an attempt was made to improve the pre-existing program algorithm in ABB RobotStudio and the way a heat exchanger is measured and mounted in the robot cell, mounting it horizontally instead of vertically, allowing the robot to reach both sides of the heat exchanger at the same time whereas the pre-existing program required a manual flip of the heat exchanger during operation. This horizontal approach also introduced an asymmetric gravitational effect on the weld quality that was then analyzed. This resulted in a more automated solution for the bolt welding process where the HMI could be used to create a work order, create different bolt configuration recipes, monitor the process, and control the pre-existing as well as the new and improved robot program.

A planned future state for the robot cell was then programmed and simulated in Tecnomatix Plant Simulation and integrated into SWEP's factory-wide simulation model to analyze the upcoming robot cell cycle time and output potential. A light 3D model of the cell and its used equipment was then created in Autodesk Inventor to make the simulation as graphically detailed as the rest of SWEP's factory-wide model.

Finally, conclusions were drawn on the best method for calculating the center point of a SWEP heat exchanger, the effect of gravity on weld quality in horizontal welding, the value of virtual commissioning and cycle time analysis in material flow simulation, and general conclusions concerning the project of fully automating the stud bolt welding process at SWEP. The report ends with a reflection on the rise of automation in our society.

Keywords: stud bolt welding, automated welding, brazed plate heat exchangers, PLC, HMI, IRC, ABB RobotStudio, Siemens TIA Portal, Tecnomatix Plant Simulation.

Sammanfattning

Vikten av hållbar och effektiv energianvändning växer snabbt. SWEP International AB leder tillverkningen av lödda plattvärmeväxlare, en produkt som erbjuder effektiv värmeöverföring i ett brett område olika värmesystem och industrier. I sin tillverkningsprocess svetsas bultar manuellt på värmeväxlarens yta. Ett projekt för att helt automatisera denna process med hjälp av en industrirobot har redan påbörjats av SWEP för att öka processens produktivitet och svetskvalitet. Tanken är att använda en robot med en ansluten kontaktprob för att med hög precision först hitta punkter på värmeväxlarens ytor för att kunna beräkna dess mittpunkt och kunna ta hänsyn till eventuell rotation, för att sedan med en svetspistol svetsa fast bultar på ytan med hög precision. Syftet med detta examensarbete vad att fortsätta detta projekt av att skapa en helt automatiserad lösning med mål att ersätta de manuella bultsvetsstationerna på SWEP.

Det första målet var att studera deras redan existerande robotprogram som användes för initial testning, samt utföra kvalitetsanalys för automatiserad svetsning genom att variera storleken på svetsström, svetstid samt gasflöde. Ett böjtest, vridmomenttest och visuell inspektion utfördes sedan för att förstå hur varje svetsparameter påverkar svetsresultatet vid automatiserad svetsning. Utifrån detta bestämdes en lämplig uppsättning parametervärden för automatiserad svetsning. Därefter mättes positionsnoggrannheten för två olika metoder att beräkna referenspunkten i mitten. En förbättring och kalibrering av de två metoderna utfördes för att öka dess noggrannhet. Slutligen drogs en slutsats kring vilken beräkningsmetod som bör användas. Svetsenheten som levererar svetsström och gasflöde installerades sedan för att mäta och övervaka svetsparametrarna och skapa ett felsäkert system som stoppar processen vid felaktig svetsning.

En kontrollstruktur implementerades sedan för att driva det redan existerande robotprogrammet via PROFINET (branschstandard för datakommunikation över industriellt Ethernet) på en programmerbar logisk styrenhet (PLC) och tillhörande Human Machine Interface (HMI) som båda programmerades i Siemens TIA Portal. Sedan gjordes ett försök till att skapa en alternativ och förbättrad programalgoritm för hur en värmeväxlare mäts och monteras i robotcellen än algoritmen i det redan existerande robotprogrammet i ABB RobotStudio. Denna förbättring innebar att montera varje värmeväxlare horisontellt istället för vertikalt, vilket möjliggjorde roboten att nå båda sidorna på värmeväxlaren samtidigt, där det existerande programmet krävde en manuell vändning av värmeväxlaren under programmets gång. Detta horisontella tillvägagångssätt introducerade samtidigt en asymmetrisk gravitationseffekt på svetskvaliteten som sedan analyserades. Detta resulterade i en mer automatiserad lösning för bultsvetsprocessen där ett HMI kunde användas för att skapa en arbetsorder, recept innehållande bultkonfigurationer, övervaka processen i realtid samt styra både det redan existerande såväl som det nya och förbättrade robotprogrammet.

En framtida design för robotcellen programmerades och simulerades sedan i Tecnomatix Plant Simulation och integrerades sedan i SWEPs egna fabriksomfattande simuleringsmodell för att sedan analysera den kommande robotcellens cykeltid och produktionshastighet. En lätt 3D-modell av cellen och dess utrustning skapades sedan i Autodesk Inventor för att göra simuleringen lika grafiskt detaljerad som resten av SWEPs egna fabriksmodell. Slutligen drogs slutsatser om vilken som är den bästa metoden för att beräkna mittpunkten för en värmeväxlare, gravitationseffekten på svetskvaliteten vid horisontell svetsning, värdet av en virtuell driftsättning och cykeltidsanalys i en stor simulering av materialflöde, samt allmänna slutsatser som dragits angående projektet att automatisera bultsvetsningen på SWEP. Rapporten avslutas med en reflektion över automatiseringens framväxt i vårt samhälle.

Nyckelord: bultsvetsning, automatiserad svetsning, lödda plattvärmeväxlare, PLC, HMI, IRC, ABB RobotStudio, Siemens TIA Portal, Tecnomatix Plant Simulation.

Acknowledgments

I want to take the opportunity to thank my supervisors, Anders Robertsson at the Department of Automatic Control, and Jörgen Svensson at the Division of Industrial Electrical Engineering and Automation. A special thanks then go to Qian Zhao, Automation Specialist at SWEP, Oskar Persson, Lead Automation Specialist at SWEP, and Lennie Liegnell, Global Engineering Manager at SWEP, for their combined assistance, support, and motivation that they have given me throughout this thesis work. Lastly, I want to thank the Programme Director at the Department of Engineering for letting me continue building upon my bachelor's thesis to serve as my final master's thesis. In this thesis, I have learned a tremendous amount about working as an engineer, giving me a great technical foundation for my future career as an automation engineer. It has only strengthened my already existing enthusiasm and ambition for engineering, as well as my goal to always continue learning and growing as an engineer and a person.

List of contents

List of Figures

List of Tables

Abbreviations

AGV	Automated guided vehicles, working in SWEPs factory.
CAD	Computer aided design, the procedure of building digital 3D models.
Dead man's switch	A switch designed to be deactivated if the human operator becomes incapacitated or leaves the operator area.
FIFO	First-in-first-out, a type of queue processing order scheme.
GSDML	A readable text file containing device-specific identification and communication specifications.
HEX, BPHE	Brazed plate heat exchanger.
HMI	Human-Machine-Interface, a user interface that connects an operator to a device or system.
IE	Industrial Ethernet, a common standard for using Ethernet in an industrial environment.
IRC	An Industrial Robot Controller made by ABB.
MU	Mobile unit, a part that flows between stations in Plant Simulation.
OEE	Overall Equipment Effectiveness, a number from 0% to 100% indicating the effectiveness of a process.
PLC	Programmable Logic Controller, an industrial computer designed for process control in harsh factory environments.
PROFINET	An industry standard for data communication over Industrial Ethernet.
SOP	Standard Operating Procedure, a defined set of instructions made by a company or organization, used by its workers.
State machine	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.
TCP	The Tool Center Point coordinate system for industrial robots in ABB RobotStudio, the center point of the active tool.
TP	True Position, of a measurement, is twice the distance between the theoretical point and the actual point.
XML	Extensible Markup Language, a file format for storing arbitrary data in a defined set of rules and encoding.

1 Introduction

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

1.1 Background

When searching for a well-suited thesis project within electrical engineering and automation, I quickly gained interest in an available thesis project at SWEP International AB[1], presented on their website. They are a well-established and world-leading manufacturer in their field, which strongly captured my interest. I applied with excitement, and, an interview later, was offered the thesis project.

1.1.1 Motivation

This chapter presents and discusses various scientific papers and research in robotics, simulation, and automation. It provides an introductory framework for the thesis work and highlights its relevance and the chosen methodology. The use of industrial automation is skyrocketing across modern industries, making significant improvements to productivity, precision, and repeatability. By automating traditionally manual steps in manufacturing, companies can create increasingly vast and complex production flows. Throughout the manufacturing industry, there is an ongoing rise in companies investing heavily in creating automated solutions within software and simulation environments, allowing them to optimize and test processes and save substantial resources before finally deploying automated solutions to real-world automation tasks[22].

The research performed for this thesis work showed that the importance of simulation and digital analysis in industrial automation is a common and valuable tool across many types of manufacturing industries. For instance, a master's degree project in mechanical engineering by Frasca at Politecnico di Torino[10] performed virtual analysis and simulation of a conveyor cell in view of the changes within Industry 4.0. The study demonstrated the potential and benefits in simulating the cell in Siemens Process Simulate X and Tecnomatix Plant Simulation, Siemens TIA Portal V14 for programming PLC ladder logic, as well as MATLAB for data analysis.

In the International Journal of Advances in Engineering, IJAET, a research paper by Rezende and Richetto[11] presented a robot cell simulation using ABB RobotStudio and TIA Portal to simulate both a robot and a virtual programmable logic controller. The paper highlighted the effectiveness of simulating in RobotStudio and TIA Portal in designing and optimizing automation in robot cells. The paper concluded a successful integration and simulation for the robot cell and highlights the advantages and importance of choosing a good balance between detailed realism and development efficiency to achieve representative and accurate simulation results.

In a paper by Sobrino, Roman, and Holubek at the Slovak University of Technology in Bratislava[12], the risks in commissioning new manufacturing solutions without simulation are emphasized. The paper presents how virtual commissioning in the software Tecnomatix Plant Simulation combined with Siemens TIA Portal and its virtual programmable logic controllers can all be used together to simulate and analyze new manufacturing automation attempts before commissioning them for real. The paper argues that virtual commissioning is a powerful and valuable tool for manufacturing companies.

These studies demonstrate the growing importance of simulation and digital analysis in industrial automation and highlights software options from industry-leading companies like Siemens[15, 16] and

ABB[14]. This thesis project aims to contribute to this body of research and knowledge in industrial automation using these software tools.

SWEP International AB is one company that invests heavily in robotics and automation. Their manufacturing process and product designs contains steps that are well suited to be automated[2].

1.1.2 SWEP International AB

SWEP International AB[1] is a global manufacturing company headquartered in Landskrona, Sweden. It has been a part of the Dover multi-industry company since 1994. SWEP has over 1000 employees, production in five countries, and multiple offices worldwide. In a world where sustainable and effective energy use is rapidly growing in 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[2]. Figure 1 shows three different models of BPHEs from SWEP.

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

1.1.3 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 a 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. It can effectively be used as a condenser, evaporator, gas cooler, and more[2]. 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.

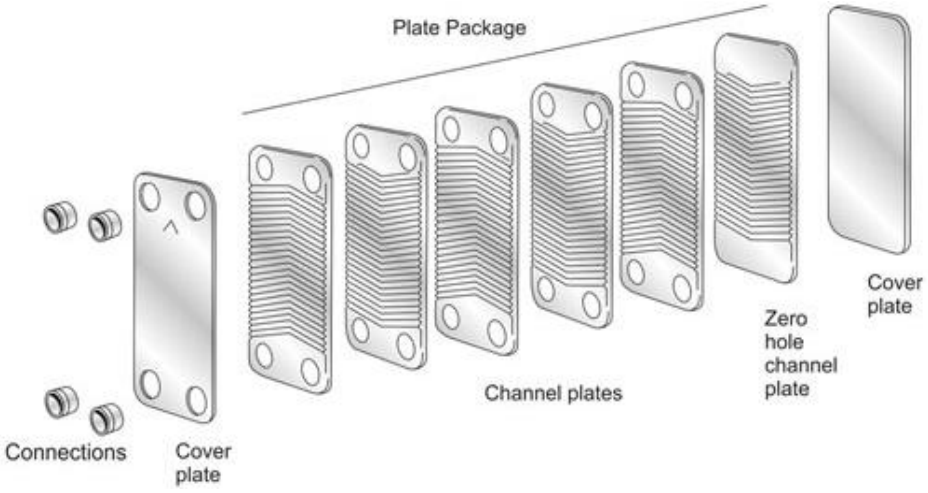


Figure 2: The inner parts of a BPHE. Image from [2].

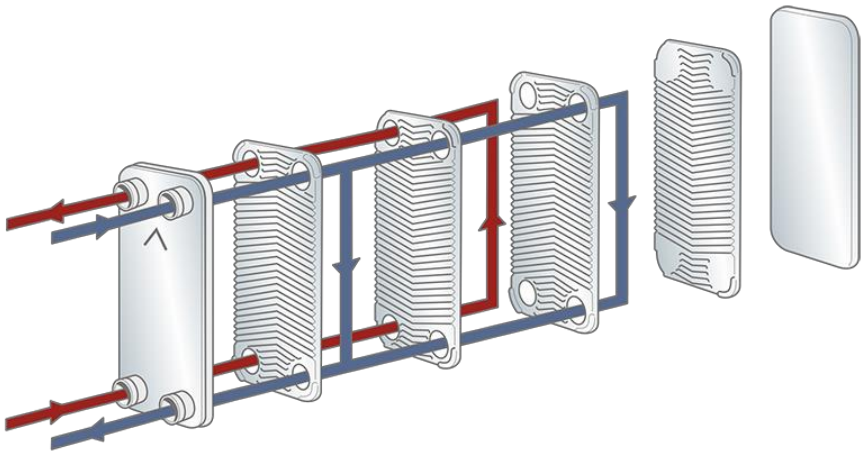


Figure 3: A BPHE used as an evaporator. Image from [2].

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 require 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 become a very cost-effective method[2].

1.1.4 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 heat exchanger models. 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 for manually performing a weld on the P-side by an operator is as follows: First, the barcode scanner is used to scan the models 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.

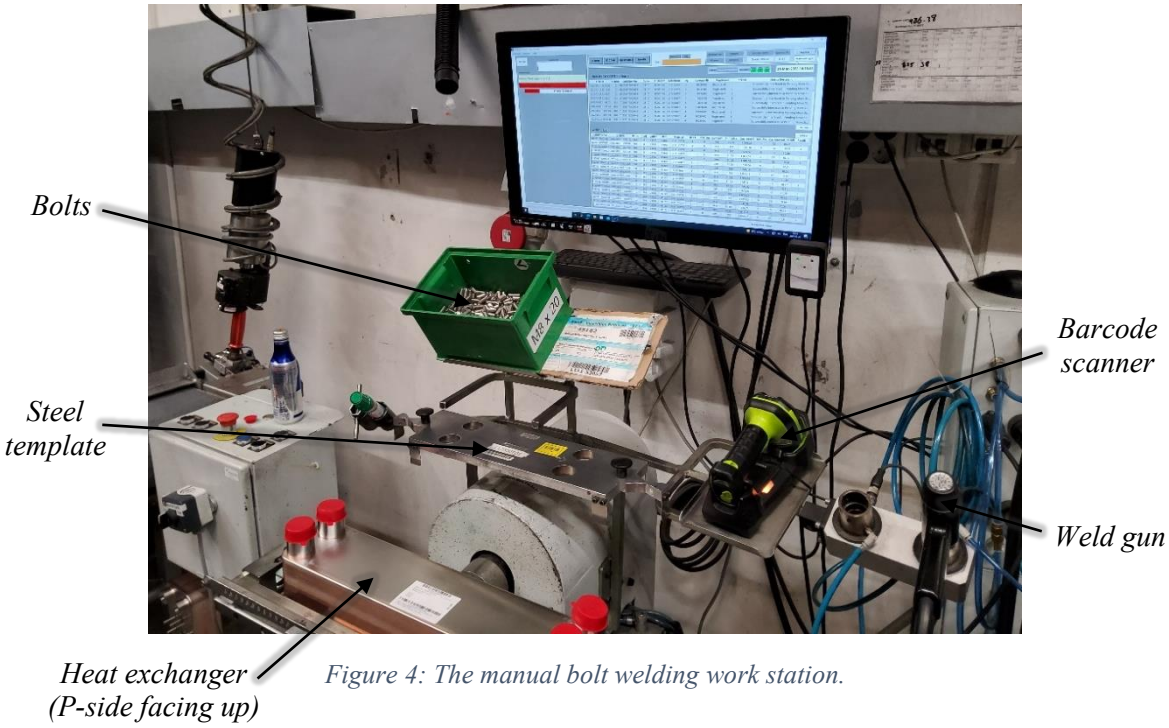


Figure 4: The manual bolt welding work station.

Figure 5: The steel template used to position bolts.

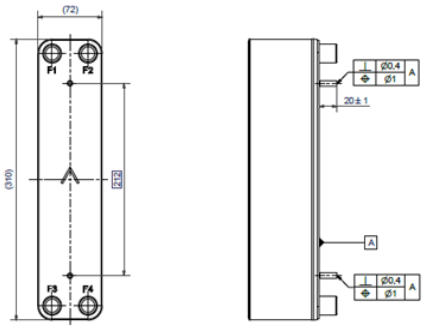


Figure 6: Example of a bolt configuration.

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 resource-expensive and time-consuming and, therefore, is to be fully automated. Some work to automate this process has been completed at SWEP by an external contractor in the form of a simulation and programming of an industrial robot[3]. It forms a semi-automated solution where heat exchangers still have to be manually placed, rotated, and removed by an operator while the robot automatically welds stud bolts on the heat exchanger surface. There is much work to be done before the manual stud bolt welding process can be entirely automated.

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

1.3 The robot cell and pre-existing robot program

In this section, the pre-existing automated state of the robot cell will be introduced and described in detail. An ABB IRB 4600 industrial robot[4] uses an attached measuring probe[5], weld gun, and gripper mechanism to execute the welding process. The pre-existing robot program[3] 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 where the thesis work started, taking over the work done by the contractor from Bravida. The robot operates on the heat exchanger placed on the workstation using its attached tools at the end of the robot's wrist. A stud bolt welding unit from HBS[6] located on the floor behind the robot provides the weld gun with current and gas flow during the weld. The industrial robot controller[7] cabinet behind the safety fence contains the active robot program that controls the robot's movements and tools.

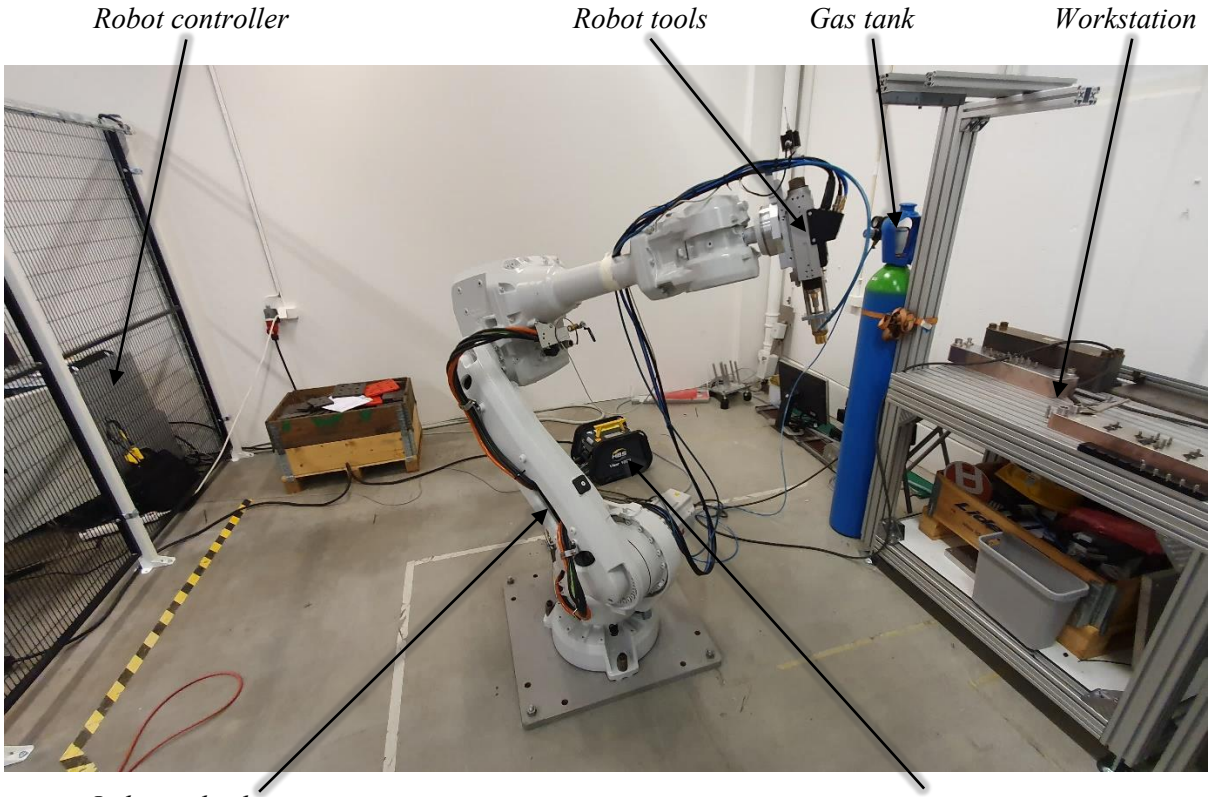


Figure 9: Overview of the robot cell.

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 ensure 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.

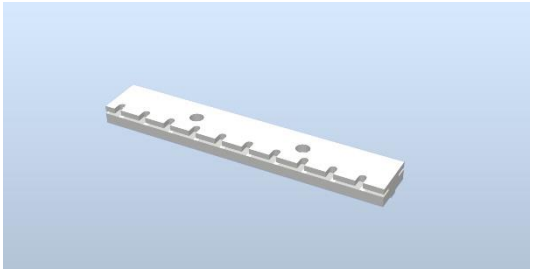


Figure 10: 3D model of the bolt tray.

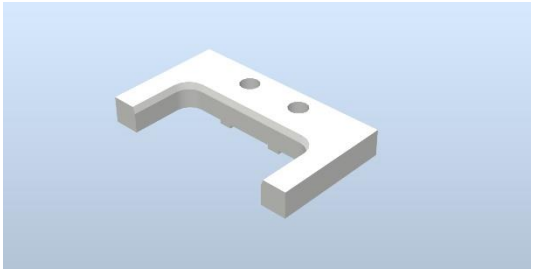


Figure 11: 3D model of the heat exchanger fixture.

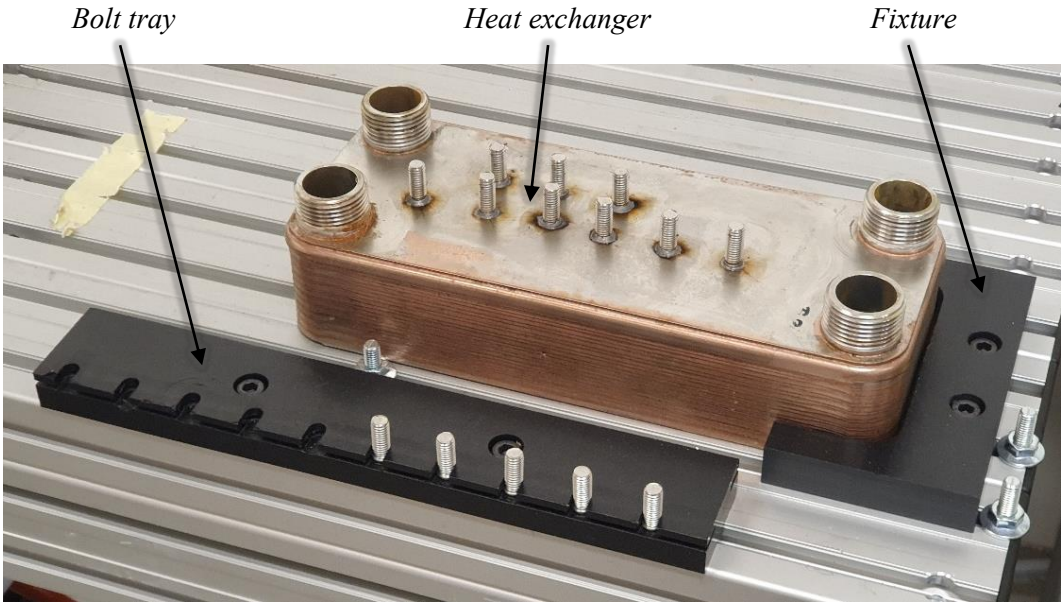


Figure 12: Workstation setup with bolt tray and fixture.

1.3.3 Welding gun and measuring probe

Two separate tools are attached at the end of the robot's 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 probe's metal tip rod is movable as it is connected to an internal spring. The tip moves when it contacts the heat exchanger surface, 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's wrist, as seen in Figure 15. The robot can then rotate the tool frame and use both tools.

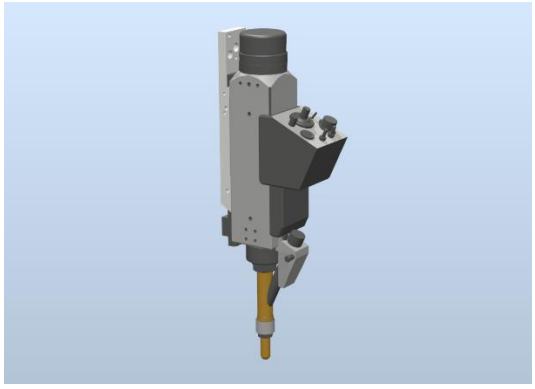


Figure 13: 3D model of the weld gun.

Figure 14: The touch probe. Image from [5].

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 it to

operate close to other machines and workstations, saving space and thus expanding productivity. Figure 15 shows the robot and its attached tools.

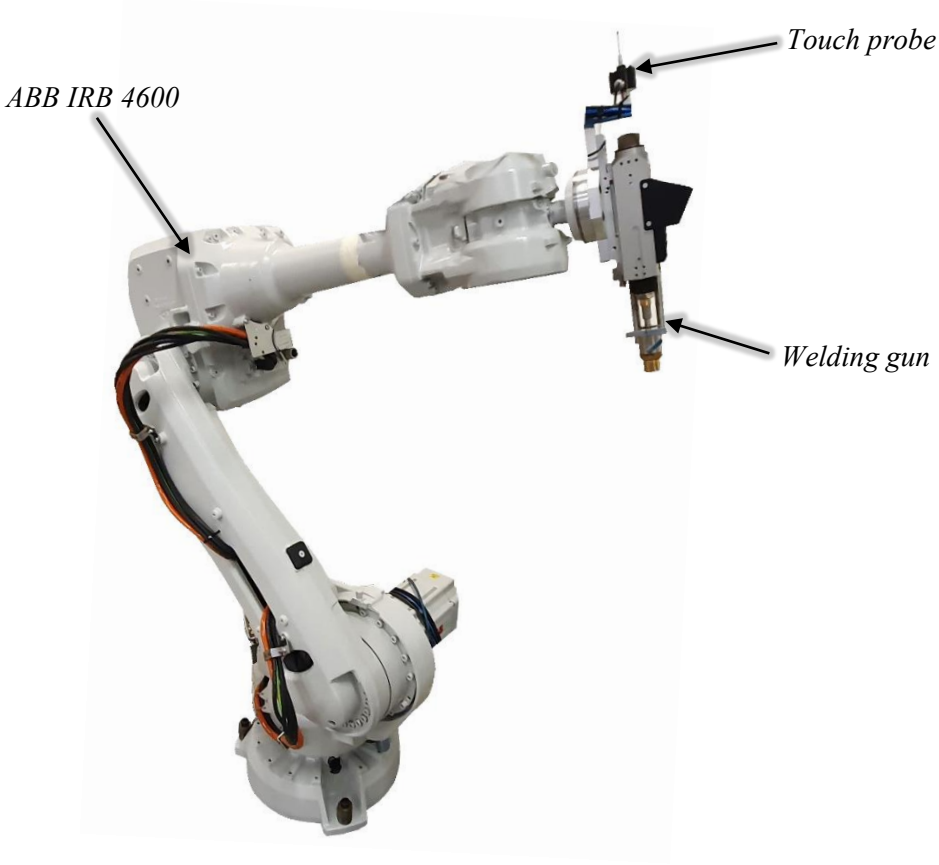


Figure 15: The industrial robot used in the welding process.

1.3.5 ABB 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 modes: automatic and manual. In automatic mode, the robot controller operates the program independently without needing 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. The speed is limited in manual mode, and the program may only continue while an operator is present. This mode should be used while evaluating 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[18] 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 primarily used in manual mode. Its enabling switch on the back acts as a dead man's switch, forcing the operator to continuously hold down the switch during program execution. As soon as the switch is released, all robot movements halt.

1.3.7 The HBS 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, such as the 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 for externally exporting the data monitored. To monitor these parameters in an external controller they must be measured and imported separately.

Using an HBS welding unit ensures the welding result is 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. [18]

1.3.8 Grounding the heat exchanger

During the weld, the heat exchanger must be completely stationary. Also, it must be electrically grounded to prevent any current from flowing through the heat exchanger from the weld gun. Since the magnitude of the 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 put to work in the factory, this way of grounding the heat exchanger will 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 pre-existing robot program, two methods for center calculation

This section will introduce the pre-existing robot program. More in-depth details will be presented on how the overall algorithm works, how the robot accurately searches for points, and how the center reference and orientation are calculated. Figure 19 shows the program's initial starting position and the workstation's configuration. 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.

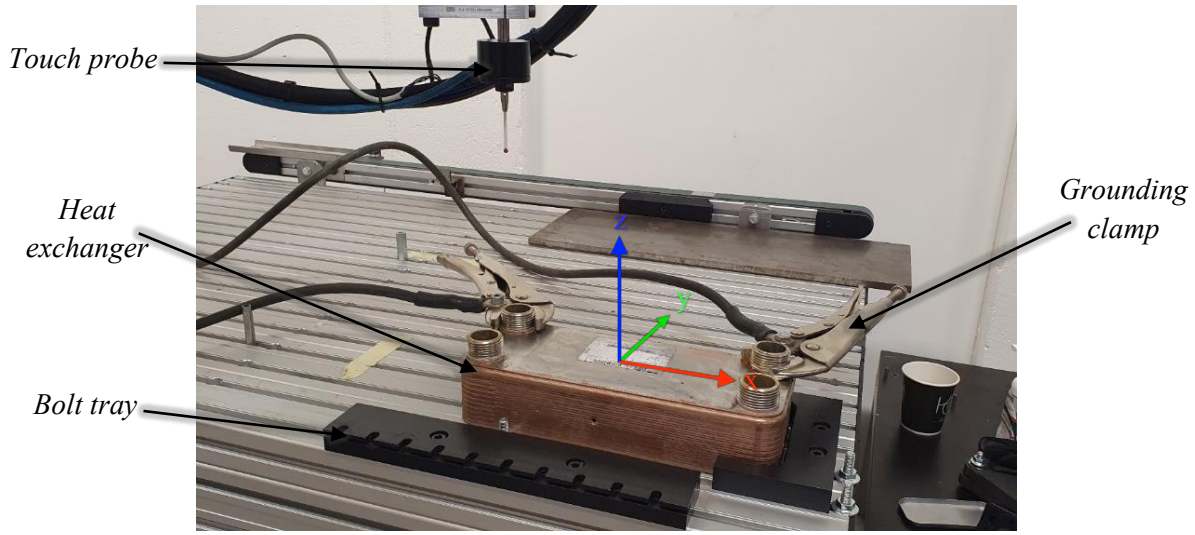


Figure 19: The welding program starting conditions.

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. The program then calculates the surface plane with these three points saved by the robot controller. 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 by 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 measure the side point accurately. When these points have been measured and saved to the robot controller, the orientation and center reference point can be calculated.

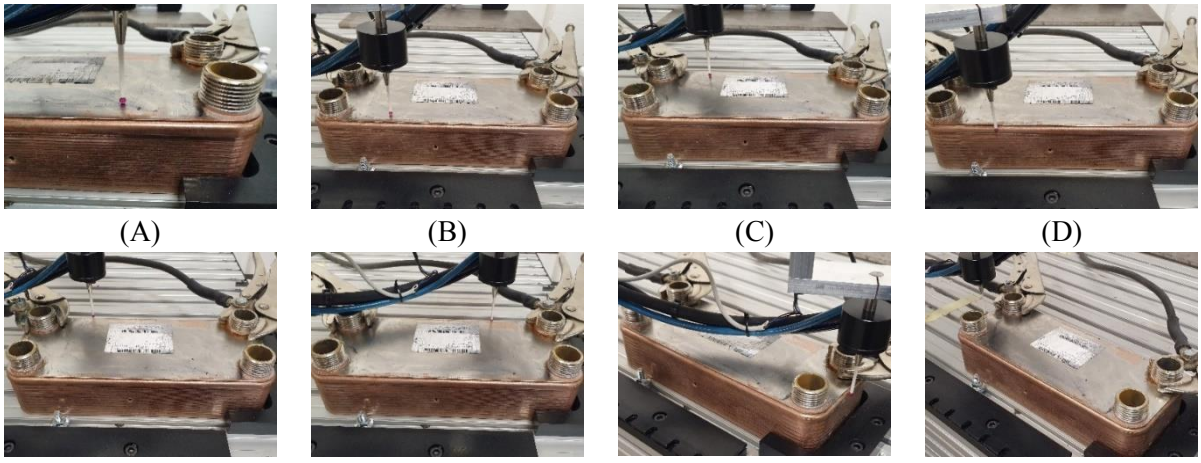


Figure 20: Points measured to calculate the heat exchanger's center reference point and orientation, Method I.

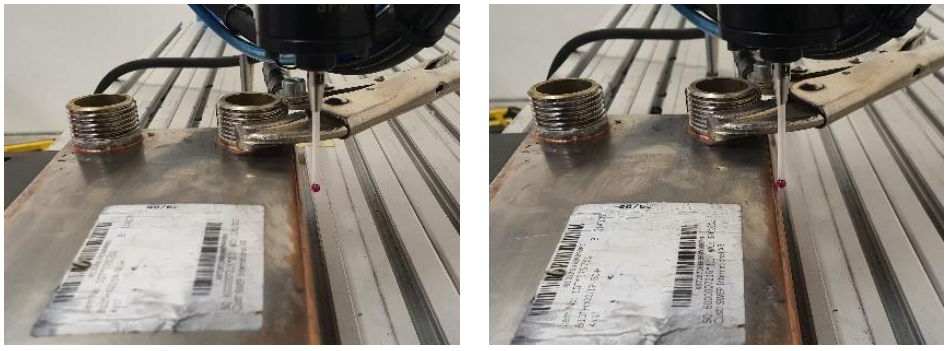


Figure 21: A side point being approached by the probe.

To calculate the orientation and z-coordinate of the surface plane, two vectors on the surface plane are defined from the three surface points A, B, and C, 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 G and H. Save the x-value of this new point. Calculate the average point of E and D. 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 E to F. Calculate the angle between this vector and the x-axis. Finally, rotate the previously calculated surface plane around the z-axis with the computed 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.

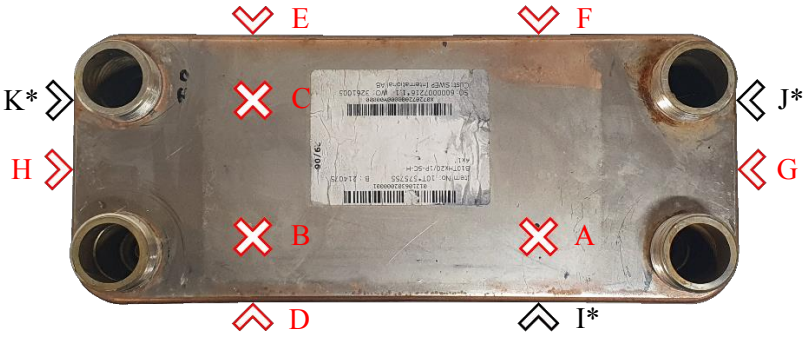


Figure 22: The measured points saved to the robot controller.

* 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 A, B, and C, as described in Figure 20. However, the algorithm for calculating the center reference point and 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 I, J, and K are measured as seen marked with an asterisk in Figure 22 (*).

To calculate the center reference point, the following algorithm is used: Calculate the line through G and K, as well as the line through J and H. Calculate the intersection point of these two lines. Save the x-value of this intersection point. Calculate the line through D and F, as well as the line through E and I. 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.

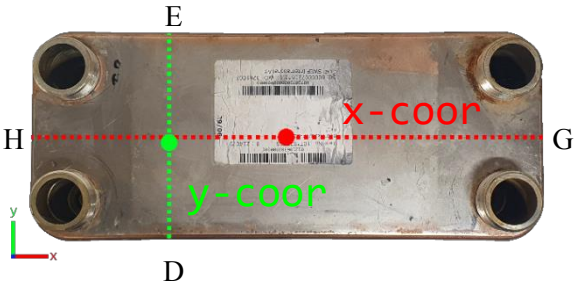


Figure 23: Calculation of the center point, Method I.

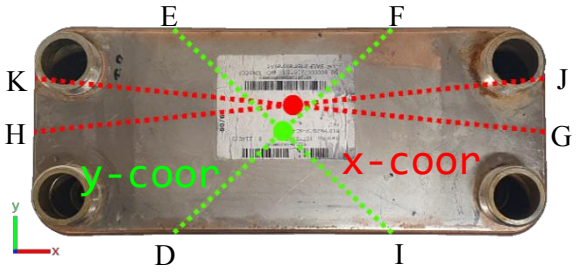
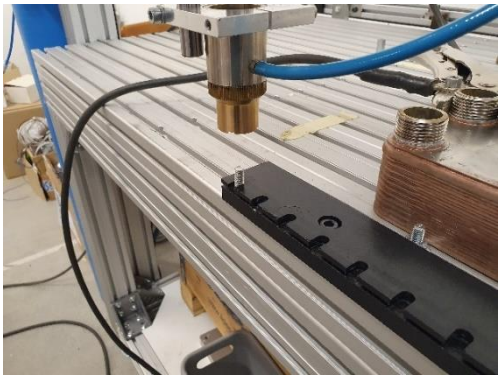


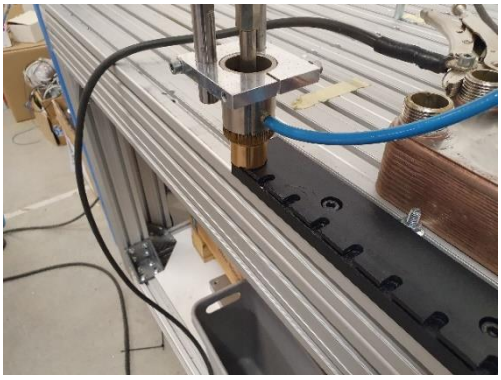
Figure 24: Calculation of the center point, Method II.

Performing the welds

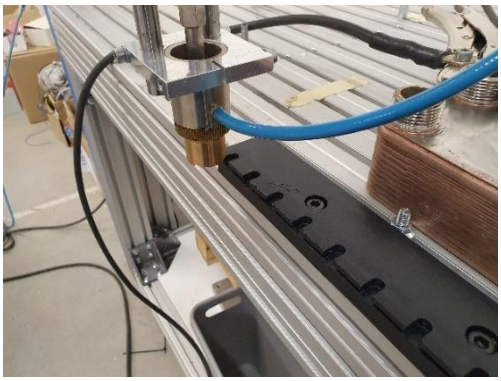
In Figure 25, the program function for welding a bolt onto the heat exchanger is summarized. This function can only be used after the center reference point and orientation measurements have been performed for the used heat exchanger model. First, the weld gun mounted on the opposite side of the touch 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.



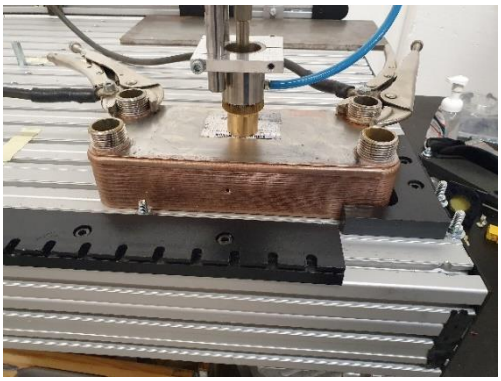
(a) Weld gun above bolt tray position



(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 task objectives of the thesis work in continuing the work of automating the stud bolt welding process.

1.4.1 Perform quality analysis for automated welding

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 pre-existing robot program

The pre-existing robot program includes a precise calculation of the heat exchanger's center point and rotational offset, and these algorithms should be studied, verified, and evaluated. The program contains two different methods of calculating the center point, and their accuracy should be measured and compared to determine which method should be used in the finished automated process.

Also, an attempt to improve the robot program should be made if the evaluation or measured accuracy determines that there is a need for it. A suggestion from SWEP was to create a new RAPID program that mounts the heat exchanger horizontally instead of vertically, allowing the robot to access both the P- and F-side simultaneously without requiring an operator or second system to rotate between

them. However, the now asymmetric effect from gravity must be analyzed to see if gravity affects the weld in its (liquid weld state) cool-off period, as tolerance and weld quality must be maintained.

1.4.3 Implement a PLC control structure

A fully automated control and communications system between the HBS welding unit, IRC, a PLC[8], and a HMI panel[9] 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 number 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 the safety system within 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. 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 an HMI for operator control

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. 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.

1.4.6 Investigate the possibility of using the welding gun with multiple bolt sizes

Stud bolts at SWEP are needed in various lengths. The automated welding gun supports a range of bolt lengths, but it is currently unknown if multiple sizes can be used in operation at the same time without the need to adjust mechanical settings, which might prove difficult to do automatically or perhaps affect positional tolerance and need of recalibration when switching between lengths. This should be investigated as well.

1.4.7 Simulate an automated cell in SWEPs factory-wide material flow model

SWEP has created a factory-wide production flow simulation, where the future vision of the automated stud bolt robot cell needs to be built and programmed, as well as integrated and simulated in SWEPs factory-wide simulation model. This includes creating a representative 3D model of the cell and hardware as well as programming to simulate all robot movements, material flow and operator tasks in the model software. Then, the simulated cell must be tested and verified using real configurations to confirm it is a representative model of the automated stud bolt welding cell and its various cycle times. Finally, the cell must be integrated in to SWEPs main factory-wide simulation model to run simulations.

1.5 Problem formulation

The section presents the purpose and objectives already described 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 pre-existing robot program be improved? Is it possible to weld horizontally, and if so, will gravity affect the weld quality?
- 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?
- Is it possible to use the welding gun for different bolt lengths in an automated solution?
- What tasks will be left before the process can be fully automated and replace today's manual process?
- How can the upcoming fully automated cell be programmed and simulated in SWEPs production flow model to simulate its material flow and cycle times?

1.6 Limitations

This section presents the limitations of the thesis project and were decided early on from discussions with the supervisor at SWEP as well as the supervisor at LTH.

- The HMI and PLC hardware used must be a Siemens KTP400 Basic HMI and Siemens SIMATIC S7-1200 PLC, respectively.
- Any HMI and PLC programming must be done in Siemens TIA Portal
- Any changes or additions to the robot program must be done within ABB RobotStudio
- Testing the HMI and PLC with the robot program must be done within the existing testing robot cell at SWEP

2 Method

This chapter presents the chosen methods to complete the defined objectives of section 1.4. An important mention is that this twenty-week thesis project was done in two periods: one fifteen-week period in 2022 as a bachelor thesis[13] and then further built upon during a five-week period in 2025 to combine and form my master thesis project.

Performing quality analysis for automated welding

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 welding unit. To understand how and why each weld parameter affects the welding quality, every parameter will be evaluated 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 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 will be made to improve any part of this algorithm. 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 with a PLC

To control the welding process in a fully automated solution, a Siemens SIMATIC S7-1200 programmable logic controller (PLC) and a Siemens KTP400 Basic Human-Machine-Interface (HMI) will be used. The PLC and HMI will be programmed in 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 internal 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 the safety system within the welding unit

The HBS Visar 1200 model operating manual will be studied to implement a fail-safe system for the welding unit. 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 an HMI for operator control

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.

Investigate the possibility of using multiple bolt lengths

To find out if multiple bolt lengths can be used in a fully automated solution, the technical manual of the welding gun was studied. A contact with SWEPs provider of the welding gun, BM Svets AB, was also held regularly as a support to understand the manual and discuss if a solution was possible. After fully understanding how the welding gun works and how it can be mechanically calibrated to new bolt lengths, several bolt welds were performed across all available lengths at SWEP to find the shortest and longest bolts possible to use with the welding gun.

Simulate and integrate the automated cell in SWEPs material flow model

To build a module in Plant Simulation that simulates the upcoming fully automated cell, full understanding of material flow, robot algorithms, cycle times, etc, was crucial. First, a meeting with engineers responsible for the automatization project is needed to receive a detailed explanation of the cell and to answer any questions or clarifications needed. From this, the module can be built and programmed in Plant Simulation. After finishing the programming, a representative 3D model will be created in AutoCAD Inventor and included in the program. The module will then be simulated in Plant Simulation with real product data, both locally within the cell and then integrated into SWEPs factory-wide simulation model, to validate its performance and cycle times.

3 Automation tools

This chapter will introduce the three software programs used heavily throughout this project: ABB RobotStudio[14], Siemens TIA Portal[15], and TecnoMatix Plant Simulation[16].

3.1 ABB RobotStudio

3.1.1 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 completely isolated from the real robot cell without disrupting or modifying 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.

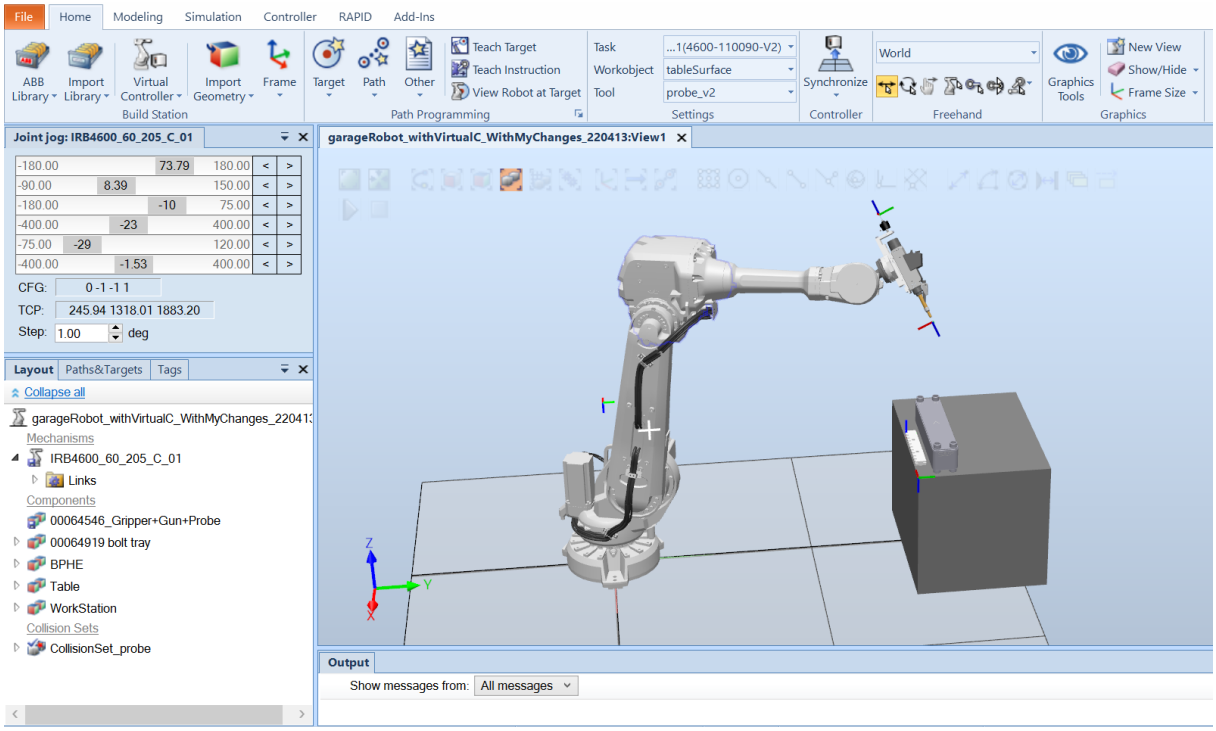


Figure 26: The RobotStudio simulation environment.

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. Capable of simulating the IRC5, the pre-existing robot program could be tested and investigated early on, even when not working with the physical robot cell.

3.1.2 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). This section explains the most relevant variable types and instructions of the RAPID programming language within this project.

Basic RobotStudio objects:

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 axes) to place the tool at some point vector in space, a solution may be chosen as the "axis configuration" used, defining what rotational quadrants the six axes 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 moving toward the next position, letting it "cut 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 programming 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.

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 specified amount of time.

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

SetDO: Changes the value of a digital output signal.

3.2 Siemens TIA Portal

3.2.1 The development environment

The Totally Integrated Automation software platform from Siemens, TIA Portal, offers a solution of centralizing all development work for Siemens PLCs, HMIs, drivers, and PROFINET[17] 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[18] for simulating a PLC, along with Siemens SIMATIC WinCC[19] 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.

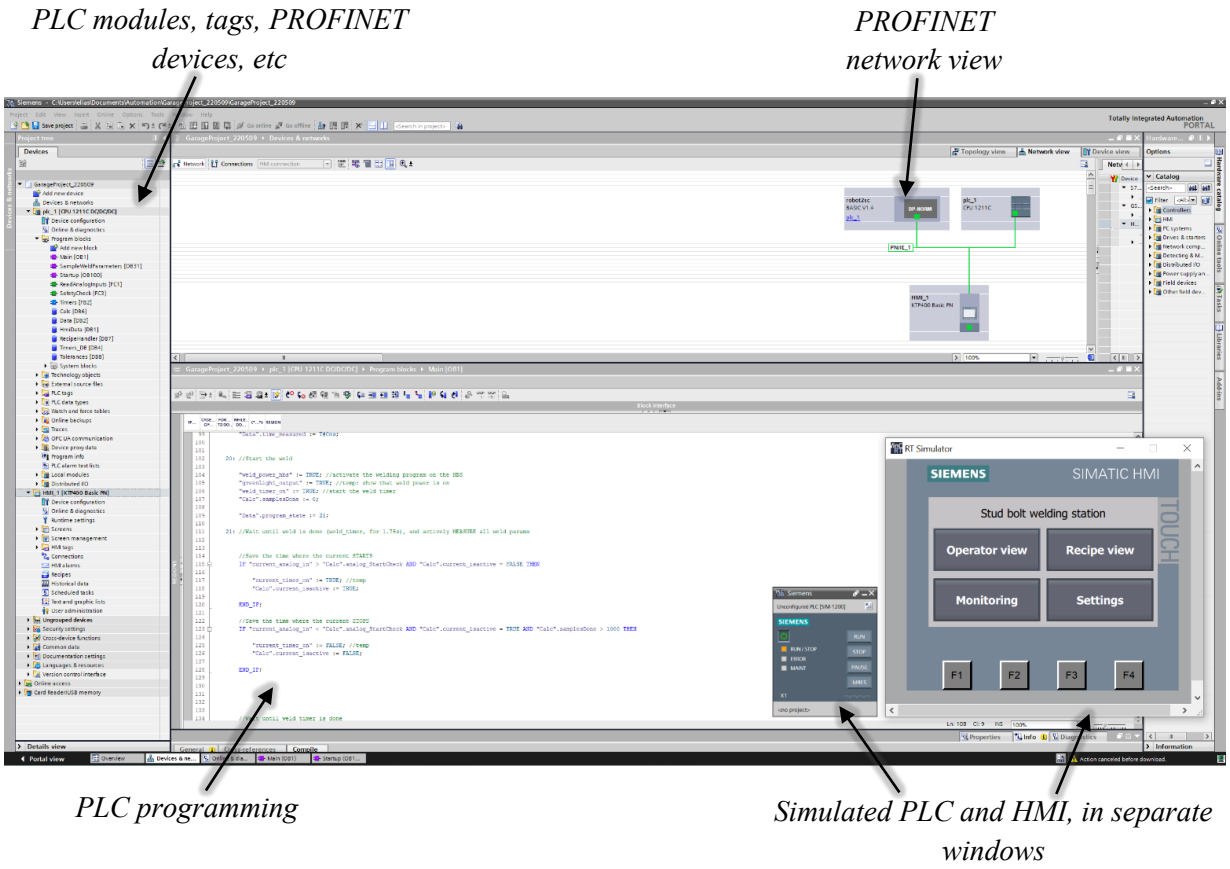


Figure 27: TIA Portal with simulated PLC and HMI.

3.2.2 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[14]. 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.

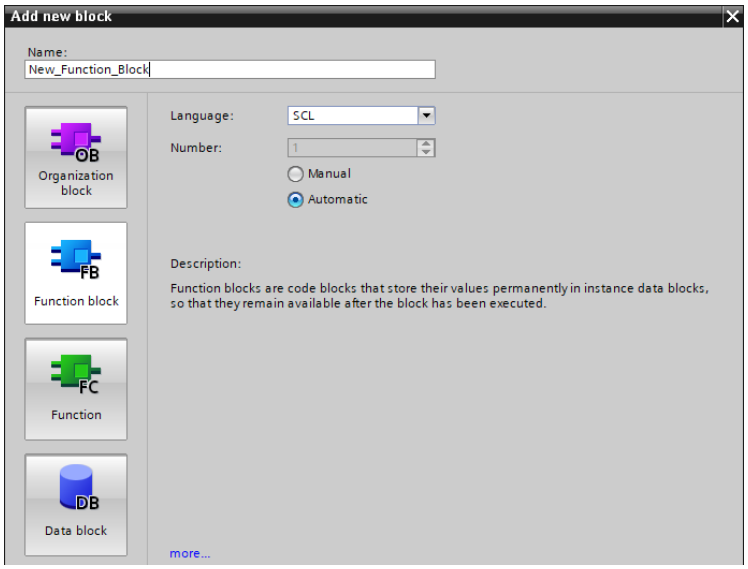


Figure 28: TIA Portal – the four types of program blocks.

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 and 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) has the prefixes %IW, %QW, and %MW. Following the prefix, the bit or bit's addresses are defined. For example, %I68.1 refers 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		☑	☑	☑	
2	weld_power_hbs	Bool	%Q0.4		☑	☑	☑	
3	weld_done	Bool	%Q2.0		☑	☑	☑	
4	weld_timer_done	Bool	%M0.2		☑	☑	☑	
5	weld_timer_on	Bool	%M0.3		☑	☑	☑	

Figure 29: TIA Portal - PLC tags.

3.2.3 Creating an HMI layout

In TIA Portal, an HMI layout can be designed 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.

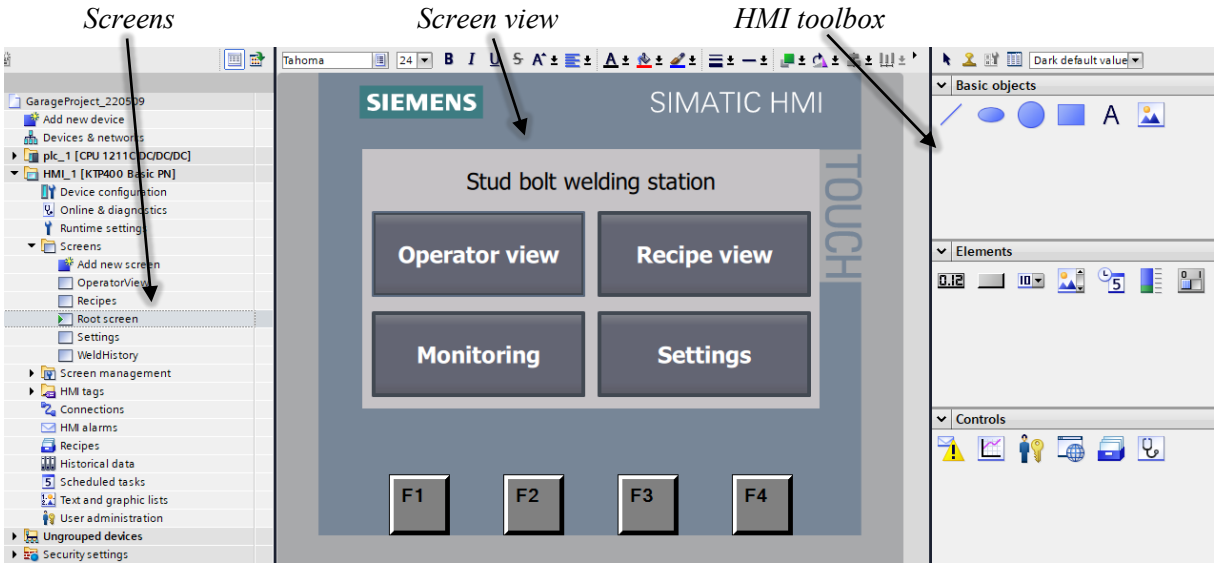


Figure 30: TIA Portal - HMI development.

Other than the basic objects, TIA Portal offers more advanced HMI elements, which are often linked with PLC tags and variables. Some of the HMI elements used within the thesis work are:

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 push button 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.

3.3 TecnoMatix Plant Simulation

3.3.1 Working with Plant Simulation

TecnoMatix Plant Simulation is a program that can model, simulate, visualize and analyze complex production flows and their logistics. In Plant Simulation, companies can build a simulation of their production process, including objects like robot cells, conveyors, material buffers, workstations, autonomous vehicles, operator duties and schedules, cycle times, and much more. Even a very abstract and simplified model of real processes opens the possibility of simulating factory scenarios and drawing valuable results about its processes. Alternatively, companies can use Plant Simulation to simulate down to any desired depth of detail, simulating every piece of equipment or material flowing in the factory at any point in time. Plant Simulation also offers tools for analyzing energy usage, random failure and recovery, bottleneck analysis, and more. Figure 31 shows an example of a factory simulation created by TecnoMatix. To protect confidentiality, the entire SWEP production model can not be shown in the thesis report. Figure 32 shows the Plant Simulation development environment used in the thesis work, and an early version of the stud bolt robot cell simulation created in this thesis work. The 3D simulation view shows the simulation and allows for live interactivity. The console serves as a valuable debugging tool and data output. The class library contains an extensive list of native Plant Simulation objects to build up a model with and will be presented more in detail in section 3.3.2.

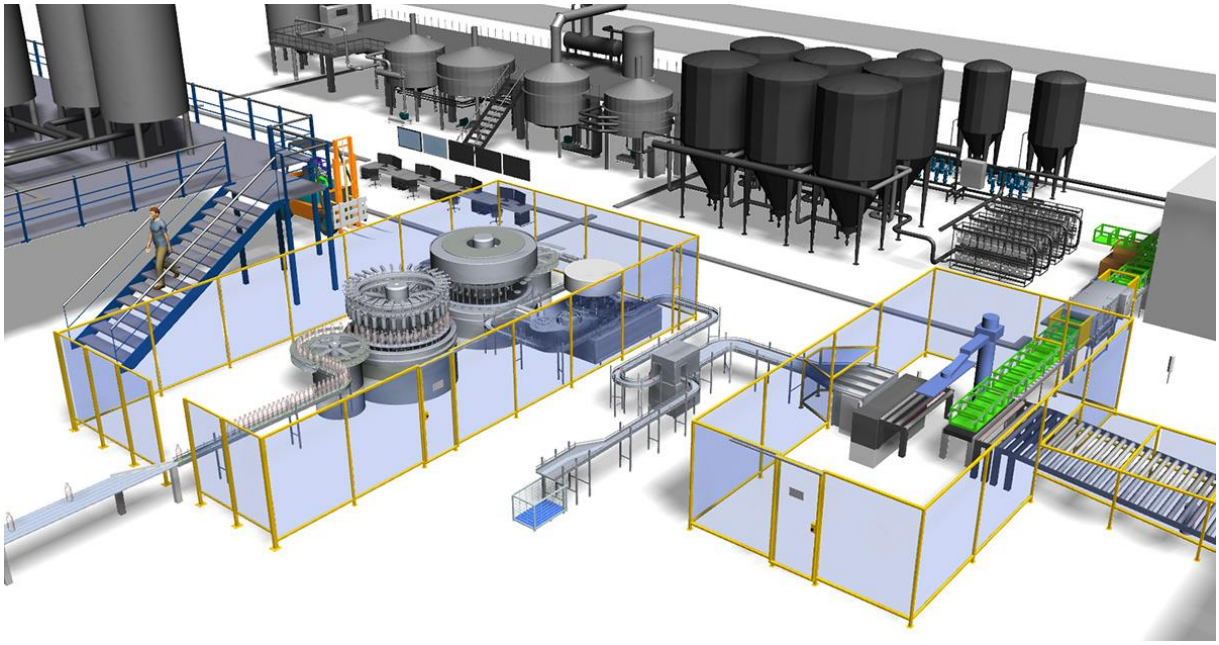


Figure 31: Example of a factory model in Plant Simulation.

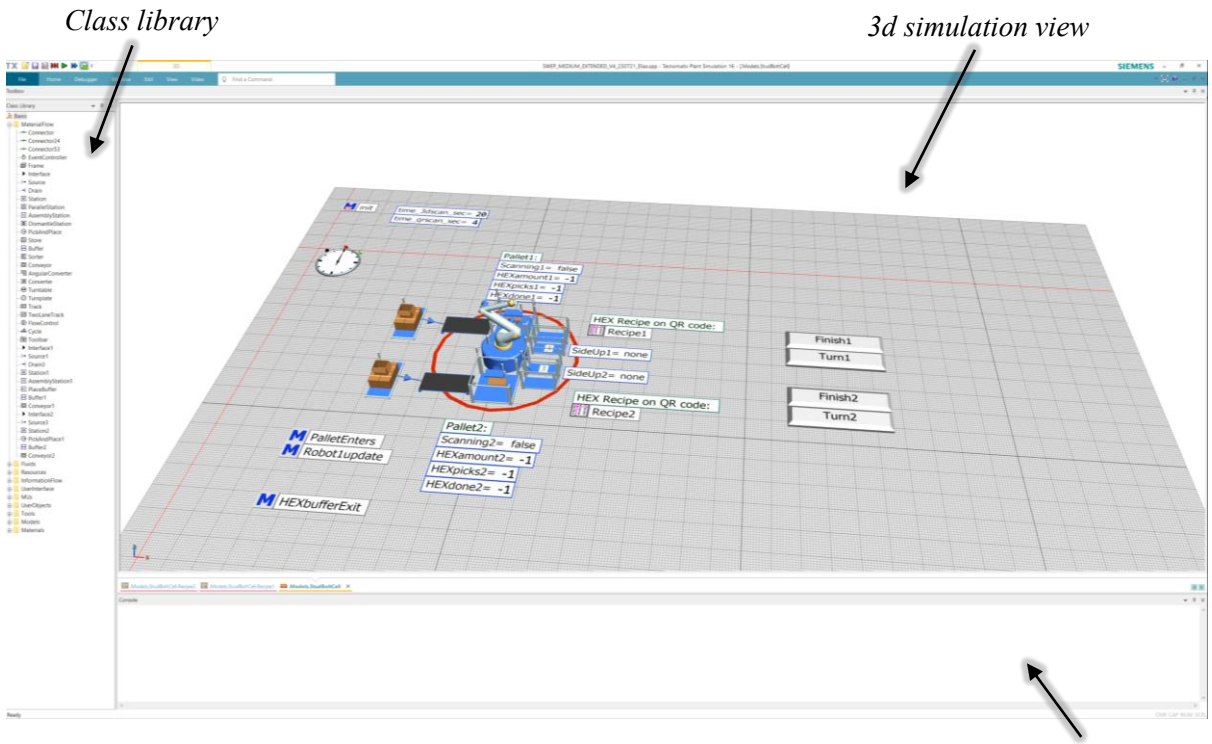


Figure 32: The Plant Simulation development environment.

Figure 33 shows the control window of a common object in Plant Simulation, a robot, serving as an example to highlight the deep level of control that Plant Simulation offers. In *Attributes*, settings and controls regarding robot joint movements can be set. In *Failures,* automatic and random failures of equipment can be chosen to simulate realistic process failures. In *Controls,* programming methods and functions written in SimTalk can be specified to run when a moving part enters, exits, or is operated on by the robot, among other similar controls. In *Exit,* different flow types can be used to simulate blocking, waiting queues, and similar behavior when the next station cannot receive the part. In *Statistics*, data and statistics around the robot are logged, such as worked time, failure times, blocking frequencies, etc. In *Importer,* settings regarding factory worker interaction are set, choosing what services for the robot are done by operators, which operators to request, etc. In *Energy,* the robot's power usage in its various operational states can be set. In *Costs,* financial details of the robot cell can be set, including investment costs and operating costs.

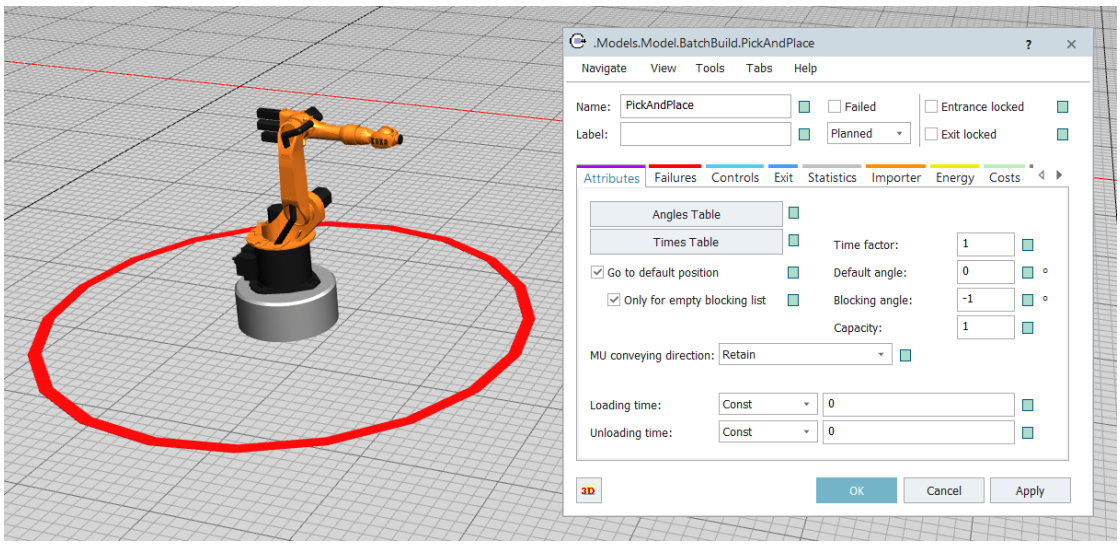


Figure 33: Example of available controls in Plant Simulation.

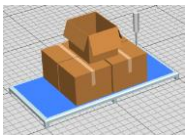
For SWEP, building a detailed model of their heat exchanger production line offers them the opportunity to answer questions like:

- How do productivity, costs, and energy usage change if another cell is added or removed?
- What is the minimum number of operators to preserve productivity?
- How do specific errors and unproductivity affect the processes further down the line?
- What kind of bottlenecks exist, and what causes them?
- How will the current factory handle future scenarios and tricky production orders?

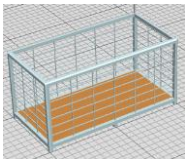
Answering these types of questions in simulation alone is of high value to SWEP in making decisions for future factory scenarios. Of course, the quality of the answers depends highly on the abstraction and level of detail put into the simulation. As a seasoned engineer might say, garbage in means garbage out. Therefore, when creating models in Plant Simulation, one must make careful planning in designing, programming, and choosing abstraction within the model so that it as much as possible represents the real process in the factory.

3.3.2 Common class library objects in Plant Simulation

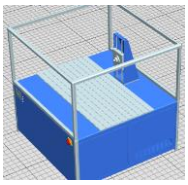
The class library seen in Figure 32 contains a large set of objects native to Plant Simulation and is often enough to simulate any type of factory behavior. Their 3D graphics are native but can easily be replaced by a CAD of choice. A short introduction to the most commonly used objects in Plant Simulation is presented below for the curious reader or a developer new to Plant Simulation. All of them were used in the thesis project, but their details are not crucial to understand the report and may be safely skipped.



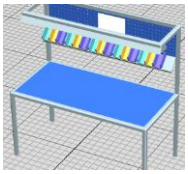
Source: spawn's mobile units, MUs, with a chosen behavior. It could simulate the delivery of goods from outside the factory.



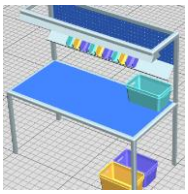
Drain: deletes MUs that enter it. Every MU spawned should eventually be deleted to preserve model performance, either by code or in a Drain. It could simulate shipping from the factory or general material use.



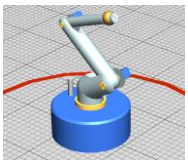
Station: may represent any station or machine on which MUs spend a certain amount of time to be processed. It could simulate a station where operators must approach to manually weld bolts on the attached MU.



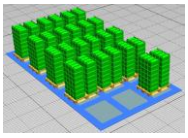
Assembly station: adds mounting parts to a specified main part. It could simulate a station where channel plates are stacked with two cover plates and together is outputted as a single new heat exchanger MU.



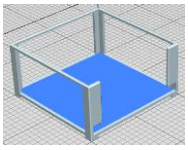
Dismantle station: removes mounting parts from a main MU, or deletes it and spawns new MUs. It could simulate dismantling a heat exchanger MU back to its individual part MUs.



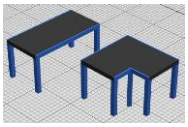
PickAndPlace: a simple pick-and-place robot that picks up one or several MU at one station and moves it to another station. It can be controlled in detail in code to produce a more complex robot operation.



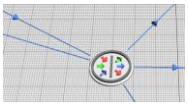
Store: stores any number of MUs. It could simulate a storage of many heat exchanger MUs.



Buffer: temporarily holds one or several MUs for a station where the next station is blocked or has failed. It can also move parts forward when the current station stops working. It could simulate a table where heat exchangers are placed, a place where a pallet is placed, or similar.

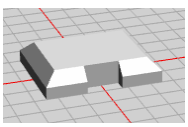


Conveyor: models a conveyor transporting one or more MUs in a queue, at some desired speed. It could simulate a conveyor transporting heat exchangers.

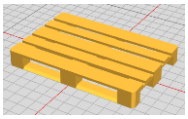


Flow Control: models common flow and blocking behaviors for MUs. For example, a FIFO queue, a cyclic flow, a random flow, etc. It could simulate the logic for distributing heat exchangers equally or randomly to multiple stations to simulate real factory behavior.

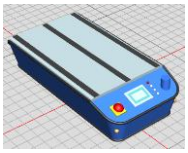
Worker: models a human person working in the factory. It needs a Broker object and ShiftCalender object that controls how work is requested and what defines the worker availability. It could simulate an operator with a set working schedule who walks between stations to do his job.



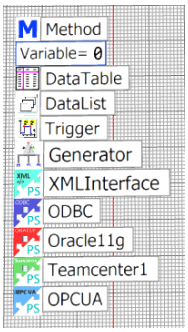
Part: the most common and most basic MU in a factory. It could simulate a heat exchanger, a channel plate, a bolt, or any other object that flows in the factory.



Container: just like a Part but with an added possibility of storing other MUs while it flows. It could simulate a pallet of heat exchangers that move between stations or a heat exchanger with each channel plate as its own simulated object within it.



Transporter: an active mobile material flow object. A Part or Container only moves through other objects like Stations or Buffers, but a Transporter is self-propelled and moves freely on defined paths. It can load and transport Parts, Containers, and other Transporters.



Programming and data objects: a large part of the work while working in Plant Simulation is done in programming to simulate any behavior that the native objects presented can not fully cover. The programming language is called SimTalk and is an application-specific integrated scripting language. Any object has built-in properties and attributes accessible through code, providing many useful features. There also exist objects for common programming features like methods, functions, arrays, loops, and objects that simplify connecting the simulation to XML files, ODBC, Oracle, TeamCenter, OPCUA, and more.

4 Analysis, design and implementation

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 Evaluating the quality of automated welding

4.1.1 Performing the tests

A series of tests were performed for each welding parameter to understand how and why they affect the quality of a weld. 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. Figure 34, Figure 35, and Figure 36 show the parameter values chosen and their resulting welds. The weld collar is seen to become larger and also create more splatter with higher weld energy, that is, high weld current and or high weld time. A higher gas flow is seen to remove oxidation around the weld collar, decrease splatter, and result in more even weld collar quality.

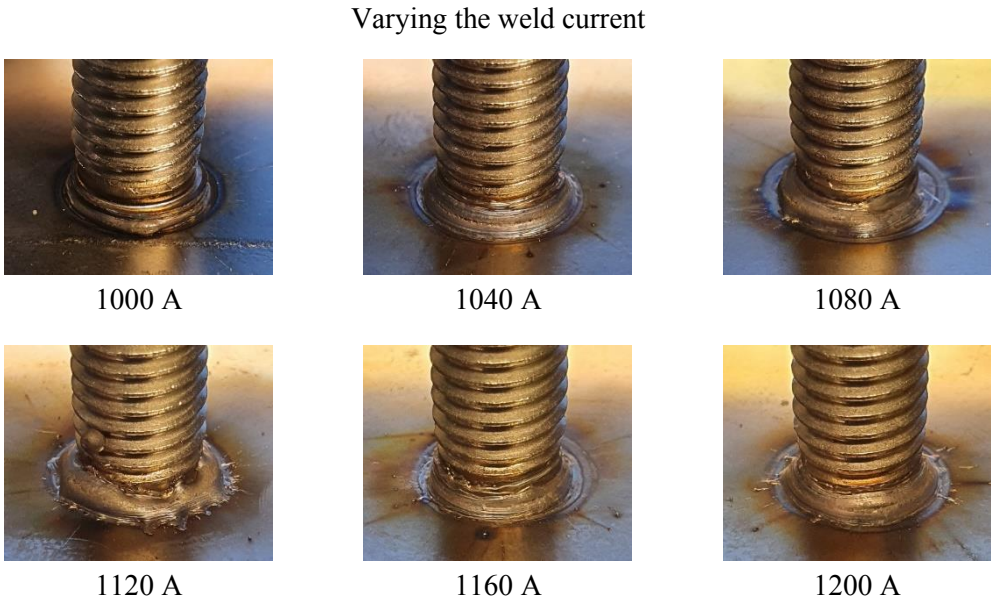
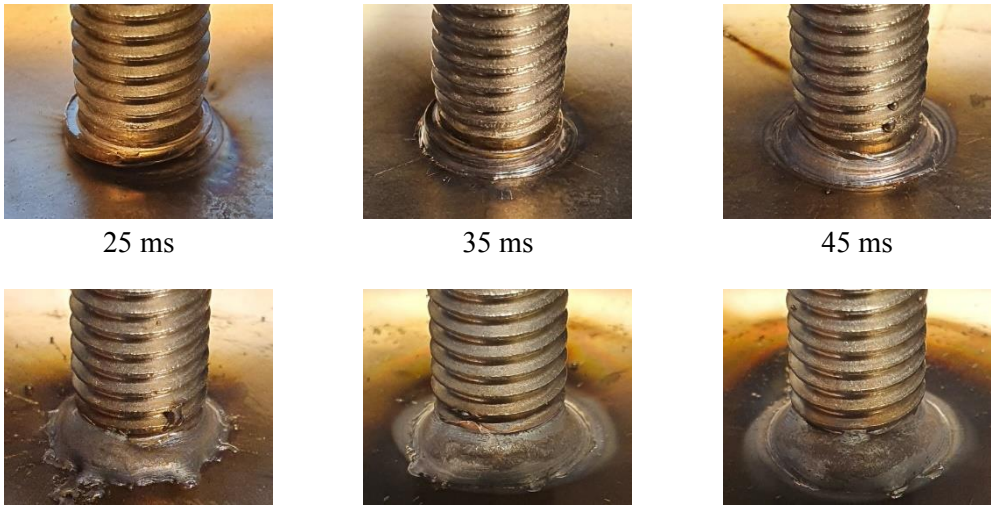


Figure 34: The weld quality tests, varying the weld current.

Varying the weld time



25 ms | 35 ms | 45 ms

55 ms | 65 ms | 75 ms

Figure 35: The weld quality tests, varying the weld time.

Varying the gas flow

No gas | 15 liters/min | 25 liters/min

Figure 36: The weld quality tests, varying the gas flow.

4.1.2 Visual and physical inspection

SWEP provided a standard operating procedure document that explained how to examine the weld quality that the operators of the manual welding process regularly use. 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 were given for the kind of fault. After completing the visual test, physical bending and torque tests were performed on each welded bolt.

The torque test was performed using a torque wrench, as seen in Figure 37. 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 38. 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 37: Torque wrench used in the physical tests.

Figure 38: Bend equipment used in the physical tests.

4.2 Studying the pre-existing robot program

4.2.1 Studying the program algorithm

The pre-existing robot program code was studied early on to be able to work with and later expand its development. Then, before operating the program, the necessary skills for operating the robot and robot controller were learned. This included safety around the robot cell, how to use the FlexPendant as a valuable and safe debugging tool in manual mode, controlling the gas tank, and general good practices for 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 39, the inclination of the heat exchanger can be seen to be accounted for by the robot. In Figure 40, the calculated center point of a larger model is shown by the touch probe.

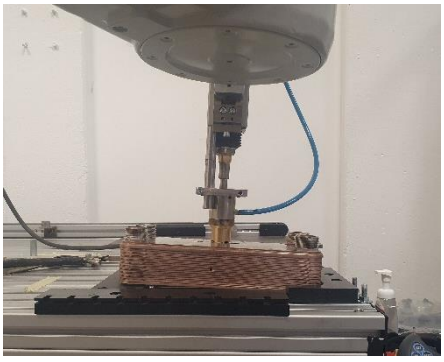


Figure 39: Inclination of a heat exchanger accounted for.

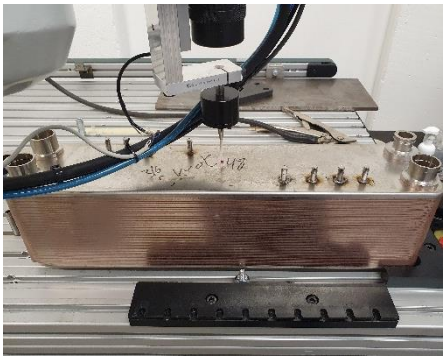


Figure 40: The calculated center point showed by the probe.

4.2.2 Creating a 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

Three bolt welds were performed using each method to compare the two methods of calculating the center reference point and orientation. The full measurement procedure and calculation of the center reference, as described in subsection 1.3.9, were executed again before each weld. If the center had been calculated only once 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. Three similar welds for each method were performed 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 41, was used to measure the actual positions of the bolts with the 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 follows the actual bolt threads towards the top of the bolt, resulting in a very fine tolerance and high accuracy. 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 6, 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, these deviations were used when investigating methods to improve the overall accuracy. 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.

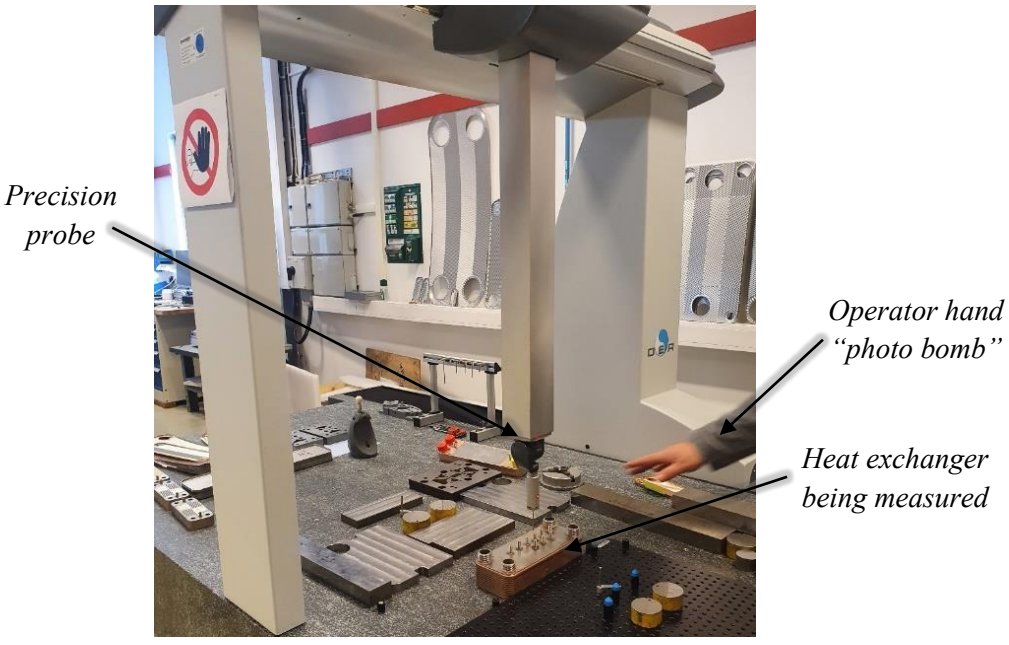


Figure 41: The machine used in the lab to measure the bolt position accuracy.

4.3 Designing a control structure and programming the PLC

4.3.1 Using a Siemens S7-1200 PLC

While the RAPID welding program is running and wants to perform a weld, the robot controller must signal to activate the HBS welding unit program to perform a stud bolt weld. To implement this control of the weld gun power, the PLC will be used and given complete control of the main program flow and the welding unit. Figure 42 shows the implemented control structure.

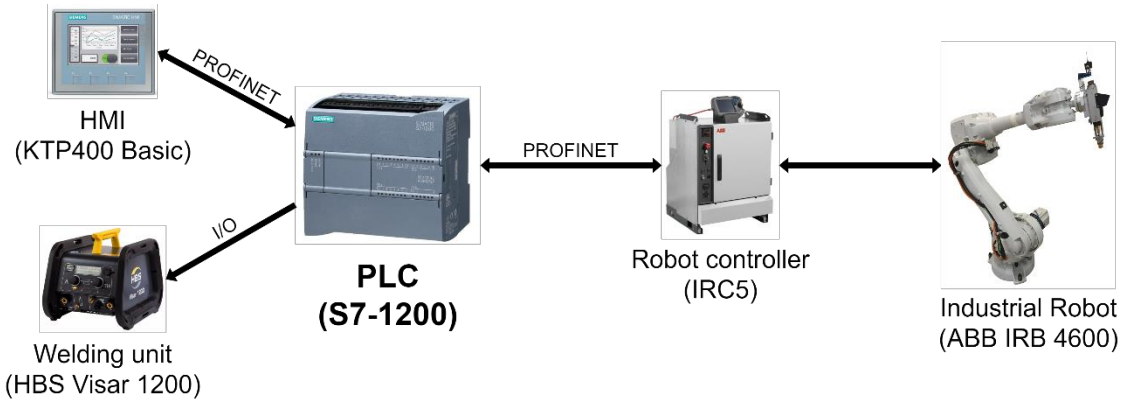


Figure 42: The chosen control structure. Images from [6], [7], [8] and [9].

4.3.2 Setting up the PROFINET communication system

Figure 43 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 to collect and deliver industry device data faster than alternative standards. To use PROFINET, an XML file coded in GSDML[17] (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.

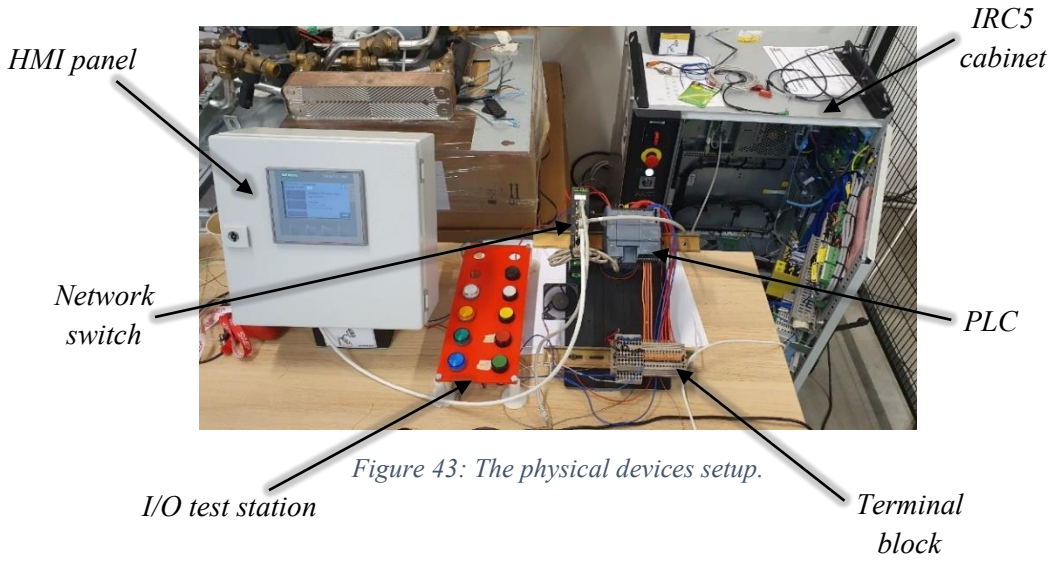


Figure 43: The physical devices setup.

4.3.3 Exporting weld parameters to the PLC

The welding parameters (amount of current, time duration, gas flow) of each weld in the finished automated solution must be exported to the PLC from the HBS welding unit 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 would be stopped until the HBS welding unit has been inspected and the values have returned to their allowed ranges. Unfortunately, the HBS Visar 1200 was found to not support data export from individual welds during operation. Instead, it only allowed exporting them all together to a connected USB drive. 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 separate sensors to directly 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 Activating the weld program from the PLC

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 digital signal "want_weld" through the PROFINET network and then wait for the PLC to activate the weld and then alert the IRC of its completion using another signal "weld_done", where the PLC 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 44, 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_DO8,1; !activate the HBS welding program
WaitTime(1.75); !wait for program to finished, and let bolt cool off
SetDO Local_IO_0_DO8,0; !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 44: RAPID code example, moving control of the weld power to the PLC.

4.3.5 Programming the PLC to control a full production 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 welding 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. Appendix B presents the finished state diagram of

the PLC program. 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 used between the PLC and IRC.

Signal name	Signal type	Description
irc_is_online	Digital input	Notifies the status of the manual mode safeguard.
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 45. 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 46.

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".

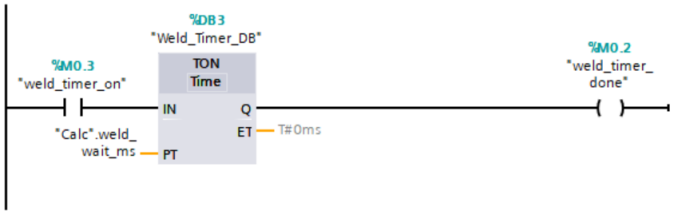


Figure 45: LD code used to generate a delay for cool-off after each weld.

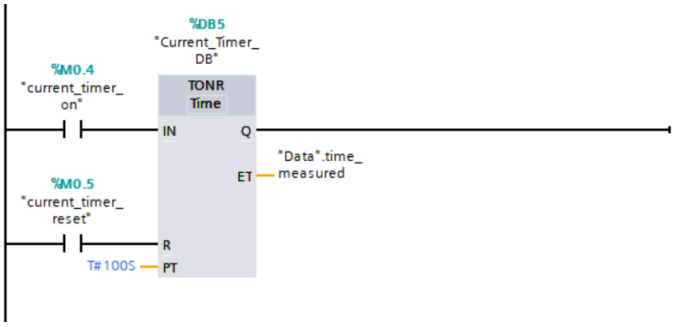


Figure 46: LD code used to measure the weld current duration.

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 47. For a Siemens PLC, the maximum value of an analog input is 27648, given by the maximum allowed input of 10 V. 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.

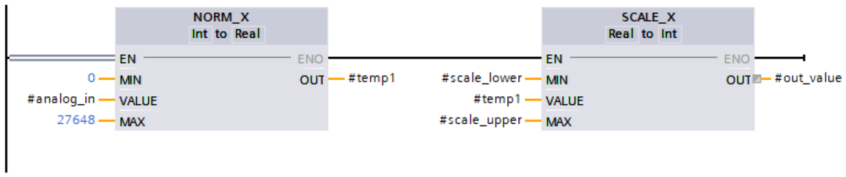


Figure 47: LD code used to measure the sensors analog inputs.

4.4 Enabling the safety system for the welding unit

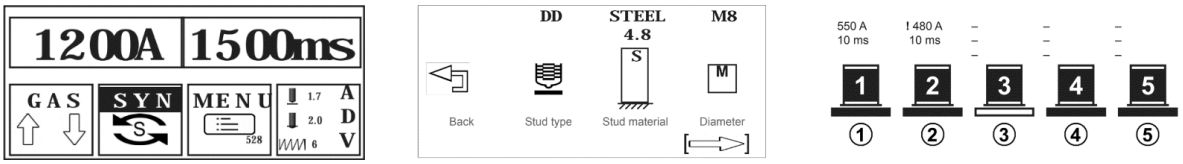
4.4.1 Studying the Visar 1200 manual

As the Visar 1200 model had never been used before in production at SWEP, it was not fully known how to create new welding programs, define tolerances, or export them 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 as seen in Figure 48, 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 48 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.



Activating the SYN mode | Selecting properties | Performing reference welds

Figure 48: 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, the sensors were ordered by SWEP but did not arrive during the thesis work. 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 of 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

4.5.1 Defining the HMI requirements

After obtaining 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 work order.
- Cycle-stopping the program execution at any time, finishing the active cycle before stopping. For example, finishing an ongoing weld before stopping.

- 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 to provide situational awareness to the operator. For example, it would 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 from 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 49 of an early version of the final HMI program, it becomes easily understood what actions are allowed in the moment:

- 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.

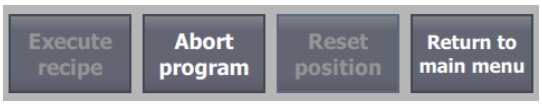


Figure 49: Designing an HMI: lead the way by only highlighting allowed actions.

4.5.3 Programming the HMI

Many of the more straightforward or standard features often needed in an HMI can be implemented directly in TIA Portal without programming. To show an example within the thesis work, consider the simulated HMI view as seen on the left side in Figure 50, 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.

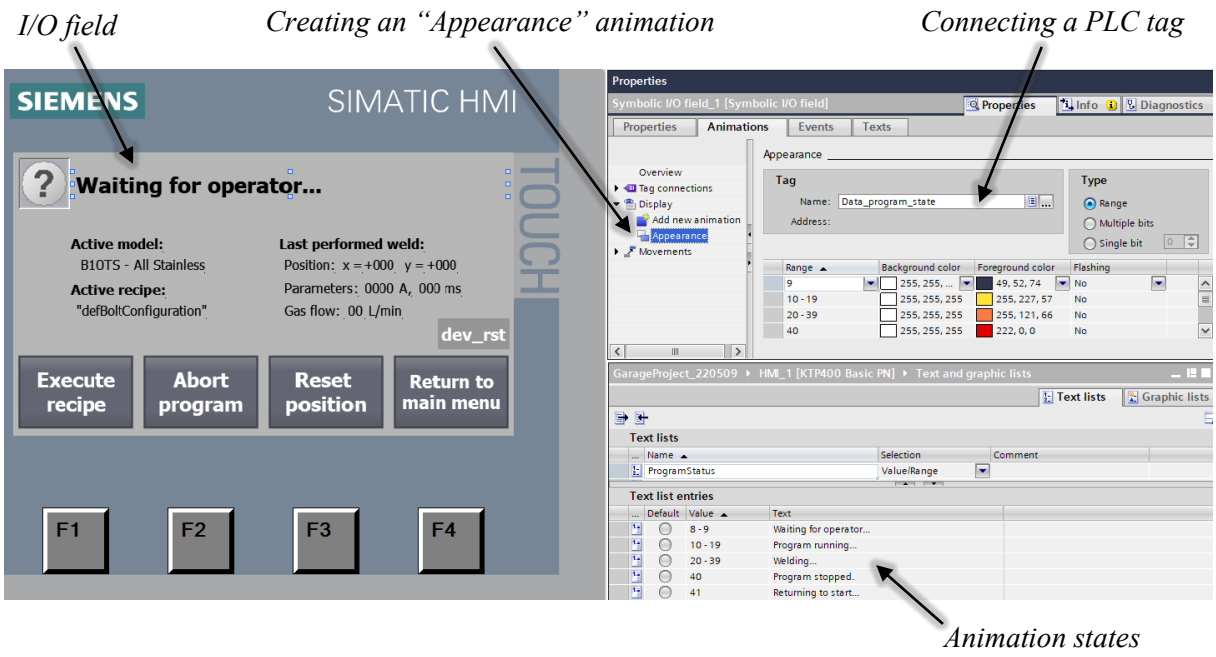


Figure 50: Programming example of an early version of the final HMI design.

4.5.4 Additions made to the pre-existing 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 subsection 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 51.

```
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
    ENDIF

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

ENDIF
```

Figure 51: 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 52.

```
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 52: 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 then wait 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 53.

```
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; !Syncronize 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 53: RAPID code example: importing a recipe.

4.6 Improving the pre-existing RAPID program

4.6.1 The possibility of welding bolts horizontally instead of vertically

So far, it has only been considered to mount a heat exchanger vertically when bolt welding as rotating heavy heat exchangers is difficult for operators. It is also difficult for robots as the gripper tool used to pick up robots by their cover plate can not survive the large torque resulting in holding the heat exchanger in any other way than pointing down towards the floor. Eliminating the need to flip a heat exchanger by 180 degrees between the P- and F-side in the middle of the robot program would save cycle time. However, it still has to be rotated by 90 degrees at the start to be mounted horizontally, but this could be done mechanically and more manageable than a complete 180 degrees turn. Nonetheless, it is worth analyzing if horizontal welding is even plausible. It would then be possible for the robot to finish the whole welding program with no rotation of the heat exchanger, even opening up the possibility of having one robot welding the P-side while another robot welds the F-side, which would tremendously decrease the cycle time. For clarity, Figure 54 and Figure 55 shows what is meant by vertical and horizontal welding.

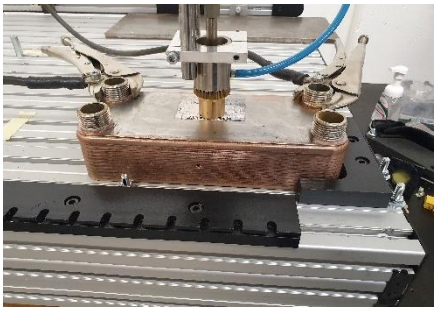


Figure 54: Clarification of "Vertical welding" – weld gun operates vertically

Figure 55: Clarification of "Horizontal welding" – weld gun operates horizontally

4.6.2 Designing a fixture for horizontal mounting of a heat exchanger

In vertical welding, the heat exchanger was placed only on the F- or P-side, creating flat and perpendicular contact with the workstation table. If flipped 90 degrees and placed on its side, it would not be a perpendicular nor mechanically stable mount as the cover plate sticks out a few mm around the channel plate stack. Additionally, for the robot to measure points on the edge closest to the table, the heat exchanger must be held some distance above the table to allow the probe to enter underneath it to reach all cover plate edges. Therefore, a completely new design of fixture for mounting the heat exchanger must be used. In designing this new fixture, the following requirements were defined. Two concept designs of a fixture that hold up to these requirements were designed, and are presented in section 5.4:

- The fixture must position and lock the HEX firmly in place to not lose positional tolerance between welds from the strong Lorentz forces from the large weld current.
- The fixture should not use the connection holes as grip as they would get damaged and vary greatly in size and looks.
- The HEX should, when placed in the fixture, have one side position roughly known so that the other side's rough locations can be quickly approached using data of the heat exchanger dimensions before more slowly and accurately searching with the touch probe.
- The fixture should be able to handle various HEX sizes while still allowing the touch probe to be able to reach all touch points on the P- and F-side simultaneously, as defined in Figure 22.
- The fixture should have a minimal number of moving parts, and it should be possible to control electronically.

4.6.3 Testing and programming

The pre-existing RAPID program for vertical welding was used as a foundation along with its math functions for calculating planes and intersections. Most of the code was remade to operate horizontally on the now two working areas simultaneously, the P-side and F-side. The work included creating new target points and orientations for the new movement paths, finding new axis configurations far away from joint singularities, programming changes, etc., while still being compatible to be run by the PLC and HMI program created.

A new complication in horizontal welding of SWEPs heat exchangers arose quickly, namely handling the different-looking F-sides across some product models, as seen in Figure 56. On the P-side, the cover plate sides are measured on the cover plate edge, a set distance below the cover plate plane. Now, on the F-side, the height of the heat exchanger had to be considered in the program to accurately

hit the smooth part of this F-side rim, as well as welding bolts on the surface that now exists at a distance less than the full height of the heat exchanger from the P-side.

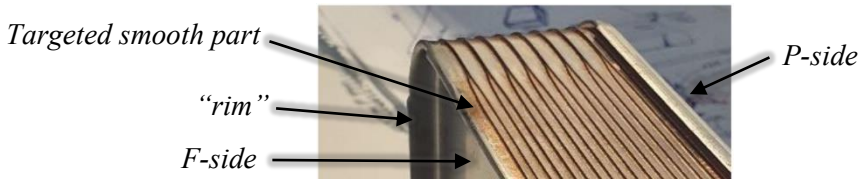


Figure 56: F-side rim, compared to P-side plate.

The RAPID program was changed and built upon, and a full robot program where the robot first welds on the P-side and then the F-side without moving the heat exchanger was successfully created. The improved automated solution is presented in section 5.4.

4.6.4 The effect of gravity in horizontal welding

A shielding gas flow is used during a weld to remove oxygen from the gun chamber to avoid oxidation marks around the bolts after a weld. It was unknown how the gas flow would affect the welding quality for the now asymmetric effect from gravity when welding horizontally. To analyze this, bolt welds were performed under the same circumstances with only them being vertical or horizontal as the only difference, to visually see if gravity would affect the weld collar quality. Images of these welds and results from this analysis are presented in section 5.4.3.

4.7 Investigate using the weld gun with multiple bolt lengths

4.7.1 Understanding the inner workings of the KAH 412 LA welding gun

Early on in the thesis work, it was noticed that using 20mm bolts sometimes resulted in very weak welds, while 15mm bolts were no problem. In the first period of the thesis work, it was not exactly known how to configure the weld gun for different bolt lengths and if any limitations existed. The technical manual of the weld gun was studied and a few meetings with SWEPs provider of the gun, BM Svets, were held to get some clarifications on how the weld gun works.

An explanation of how to weld gun work will now follow. Figure 57 shows the welding gun and what will be mentioned in this section as its static "nozzle" and spring-loaded "piston". After a bolt has been inserted in the piston, the weld gun nozzle head where the bolt is located should be placed in direct contact with the welding surface. This will push the bolt and piston upwards, acting against an internal spring. That is, the piston will be pushed inwards the same length as the length of the bolt sticking out of the nozzle head. When the weld program is triggered by the welding unit, the piston is rapidly pushed upwards an additional distance, called the "plunging depth", and then immediately released at high speed, hitting the bolt on the welding surface, where the weld current flow is triggered, melting part of the bolt and welding it to the surface as it solidifies. The plunging depth can be manually chosen by turning the grip in Figure 58, where the minimal depth possible is 0.5mm, achieved by turning it counterclockwise until it stops turning. Then, turning it clockwise, the plunging depth increases by 0.1mm per felt click. There is no label or indicator of the active value, so turning it all the way to 0.5mm and counting the felt clicks is necessary. In KAH 412 LA, the piston can only be pushed up against the spring for a total of around 9mm, and BM Svets recommends a plunging depth of 1.5mm for the short-cycle welding done at SWEP. This means that the part of the bolt that sticks out of the nozzle head may not be longer than 7.5mm. The bolt holder in the piston can contain 12mm of the bolt, and the rest will

stick out. If the bolt is shorter than 12mm, there can be no surface contact, so 12mm is the minimal bolt length possible. So, the maximum bolt length that can be used with a 1.5mm plunging depth is 19.5mm. This explains why 20mm bolts only occasionally resulted in bad welds, and 15mm bolts always resulted in great welds. The 20mm bolts only had 0.5mm of room for plunging, which only sometimes worked.

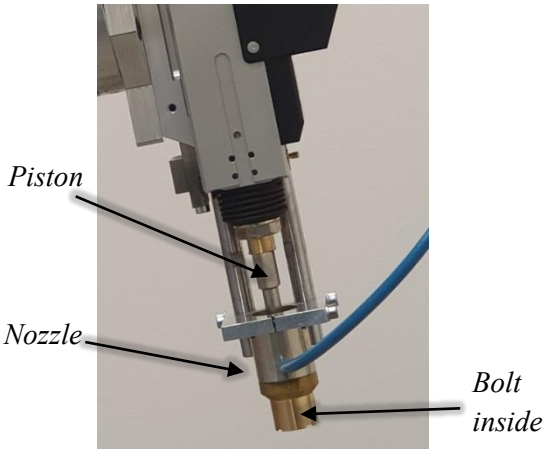


Figure 57: The welding gun - piston and nozzle.

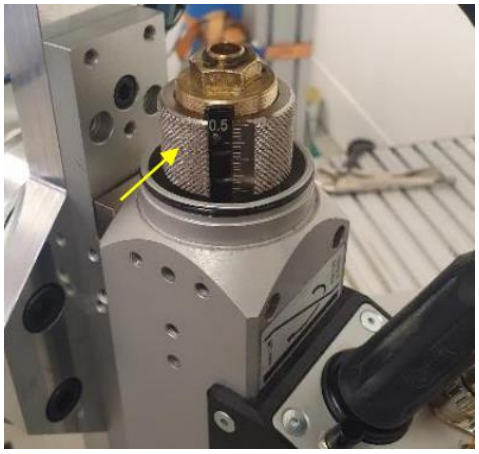


Figure 58: The welding gun - changing plunging depth.

4.7.2 Limitations in using multiple bolt lengths in the same weld gun

In the context described in 4.7.1, the welding gun only allowed bolt lengths between a minimum of 12mm and a maximum of 19.5mm. To move this 7.5mm interval, the entire nozzle can be loosened, positioned at a new distance, and fastened again. If the nozzle is moved down 2mm, the new range of allowed bolt lengths would be 14mm to 21.5mm. How these results affect the use of the welding gun at SWEP is presented in section 5.5.

4.8 Simulating the cell in factory-wide production flow

This chapter describes the work of creating a simulation of the future state of the automated stud bolt robot cell in Plant Simulation. This was done in the second thesis period, so the project of automating the stud bolt process had been greatly built upon, and a completely new design and simulation of the cell had been created in RobotStudio by SWEP, presented in subsection 4.8.3.

4.8.1 Multiple welding guns and bolt configuration frequencies

There are four lengths of M6 bolts and seven lengths of M8 bolts being used in the stud bolting at SWEP. Figure 59 shows the frequency of use between these different bolts across two recent years. Five bolt types stand out as much more frequent than the rest: M6_10, M6_20, M8_20, M8_25, M8_30, that, if summed up, account for around 92% of all bolt welds. For the welding gun described in section 1.3.3, a mounting head must be chosen to match the bolt diameter and will also only allow bolt lengths within a chosen range of roughly 7mm. Therefore, SWEP decided to use four welding guns, which could then be configurated as, for example:

M6 mounting head, calibrated to allow lengths of 7-14mm	(M6_10 covered)
M6 mounting head, calibrated to allow lengths of 17-24mm	(M6_20 covered)
M8 mounting head, calibrated to allow lengths of 17-24mm	(M8_20 covered)
M8 mounting head, calibrated to allow lengths of 24-31mm	(M8_25, M8_30 covered)

This will then handle the 92% majority of all bolt welds. SWEP decided that the ones that are not covered in this majority of configurations are to be welded manually in another station. Figure 60 shows how often bolts are needed on only one side of the heat exchanger, as well as how often bolts are needed on both P- and F-side. Around 80% of welding needs only a single side. Around 20% need both sides, which means the heat exchanger must be flipped once. In the robot cell, the robot that picks up heat exchangers from the pallets can only place them in the turners on the P-side. So, some configurations of those 80% of single-side welds actually need to be flipped as well, namely the ones that needs bolts on the F-side alone. No precise data existed of frequency between "P-only" and "F-only" separately, so only the "P-or-F" frequency in Figure 60 was initially of use.

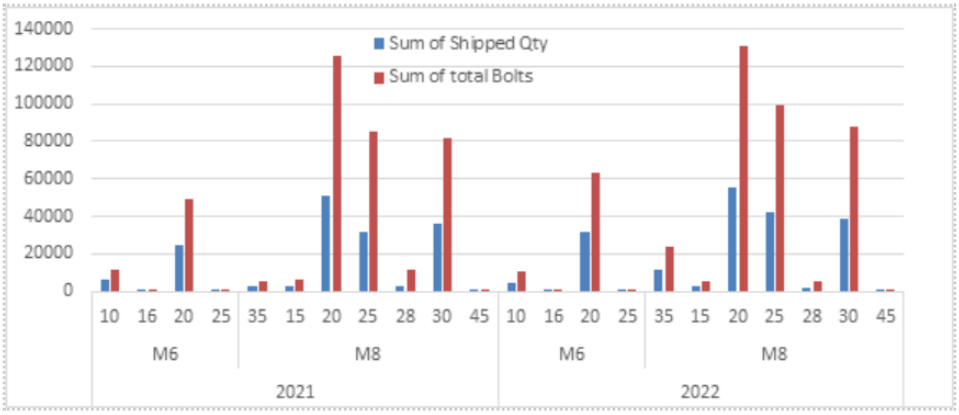


Figure 59: Bolt order frequencies at SWEP. Image from [22].

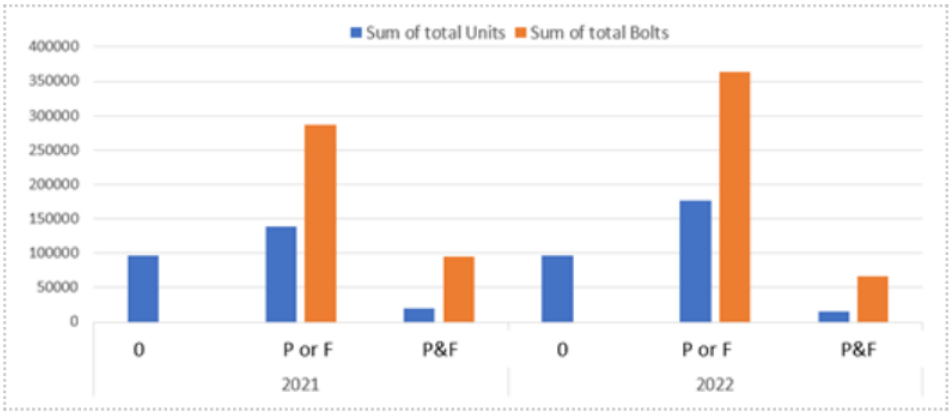


Figure 60: P/F side frequencies of bolt orders at SWEP. Image from [21].

4.8.2 Finding bolt configuration data and relative frequency

To compare the output of a manual welding station with the new stud bolt robot cell, simulating a realistic relative frequency of stud bolt configurations is crucial, as the cycle time of the two depends very differently on the number of bolts needed on each side. As no data existed that showed the relative frequency of different bolt configurations, for example showing the percentage of heat exchangers produced that need exactly a "2P+2F" recipe, a "2P+0F" recipe, etc., an estimation of this relative frequency had to be found so that representative bolt configurations could be added to the main "job list" in SWEPs factory-wide model in Plant Simulation, a large matrix of roughly 2000 rows each

containing information about an order, like the number of heat exchangers, production codes, sale order, time stamps, etc., but nothing about the number of bolts had been entered and simulated yet.

The large Excel file that the job list was a subsection from did include the bolt configuration but not in a way that made it easily exported across thousands of rows. So, the first 500 rows were manually examined, summing up the quantity of heat exchangers of the most common bolt configurations. Table 2 shows the relative frequency of bolt configurations for these 500 production orders, rounded to the nearest percentage integer. This data was crucial when deciding what to include in the simulation and what parts could be left out without affecting the cycle time of the cell, further explained in section 4.8.4. Apart from each specific configuration frequency, it shows that roughly 88% (40+33+13+2) of heat exchangers needs bolting on only a single side, and that 64% (40+13+5+3+3) of heat exchangers need bolting on the F-side and thus need to be flipped in one of the two turning stations.

Table 2: Bolt configuration relative frequency.

Bolt configuration	Total heat exchangers	Configuration frequency, 0P + 0F included	Configuration frequency, 0P + 0F excluded
0P + 0F	3616	40 %	-
0P + 2F	2165	24 %	40%
2P + 0F	1859	20 %	33%
0P + 4F	725	8 %	13%
2P + 2F	257	3 %	5%
4P + 4F	202	2 %	3%
4P + 2F	144	2 %	3%
4P + 0F	57	1 %	2%
2P + 4F	40	0 %	0%

4.8.3 SWEPs current stage of automating the stud bolt robot cell

After finishing the first period of the thesis work, as mentioned being 15 weeks long, other automation engineers at SWEP took over the work where it was left off. Almost two years later, when continuing the thesis work, a robot cell much closer to full automation had been simulated and programmed in RobotStudio, and the real cell had been constructed at SWEP. Gladly, several findings and results from the first part of the thesis work seem to have affected the final design. From the first part of the thesis work it was shown that horizontal welding was possible, but that gravity had a significant effect on the weld quality, as described in section 4.6.4. Even though vertical welding needs a second system that flips the heat exchanger between P- and F-side, SWEP finally decided to use a vertical welding solution. Also, four separate welding guns are used to cover a large range of bolt lengths.

Figure 61 shows a simple schematic of the final design of the automated stud bolt welding cell designed by SWEP. Pallets with heat exchangers enter and exit the cell at P1 and P2 by SWEPs AGVs. A robot, R1, picks up heat exchangers from both pallets and places them in two turning stations, T1 and T2, where each heat exchanger can be automatically rotated between P- and F-side. R1 prioritizes work on the oldest pallet that entered. When a P- or F-side needs bolting, the other robot, R2, uses an attached measuring probe like the one presented in 1.3.3 to precisely measure the position and rotation of the heat exchanger. It then picks up one of four welding guns, WG, and welds stud bolts as needed on the heat exchanger. If F-side bolts are needed as well, the turner will rotate the heat exchanger and R2 can again measure and then welds bolts to it. Bolts are poured by operators in bulk to bolt dispensers, B, from where they are automatically fed to each welding gun through a tube. When any of the two heat

exchangers are done, R1 places it back to the same spot on the pallet it came from. Only two widths of heat exchangers will be automatically welded, 100mm and 120mm, and R1 needs to have equipped the corresponding gripper for the width it works on. When using the 100mm gripper, the 120mm is kept at GR. If R1 needs to switch between the two grippers, it can drop off and pick up the other gripper at GR.

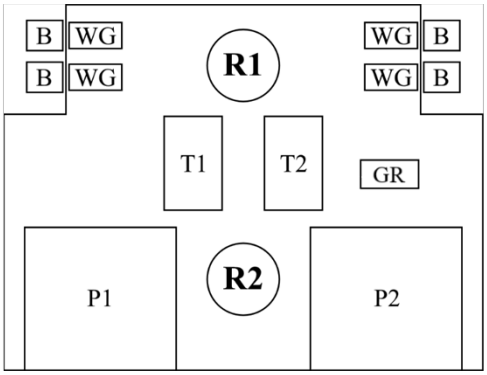


Figure 61: SWEPs current stage of automating the stud bolt robot cell – cell schematic.

4.8.4 Choosing the level of abstraction

When developing a model of the cell in Plant Simulation, it is desired to include as much detail as possible, but the level of detail, and amount of abstraction, must be defined. For the stud bolting cell, it is not worth the resources to simulate a very rare occurring event in the cell if it insignificantly changes average cycle times and factory output over the course of a whole week of production. For well over 90% of pallets entering the cell, all heat exchangers will have the exact same bolt configuration, and only one welding gun will need to be used for each pallet. For well over 90% of heat exchangers, only one bolt type (for example, M6_10) is needed. Based on this, along with the data and findings presented in section 4.8.1, the following level of abstraction was chosen for the module in Plant Simulation:

- Each heat exchanger on the same pallet is considered to have its own bolt configuration recipe but it will only consist of the number of P-bolts and number of F-bolts, not positional data.

- Only one of the four welding guns needs to be simulated, as every new P- or F-side facing robot 2 will need to be measured again by the measuring probe, then place the measuring probe in a holder, to then again pick up a welding gun. Therefore, the cycle time is not dependent on the bolt type (and with that, which weld gun to pick up) as a gun will have to be placed in the holder and picked up again regardless.

- A significant amount of heat exchangers needs to be rotated from P-to F-side and then again back to P-side before robot 1 can place it back to its pallet position, so, the turning mechanism must be simulated as it will definitely affect the cycle time on these configurations.

- Switching between the 100mm and 120mm grippers used by robot 1 takes roughly 10 seconds, which would have a relatively large effect on cycle time. However, only a single heat exchanger model at SWEP (22AS) uses the 100mm gripper width and is manufactured in relatively small quantity. In regard to that reason, it is not worth simulating. But, it will be done anyway with the motivation of creating a better visual demonstration of how the cell operates, which is something SWEP values to have in their Plant Simulation models, as well as making the simulation ready for a possible future where the 100mm gripper is used more frequently.

4.8.5 Building and programming the module in Plant Simulation

A simulation of the automated robot cell was built in Plant Simulation from scratch using the Plant Simulation blocks and programming tools described in 3.3.2, to simulate the cell as close as possible to the real process presented in 4.8.3, with the abstraction and limitations chosen in 4.8.4. Setting up the simulation blocks needed took a relatively short amount of time. Programming all necessary behavior of the two robots, their tool usage, the pallet delivery, is where most of the work was. The finished simulation module is presented in subsection 5.6.1. Parts of the SimTalk code is presented in Appendix D, to give an insight of the programming language and the programming task at hand. But, it mainly exists to act as example code to help any Plant Simulation programmer reading this report, as there is a scarcity of these on the Internet relative to other more commonly used programming languages, most likely since SimTalk is a very software-specific language that requires a relatively expensive license to use.

4.8.6 Creating representative 3D models

The static robot cell hardware, one of four identical heat exchanger gripper fingers used by robot 1, the touch probe and welding gun used by robot 2 were all modeled from scratch in AutoCAD Inventor to create more light-weight models to give an accurate visual representation of the cell while the simulation is running, which is of importance to SWEP. A 3D model with simulated joints of the actual ABB robot used could not be found, so the same temporary robot model used in the rest of SWEPs factory-wide model in Plant Simulation was used, a 3D model of a similar KUKA robot. Figure 62 through Figure 65 shows the 3D models created in the thesis work.

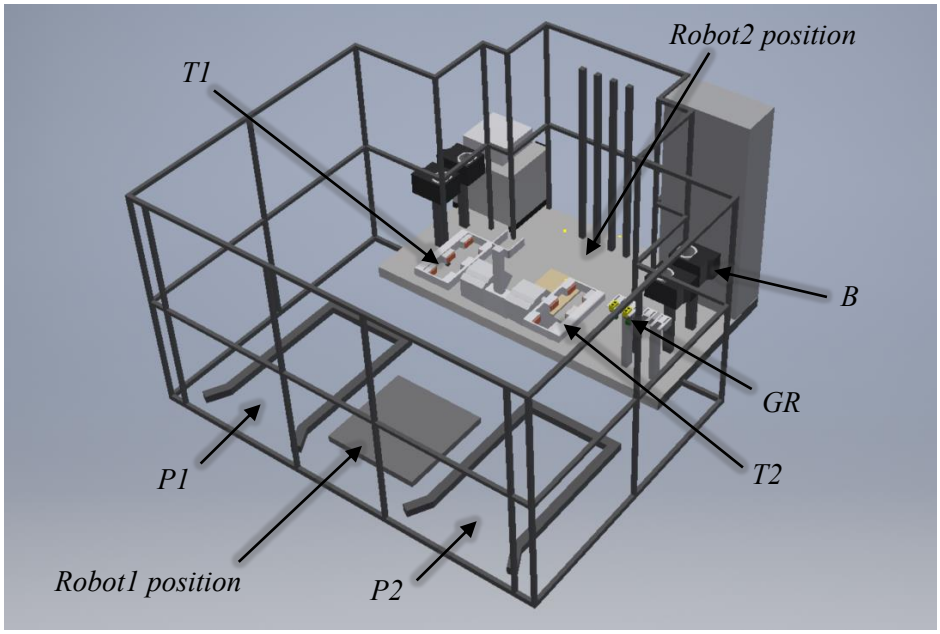


Figure 62: The finished 3D model – the new robot cell.

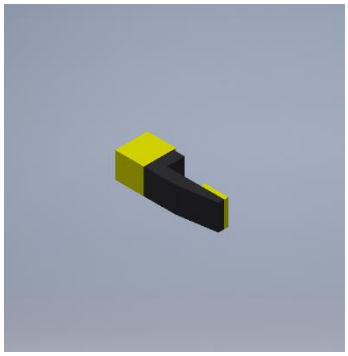


Figure 63: The finished 3D model – a gripper finger.

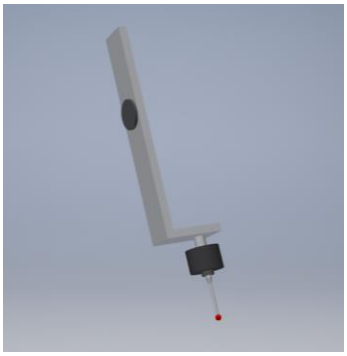


Figure 64: The finished 3D model – the touch probe.

Figure 65: The finished 3D model – touch probe and weld gun.

4.8.7 Comparing the output capacity between manual and automatic welding

After the robot cell module had been finished in Plant Simulation, its output capacity was simulated and compared to the output capacity of manual welding. Figure 66 shows the model built to compare them, with a copy of the previously existing module for manual welding at the top, and a copy of the newly created robot cell at the bottom. To the left, a Source object iterates between each production order in the job list and produces the number of heat exchangers in the order, until every heat exchanger in the entire job list has been processed by any of the two stations. A toggle switch is used to choose if the long sequence of heat exchangers goes to the manual station at the top or the robot cell at the bottom. To the right of each station, the total number of heat exchangers processed is shown during simulation next to the resulting total time to process the entire job list, as well as the resulting output as the average number of heat exchangers processed by hour. Using this model, a range of different specifications of OEE, operator shift time and cycle times were simulated for both stations. The results from these simulations are presented in section 5.6.2.

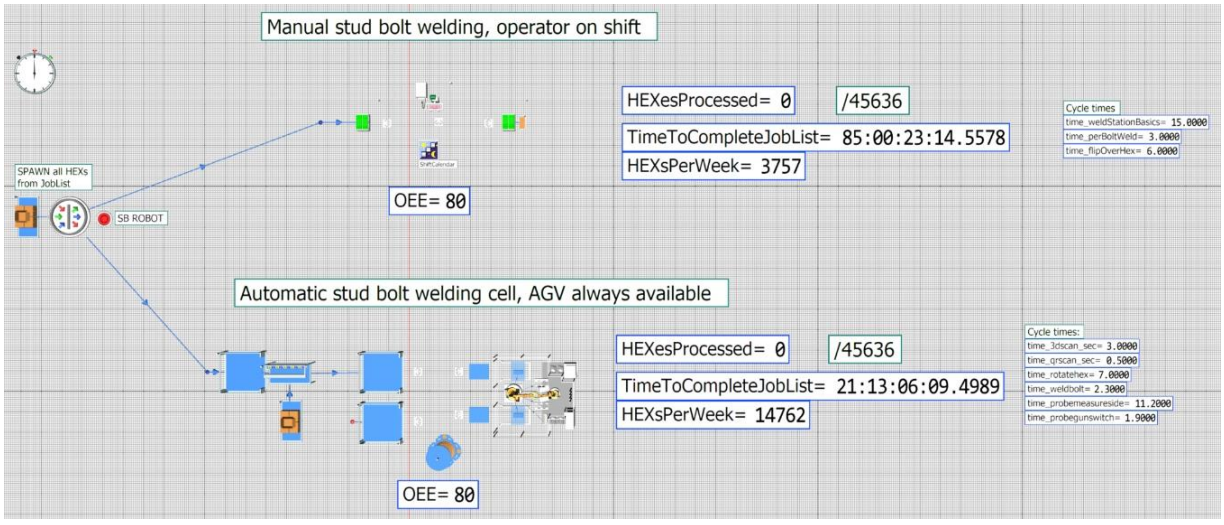


Figure 66: The Plant Simulation module created to compare output between manual and automatic welding.

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 Optimal weld parameters

The resulting bolt quality from the visual and physical tests is presented in Table 3, Table 4, and Table 5. 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 3, 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 weld 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 4, 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 3: 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 4: 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 5: Bolt quality results from the visual and physical tests, varying the gas flow.

Gas flow	Visual test result	Assessment
No gas	Large irregular collar. Lots 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 automated welding

5.2.1 Accuracy of the two methods for calculating the center point

Table 6 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 time 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 6 and Table 7.

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 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 measured by some

error distance “d”, it will have a significant effect on the calculated lines and the resulting intersection point, as seen illustrated in Figure 67 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 67 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 6: 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

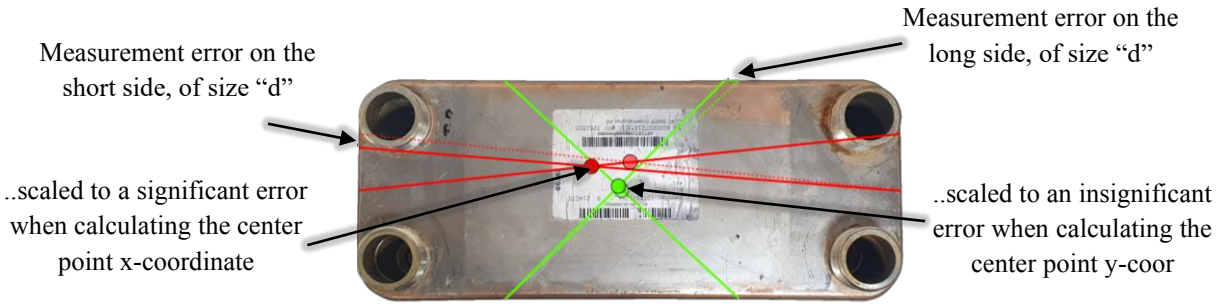


Figure 67: Graphical reasoning of why it is crucial to separate the short side measurement points as much as possible (exaggerated touch probe measurement error “d”).

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 7 shows the resulting positions, times taken, and the new resulting accuracy of the two methods.

It is seen that the measuring time has increased for both methods. This is 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 is seen to have been greatly improved, as seen in Table 7. The two methods are seen to be consistently within the tolerance of 1 mm. As most of the positions are within the second quadrant, another slight change in the offset might improve the accuracy even further.

Table 7: 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 PLC and HMI automated solution

To implement all features of the HMI decided in section 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 is 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. Figure 70 presents the "Root screen".

5.3.2 The "Operator view"

Figure 68 presents the "Operator view". 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.

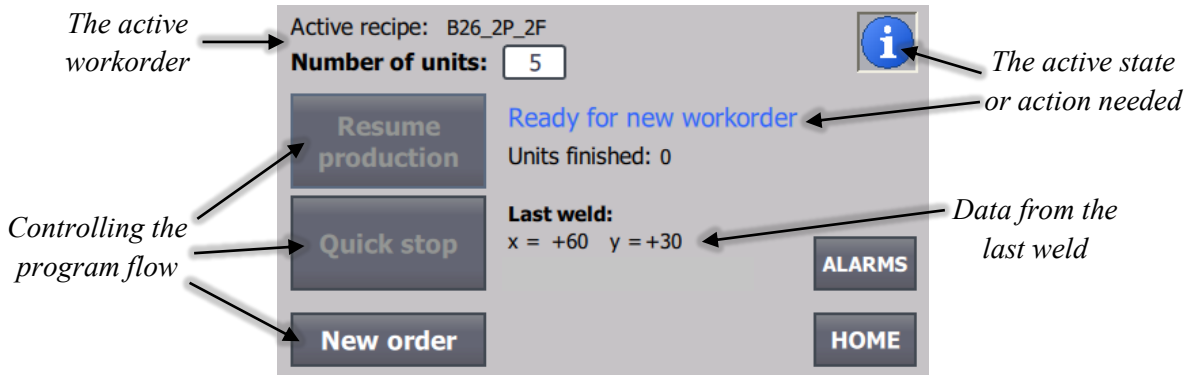


Figure 68: The final HMI - "Operator view".

5.3.3 The "Recipe view"

Figure 69 presents the “Recipe view”. 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.

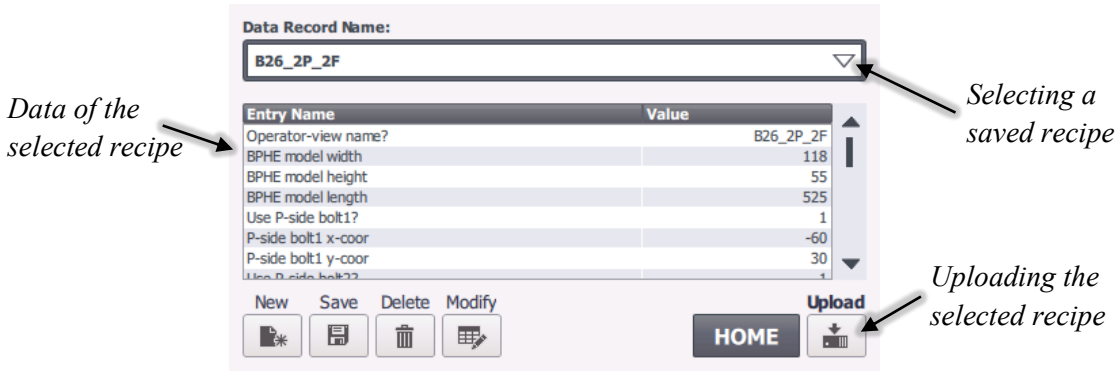


Figure 69: The final HMI - "Recipe view".

5.3.4 The "Settings" screen

In Figure 71, 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.

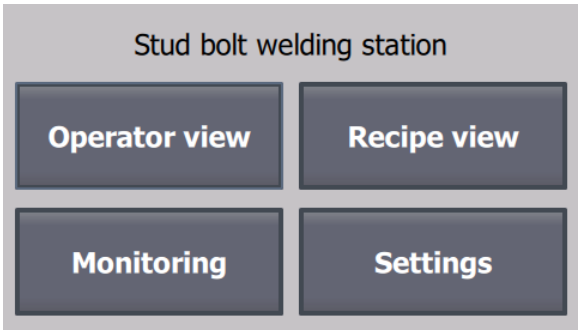


Figure 70: The final HMI - "Root screen".

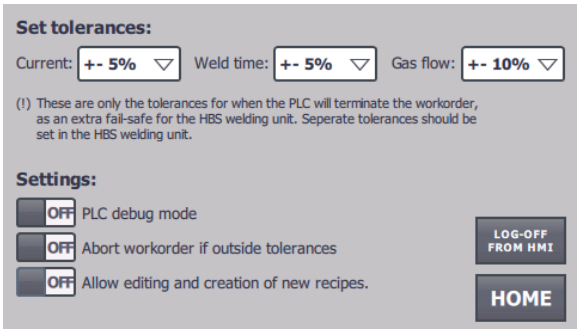


Figure 71: The final HMI - "Settings".

5.3.5 The "Monitoring" screen

In Figure 72, 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 73, 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 are outside the tolerances chosen in “Settings”.

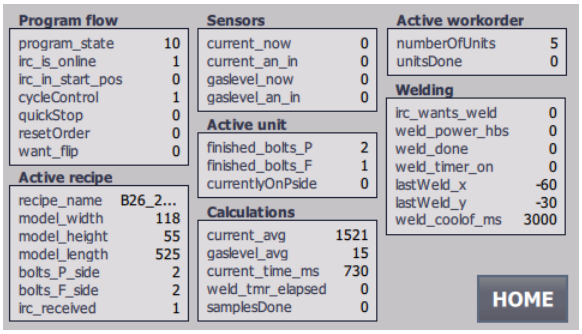


Figure 72: The final HMI - "Monitoring".

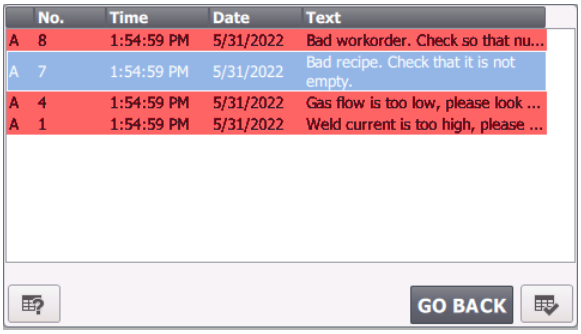


Figure 73: The final HMI - "Alarms".

5.4 Improving the pre-existing RAPID program

This chapter presents the result in creating an alternative algorithm for mounting and measuring the heat exchanger in the cell.

5.4.1 Horizontal fixture concept designs

Concept design A: Figure 74 shows the finished concept design. The heat exchanger would be placed horizontally on the P- and F-side “feet”, gripper tightly around the P-side cover plate and channel plate stack using four “grippers” and kept in place by a spring-loaded “front positioner”. This would be unaffected by most connections and still make it possible for the touch probe to access the short side points, as seen in Figure 75. The thickness of the P-side and F-side “foot” summed up must be smaller than the smallest heat exchanger height to be used in the cell, or alternatively create a limitation on the smallest heat exchanger height that is possible to use, as seen in Figure 76. In addition to the P-side “foot”, the F-side must be supported in order to not create a large torque on the P-side, solved by the F-side “foot” which then must be possible to be moved to accompany the height of the heat exchanger that is placed in the fixture. The P-side “front positioner” could perhaps be spring loaded so that the heat exchanger is held tightly, but still allowing the front positioner to be pulled back when removing the heat exchanger. This design should keep the heat exchanger firmly enough to not lose tolerance, still allowing the touch probe to reach all side measurement points. Figure 77 shows the fixture built for testing in the garage cell. Concept design A was presented to an engineer at SWEP and was imagined to work well, and that the amount of moving parts would not be an issue mechanically.

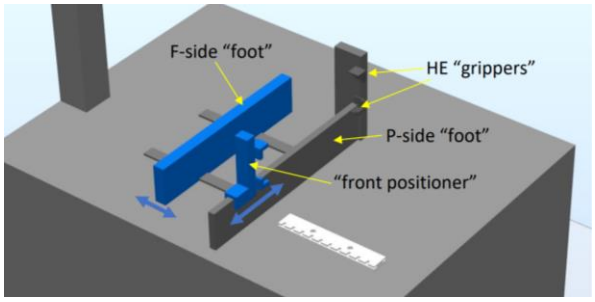


Figure 74: Fixture concept design A.

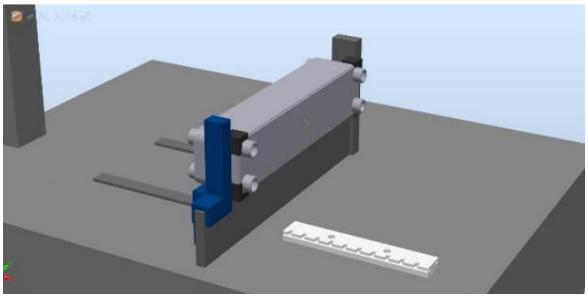


Figure 75: Fixture concept design A - HEX grip.

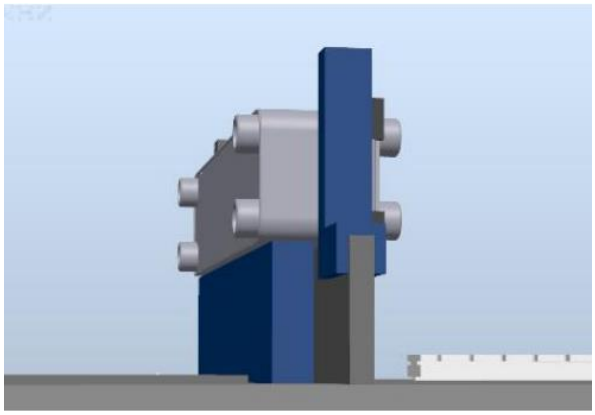


Figure 76: Fixture concept design A – height limitation.

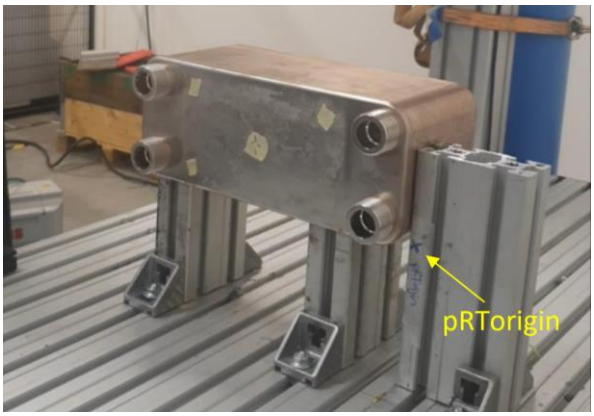


Figure 77: Fixture concept design A – garage test rig.

Concept design B: the same principle as in design A, but with the incentive of completely removing the need for moving mechanical parts. Instead, each type of heat exchanger size uses its own "model frame", as seen in Figure 78. It would require the robot to access and place these model frames from a nearby storage, which might be difficult and space-consuming. It is likely a bad solution in practice but a concept worthy to be considered. In a way, each model frame is too a moving mechanical part.

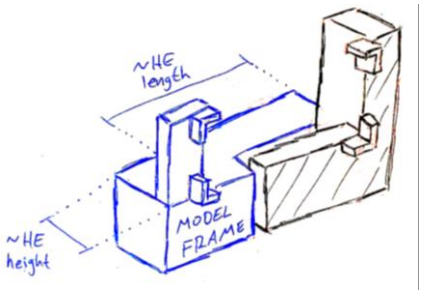


Figure 78: Fixture concept design B.

5.4.2 The improved automated solution

The new RAPID program created for horizontal welding with the new fixture design was tested and developed in the garage cell. The HMI program was edited to also support a horizontal configuration to not ask the operator for a manual flip between P- and F-side as the pre-existing program required. Figure 79 shows a series of snapshots in the RobotStudio simulation. The heat exchanger is mounted in the fixture (A). The robot then measures the P-side cover plate in the three points as seen in Figure 22 (B), then the five side points, even the more difficult locations (C and D). A bolt is then picked up (E) and welded (F). The same procedure is then performed on the F-side (G, H). When done, the heat

exchanger can be removed from the fixture (I). Figure 80 shows corresponding snapshots from the real robot in the garage. The robot program worked as intended and could be controlled by the PLC and HMI created in the thesis work. Conclusions drawn on whether horizontal welding in this way is a good approach are presented in section 6.4.

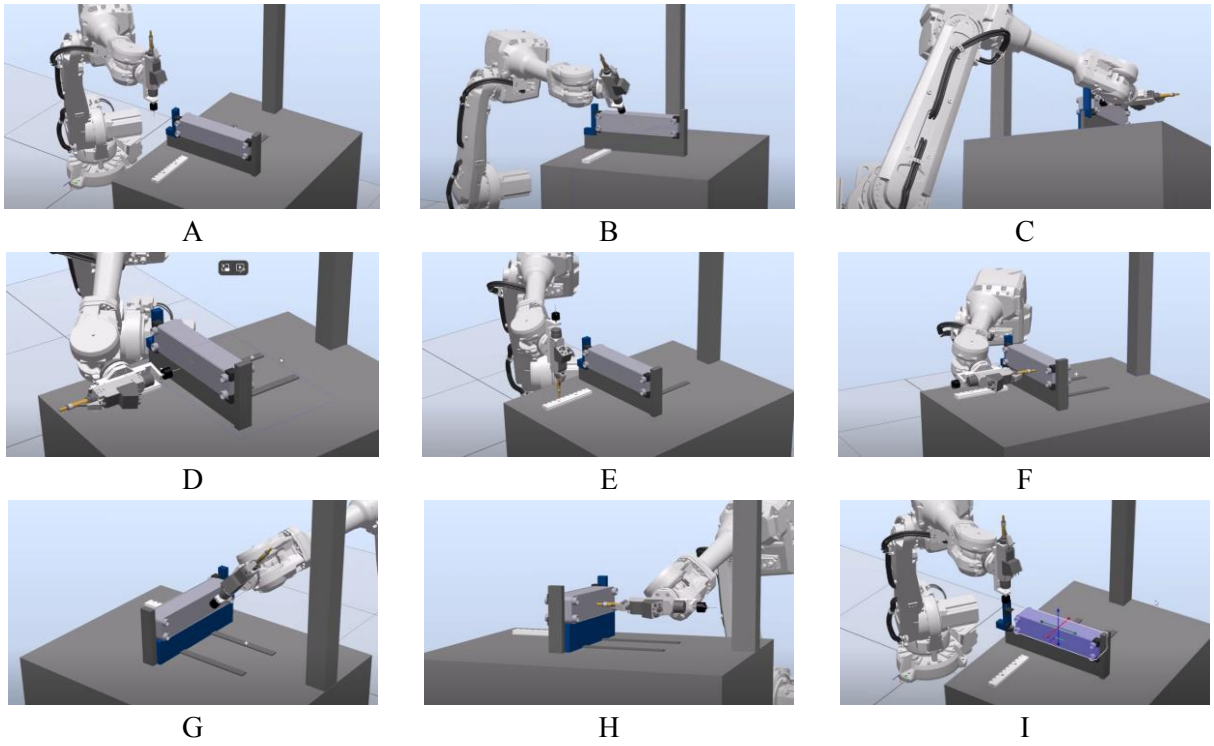


Figure 79: the improved RAPID program for horizontal welding – simulation.

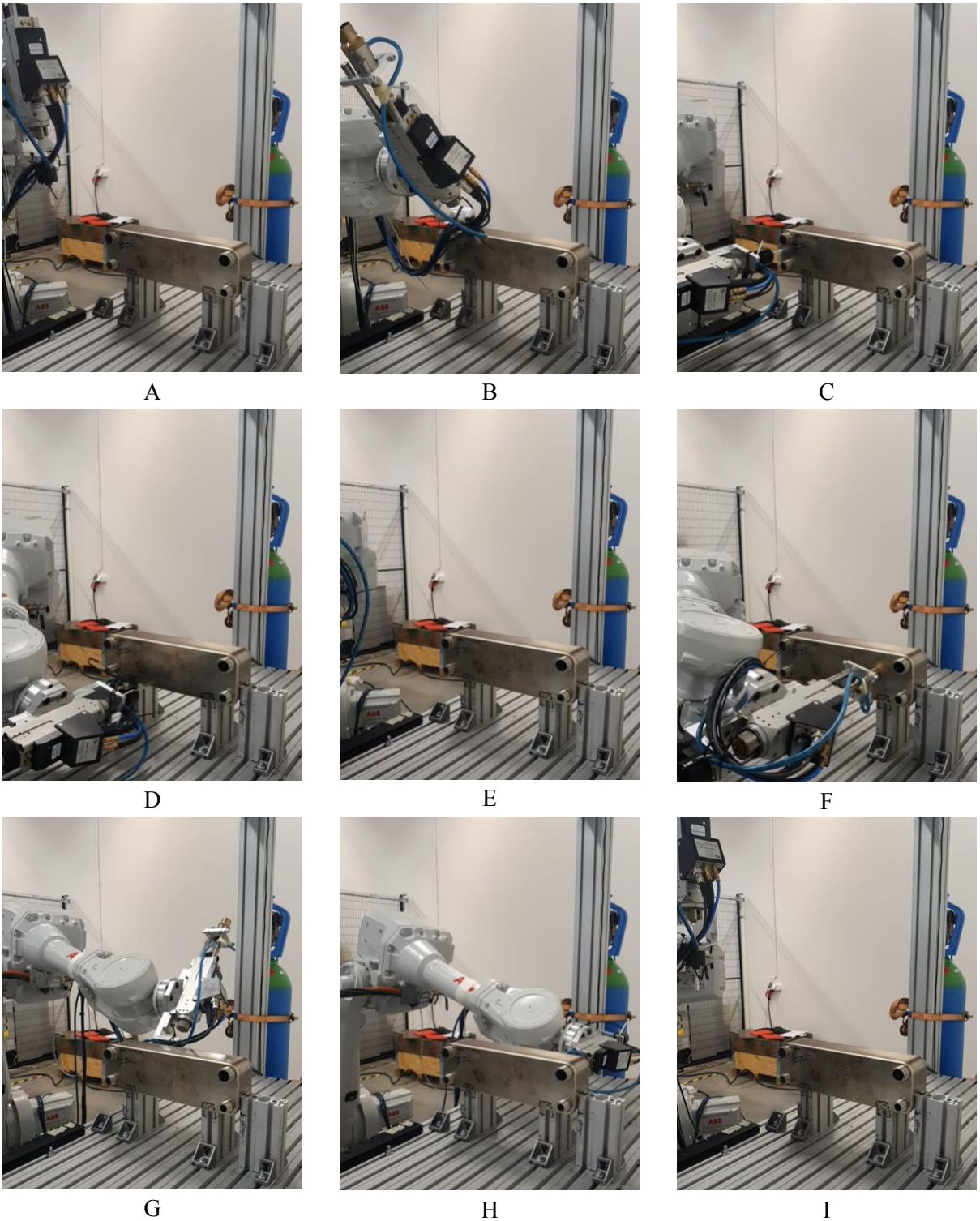


Figure 80: the improved RAPID program for horizontal welding – garage cell.

5.4.3 The effect of gravity in horizontal stud bolt welding

Figure 81 shows the resulting welds from vertical welding, from the analysis on how gravity affects the weld quality. Figure 82 shows the welds when done horizontally. Vertical welding is seen to result in symmetrical weld collars, that is, generally even quality around the bolt. When welding horizontally, having the effect of gravity pointing to the left in Figure 82, surprisingly, a major effect from gravity is seen. After a weld, the weld material is liquid for some time before it quickly hardens, and is seen to have had enough time to flow in the direction of gravity, creating a small but apparent "bulge" or weld material on one side, consistently through many of the horizontal welds.

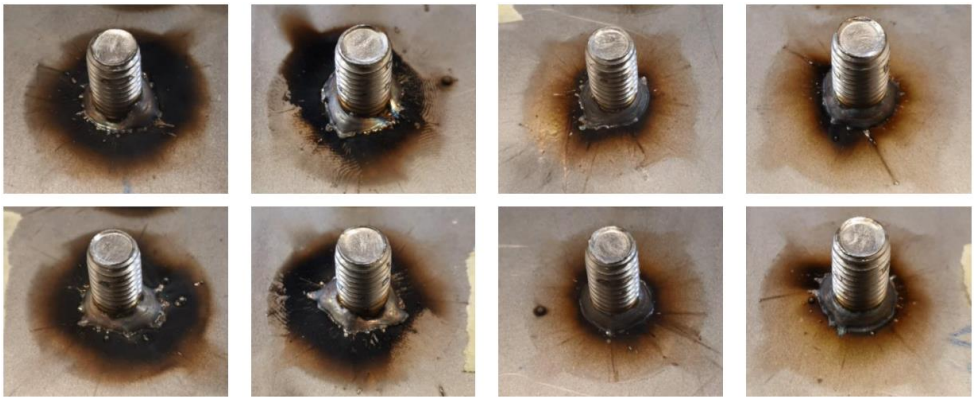


Figure 81: Analysing the effect of gravity - vertical welding.

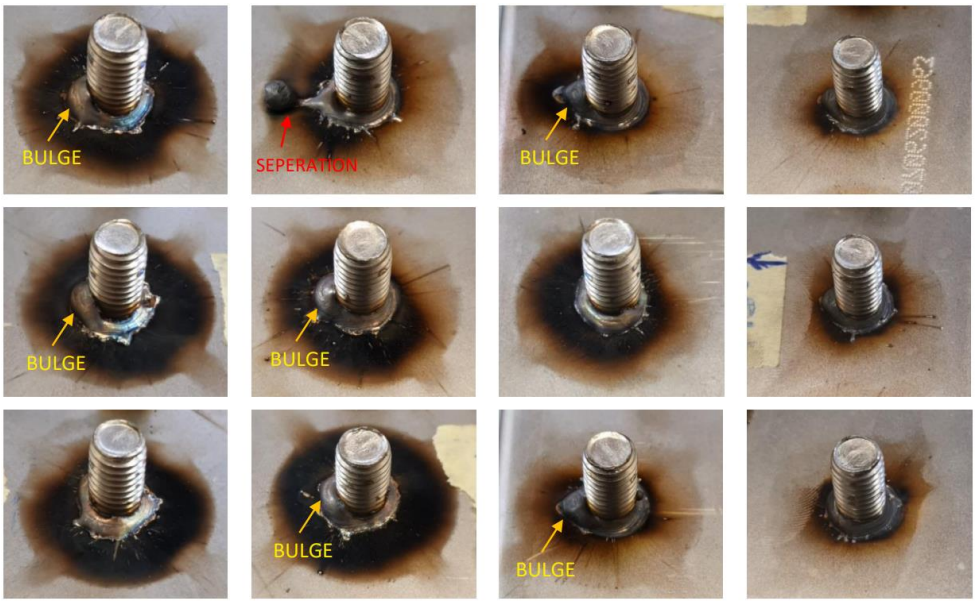


Figure 82: Analyzing the effect of gravity - horizontal welding (gravity pointing to the left in each image).

5.5 Using the weld gun with multiple bolt lengths

Following the analysis presented in section 4.7, the range of allowed bolt lengths of the KAH 412 LA welding gun is always 7.5mm in magnitude. At SWEP, the bolt lengths used in the welding station span around 10mm to 40mm, so it is not possible to configure a single KAH 412 LA welding gun to cover this full range. Instead, multiple welding guns would be needed, for example covering the ranges of 10mm to 17.5mm, 17.5mm to 25mm, 25mm to 32.5mm, and lastly 32.5mm to 40mm in four separate

welding guns if the full 10mm to 40mm range must be covered. Using a single welding gun and performing the mechanical tuning for each new length is possible but would likely cause the nozzle screws to break after some time. For each re-positioning of the nozzle, the length of the tool changes, so the tool data in RobotStudio must be updated with the new tool length.

5.6 Simulating the cell in factory-wide production flow

This section presents the results obtained in the work of creating a simulation model in Plant Simulation of the stud bolt robot cell. This includes the module built and programmed, as well as the simulation results.

5.6.1 The finished Plant Simulation module

A top-down view of the finished Plant Simulation module of the robot cell is shown in Figure 83, showing an AGV delivering a pallet of heat exchangers (colored green for clarity) to the robot cell, all surrounded by various graphical programming elements and data variables. The module may receive HEXs from the factory-wide model, or be chosen to be run independently, spawning the same HEXs locally, chosen by using the toggle buttons as seen in Figure 84. A close-up view of the robot cell during simulation is shown in Figure 85, showing robot 1 placing a heat exchanger from a pallet into one of the turning stations, as robot 2 welds bolts on the F-side on the heat exchanger on the other turner. Simulating the module in real-time speed, it closely resembled the real robot cell as described in subsection 4.8.3 and the chosen level of detail and abstraction as described in subsection 4.8.4 was accomplished, and so deemed a very satisfactory result. The module was easily integrated in to SWEPs factory-wide model as a Frame using the Interface objects, replacing some of the existing manual welding stations. In that model, only the actual robot cell graphics is shown. The rest is only accessed when entering the module Frame, then seen as in Figure 83. For integrity purposes, a picture of this factory-wide model could not be shown.

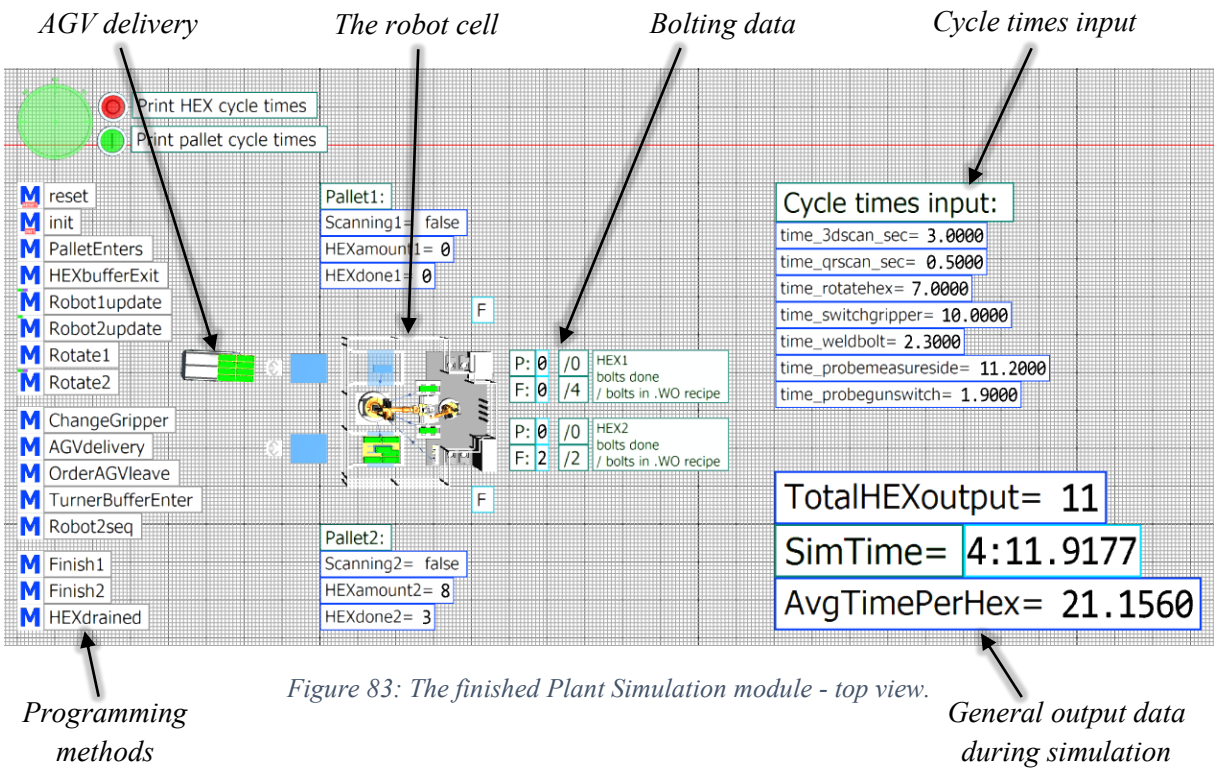


Figure 83: The finished Plant Simulation module - top view.

Receiving pallet from the factory-wide model, or choosing to spawn them locally within the module

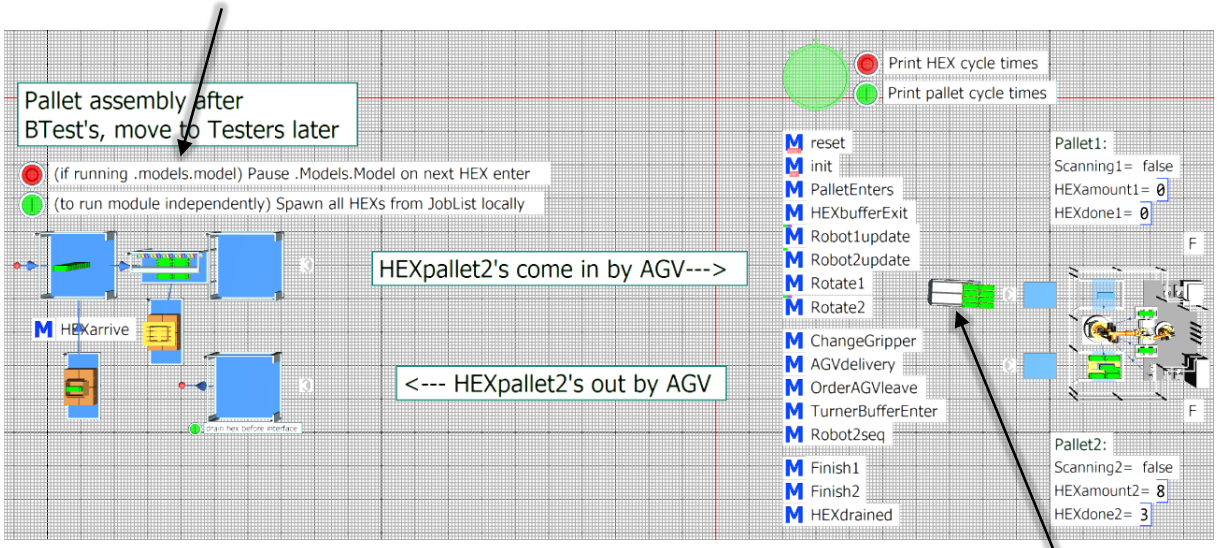


Figure 84: The finished Plant Simulation module - pallet delivery.

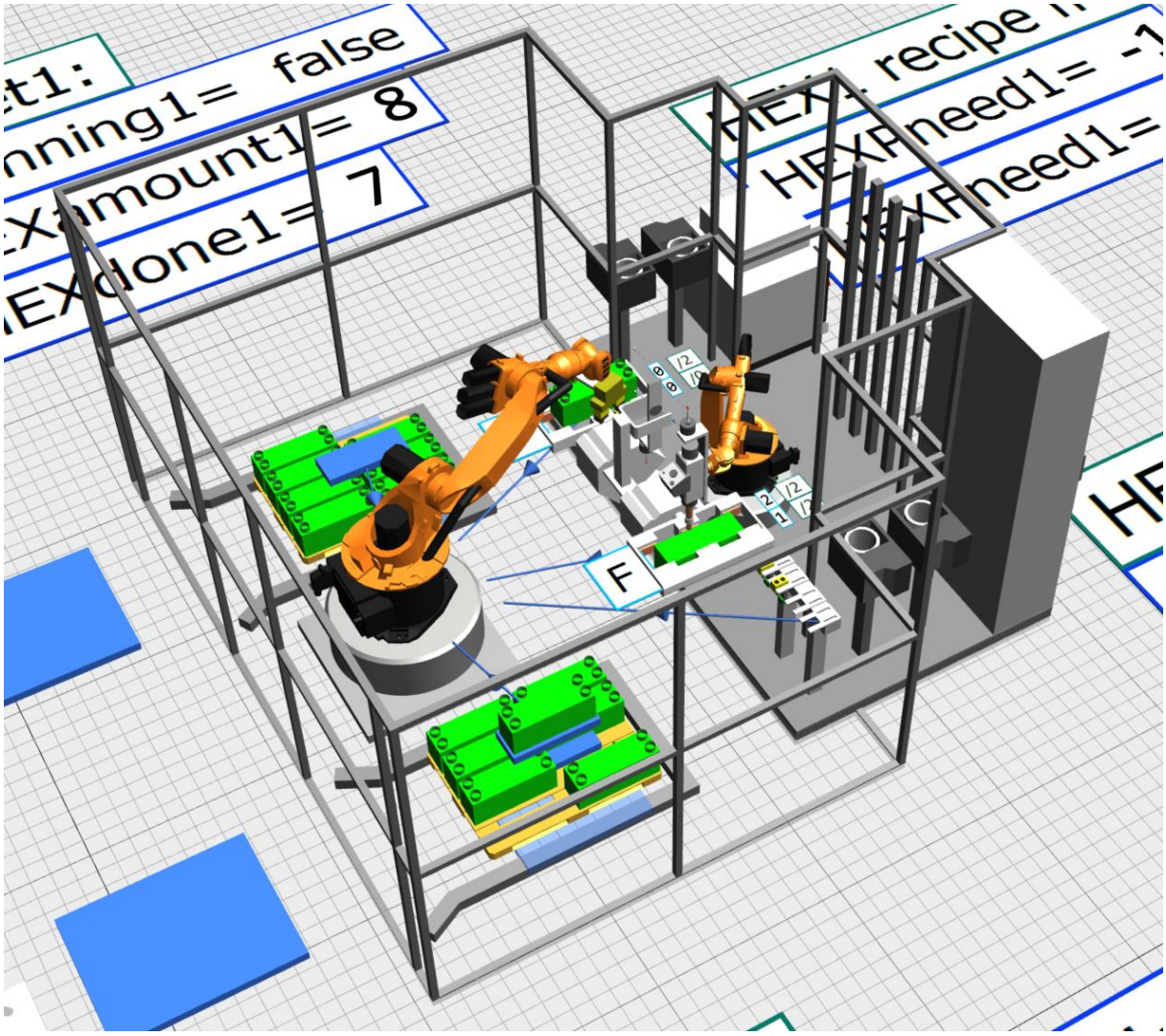


Figure 85: The finished Plant Simulation module - close view.

5.6.2 Simulating and comparing output capacity

The model presented in 4.8.7 containing one manual station and one robot cell and the ability to simulate each separately on the same sequence of production orders from the job list, was used to compare the output capacity between manual and automated welding. The chosen OEE and operator shift times to simulate are presented in Table 8 as well as the resulting maximum output in number of heat exchangers per week. Here, OEE is defined as the percentage of operational time where the station is meant to be operational. That is, excluding things like operator breaks during the day. The results show that one automated cell has roughly the same output as one manual station.

Table 8: Comparison of output between manual and automatic welding.

Cell	Shifts	Max HEX/week
1x Automatic cell, OEE 80	3-shift mon-sun	14762
1x Automatic cell, OEE 80	3-shift mon-fri	10544
1x Automatic cell, OEE 80	2-shift mon-fri	7029
1x Automatic cell, OEE 80	2-shift mon-thu, 1-shift fri (today's schedule)	6327
1x Manual cell, OEE 80	3-shift mon-sun	16207
1x Manual cell, OEE 80	3-shift mon-fri	11521
1x Manual cell, OEE 80	2-shift mon-fri	7534
1x Manual cell, OEE 80	2-shift mon-thu, 1-shift fri (today's schedule)	6643

5.6.3 Comparing cycle times

The expected maximal output capacity of the manual and automated bolting as presented in subsection 5.6.2 is important data to SWEP for when delegating work between manual and automated welding in future factory scenarios. The simulated output capacity is highly dependent on the defined cycle times, which must be measured in the real factory if an accurate simulation model is to be expected. A specification of the cycle times that were chosen to be included in the robot cell module in Plant Simulation is shown in Table 9. A corresponding specification for the manual station is shown in Table 10. The cycle times were either manually clocked in the factory while an operator was working, or they were taken from SWEPs RobotStudio simulation of the cell presented in 4.8.3. To be clear, these are the largest cycle time components existing in the cell, and the real total cycle time is some additive combination of these. Typical cycle times for one heat exchanger is presented at the bottom of each table for comparison.

Table 9: Cycle times - automated stud bolt cell.

Process	Definition	Cycle time
Pallet 3d scan	Time for 3D scanning every new pallet that enters the cell.	3.0 s
HEX QR scan	Time for robot 1 to scan the QR code on each HEX	0.5 s
HEX 180 flip	Time for a turner to flip a HEX by 180 degrees between P- and F-side.	7.0 s
Bolt weld	Time for robot 2 to weld one bolt using its attached weld gun.	2.3 s
Probe measure	Time for robot 2 to measure each HEX side using the touch probe.	11.2 s
Switch gripper	Time for robot 1 to swap between the 100mm and 120mm grippers.	10.0 s
Weld gun swap	Time for robot 2 to swap between different weld guns.	1.9 s
Typical total HEX cycle times: 2 bolts, no flip needed: **20.1 s**. 2 bolts, flip needed: **27.1 s**. 4 bolts, no flip needed: **28.5 s**. 4 bolts, flip needed: **35.5 s**.		

Table 10: Cycle times - manual stud bolt station.

Process	Definition	Cycle time
Manual labor	Time to mount the HEX in the station, retrieve and attach the bolt configuration fixture template, pick up the weld gun, and removing fixture after bolting is done.	15.0 s
Bolt weld	Time to weld one bolt using the weld gun.	3.0 s
HEX 180 flip	Time to manually carry and flip the heat exchanger between P- and F-side.	8.0 s
Typical total HEX cycle times: 2 bolts, no flip needed: **21.0 s**. 2 bolts, flip needed: **29.0 s**. 4 bolts, no flip needed: **27.0 s**. 4 bolts, flip needed: **35.0 s**.		

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 best method for calculating the center and rotation

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.

In between performing the two presented accuracy tests as seen in Table 6 and Table 7, 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 subsection 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 section 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 The effect of gravity in horizontal welding

The unsymmetrical effect from gravity in horizontal welding compared to vertical welding likely weakens the weld quality. In one case, the weld material even separates from the bolt, which is very dangerous to the rest of the robot cell. During the thesis project, well over a hundred bolt welds were done vertically, and this was not seen even once. The frequency and magnitude of this bulging phenomenon in horizontal welding could definitely be reduced by using less energy in the weld, shortening the liquid state duration, but this would affect the bolt quality and limit the range of weld energy allowed. Even though, it shows that horizontal welding is likely a bad idea as there would always be some risk of separation of weld material that could damage the cell greatly. So, vertical welding should be considered as the best alternative, and worth the effort in having a mechanical system turning the heat exchanger between the P- and F-side.

6.3 Simulating the cell in factory-wide production flow

After creating a simulation of the new automated robot cell under development at SWEP in Plant Simulation, the output capacity and cycle times of both the manual stations and the automated robot cell were then known and could be compared. The results in 5.6.2 show that one automated cell has roughly

the same maximal output as one manual welding station. At first, it might be surprising, as a big motivator for automating a process is increasing its possible output. In manual welding, the results in 5.6.3 show that a majority of the cycle time is spent moving the heavy heat exchanger by hand, and the actual welding is done relatively quickly as the operator fastens a fixture that shows where to bolt, where positional tolerance is gained automatically by it being a very precisely manufactured fixture. In the automated cell, picking up and moving a heavy heat exchanger is seen to be quick work for the robots. Instead, most of the cycle time is spent measuring each side with the touch probe for roughly 11 seconds to achieve positional tolerance when welding. Turning a heat exchanger between the P- and F-side takes up a similarly large cycle of approximately 7 seconds, but as robot 2 can always work on the other heat exchanger while the other is being rotated, rotating the heat exchangers in the automated cell barely wastes any cycle time at all.

Even though cycle time is not drastically increased, there are many advantages in replacing manual stations with automated cells. It removes the need for a large storage of fixtures for each possible bolt positional configuration. It requires no human operators, saving manpower and resources. It has an increased availability, and could operate outside normal working hours. It performs more consistent welding quality and better positional accuracy than using the manual weld gun fixture templates that are not always placed firmly enough by operators. The positives are many. To increase its output, the cycle time must be lowered. This is most likely possible by lowering the time needed to measure the sides with the touch probe, possibly by increasing the probe speed, lowering the number of points that need to be probed by having some side position known, or using a totally different system for measuring the side position, like image analysis or laser measurements.

To increase the accuracy of the simulation results, additional measurements and analysis of cycle times in the factory could be performed to input more reliable cycle times in the simulation. More importantly, when the automated cell is operational in the factory, its real OEE can finally be measured, instead of guessed. An OEE of 80% was estimated, but the real effectiveness of the cell could be much less or much more, greatly changing the output of the cell. Nonetheless, the work of building a simulation module of the cell that is compatible with the factory-wide model has been successfully completed in the thesis work and is ready to be used by SWEP to simulate the output capacity of the cell.

6.4 Automating the stud bolt welding process at SWEP

The main objective of the thesis work has been to further develop the already initiated project at SWEP of creating a fully automated solution for stud bolt welding on heat exchangers. After the first period of the thesis work, there were several essential steps left. Even though most of the process was automated from the PLC, HMI, and robot program, there was still a need for an operator to flip units between the P- and F-side, manually fill the bolt tray with bolts for the robot to pick up, as well as loading and unloading each new unit to be processed. To fully automate the process, these tasks had to be automated as well.

The first period of the thesis work showed that both vertical and horizontal weldings are possible, with both having different pros and cons. A second robot or mechanical turning station is needed to flip between P- and F-side in both cases, but more so in the vertical kind. Gravity was seen to affect the weld quality negatively in the horizontal case, which could be avoided with lower weld energy, but this would also affect weld quality and create additional constraints in weld possibilities. The horizontal method had the significant advantage of allowing the P- and F-side to be worked on at the same time, perhaps by two robots, which would greatly increase the maximum output. However, as seen in Table 2, 88% of current units being welded require bolts on only one side, so in today's orders it would not be worth it, but it should be considered in the future if the frequency of units that needs bolts on both sides increase.

It was also concluded that it is not possible to cover the full range of stud bolt lengths used at SWEP with a single welding gun, instead needing at least four to cover the range of 10mm to 40mm bolts. In that case, it must be possible for the robot to switch between these welding guns. Additionally, feeding bolts directly to the welding guns must be automated in some way.

Simulating and comparing the cell in Plant Simulation showed that the main advantages of automating the stud bolt welding process comes not in cycle time, but in accuracy, reliability, repeatability, operating manning, and other advantages. In order for the automated stud bolt cell to have a significant larger output potential than a manual cell, its cycle time must be lowered. Using a touch probe is a simple and accurate but relatively slow method of measuring the heat exchanger side. The robot can not risk breaking the probe on a fast first approach, so it quickly approaches the targeted side point roughly one centimeter outside the expected dimensions, to then slowly but safely searching until contact. So, it should be of focus when attempting to lower the whole cell cycle time. It is a good choice to begin with, but it might be a good idea to consider using image processing or laser measurements to precisely measure the heat exchanger position and rotation. The touch probe can only measure one side point at a time, but using image processing or laser measurements means that all side points and dimensions could be measured at the same time. It also would not require work from any of the two robots, so robot 2 could then put all its available time on welding, greatly reducing the total cell cycle time as robot 2 is the bottle neck between the two, as robot 1 often waits for it to finish.

Coming back to SWEP for the second period of the thesis work, the project had come a long way further. A robot cell of one robot welding vertically was chosen combined with a mechanical turning station and four separate welding guns to cover the full range of bolt lengths needed. As the main design decisions of the cell had been chosen and committed to, building the cell in Plant Simulation proved to be a valuable step in simulating and understanding its output capacity, efficiency and material use when integrated in the factory-wide simulation model, as well as serving as an educational tool in understanding how the cell operates, which is one of the main advantages that SWEP sees in building models in Plant Simulation.

6.5 An ethical reflection on the rise of automation

During the last century, the rise of automation[22] 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 why automation often replaces existing factory jobs and workers. However, 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.

It is clear that many jobs have been lost to automation, just as after many other major technological improvements to society throughout our history. I think it will bring society forward by creating more opportunities and reasons to focus on innovation and research instead of manual labor, ultimately improving life for us all.

7 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] SWEP International AB. The initial RAPID program. *Measures and welds mounting points to a heat exchanger*. [Software]. Code version 1.9.

[4] ABB Robotics. *IRB 4600*. URL: https://new.abb.com/products/robotics/industrial-robots/irb-4600 [visited on 2022-03-31]

[5] Tormach. *Passive probe technical document*. URL: https://tormach.com/media/asset/t/d/td10088_passive_probe_install_0515a.pdf [visited on 2022-03-31]

[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] Frasca, C. (2019) *Virtual analysis and simulation of industrial robots and automation in industry 4.0's development for a conveyors cell's application*. URL: https://webthesis.biblio.polito.it/10782/1/tesi.pdf [visited on 2025-02-12]

[11] Rezende, E., Richetto M. (2023). *Robot cell simulation using Robot Studio and TIA Portal*. URL: https://www.ijaet.org/media/12I77-IJAET1605070-v16-i5-pp434-445.pdf [visited on 2022-02-12]

[12] Sobrin, D., Roman, R., Holubek, R. (2019). *Into the early steps of Virtual Commissioning in Tecnomatix Plant Simulation using S7-PLCSIM Advanced and STEP 7 TIA Portal*. URL: https://www.researchgate.net/publication/337677765 [visited on 2022-02-12]

[13] Ljunglöv, E. (2022). *Automating a robot cell welding process*. My bachelors thesis, continued upon in this master thesis. URL: https://lup.lub.lu.se/student-papers/search/publication/9092903 [visited on 2025-02-13]

[14] ABB Robotics. *RobotStudio*. [Software]. URL: https://new.abb.com/products/robotics/robotstudio [visited on 22-03-31]

[15] Siemens. *Totally Integrated Automation Portal*. [Software]. URL: https://new.siemens.com/global/en/products/automation/industry-software/automation-software/tia-portal.html [visited on 2022-05-09]

[16] Tecnomatix. *Plant Simulation*. [Software] URL: https://plm.sw.siemens.com/en-US/tecnomatix/plant-simulation-software/ [Visited on 2025-02-13]

[17] PROFIBUS. *GSDML Specification for PROFINET*. URL: https://www.profibus.com/download/gsdml-specification-for-profinet [visited on 22-05-23]

[18] Siemens. *SIMATIC S7-PLCSIM*. [Software] URL: https://mall.industry.siemens.com/mall/sv/se/Catalog/Products/5000483 [visited on 2022-05-09]

[19] Siemens. *SIMATIC WinCC*. [Software] 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]

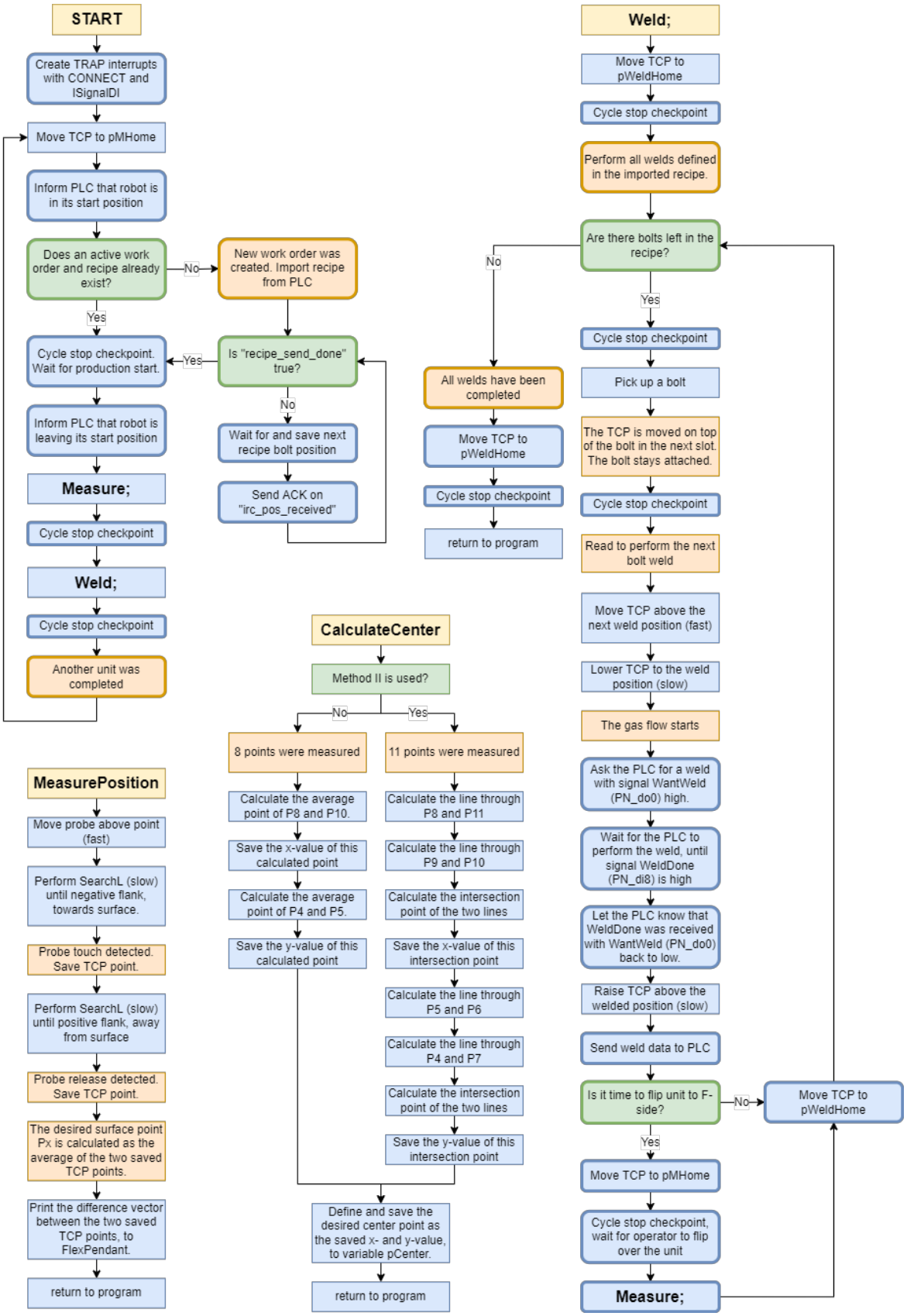
[20] International Electrotechnical Commission. *IEC 61131*-3. URL: https://webstore.iec.ch/publication/4552 [visited on 2022-05-27]

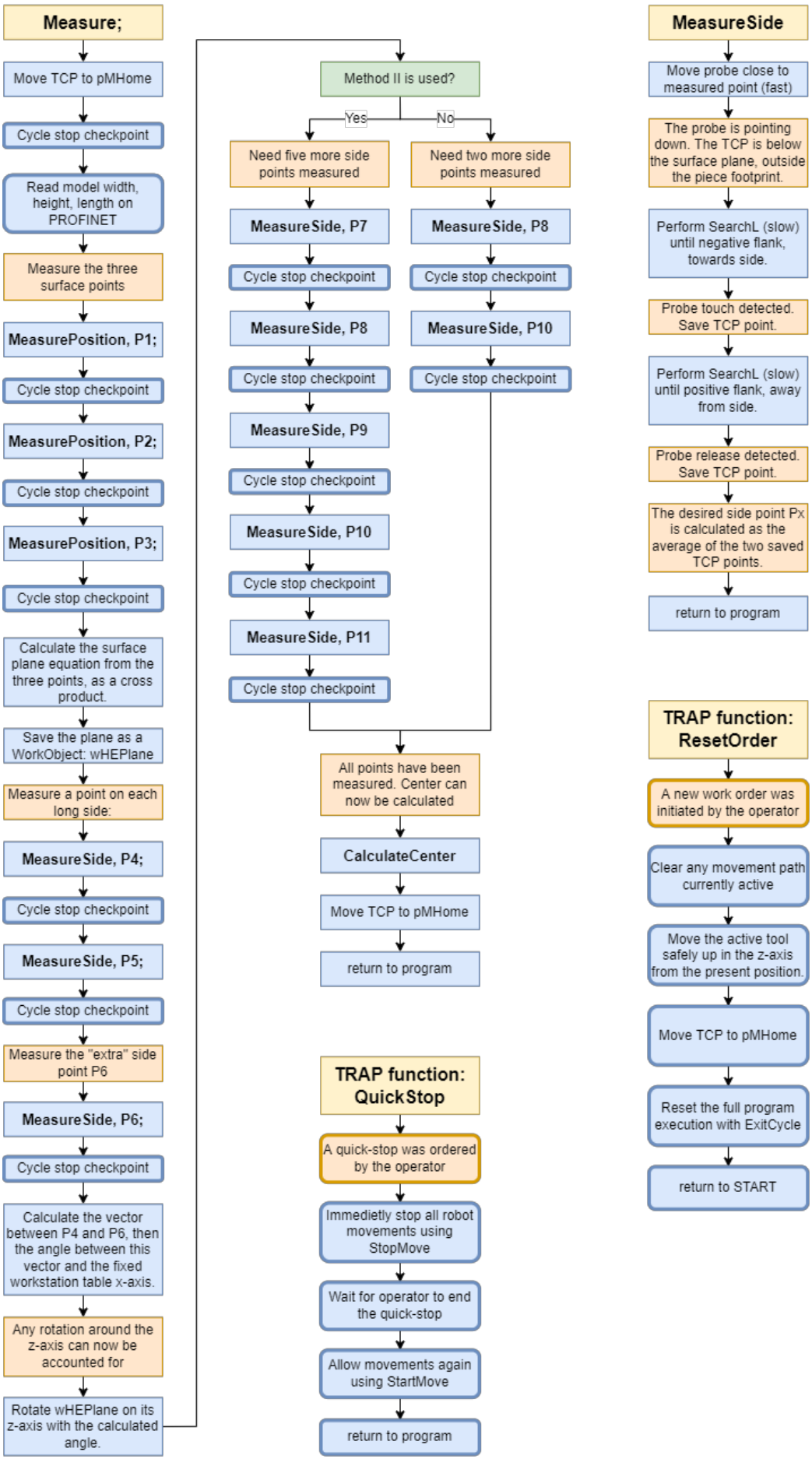
[21] SWEP International AB. *Engineering requirement specification SB welding automation.* Internal company document.

[22] The World Economic Forum. (2021). *The rise in automation and what it means for the future.* URL: https://www.weforum.org/stories/2021/04/the-rise-in-automation-and-what-it-means-for-the-future/ [Visited on 2025-02-13]

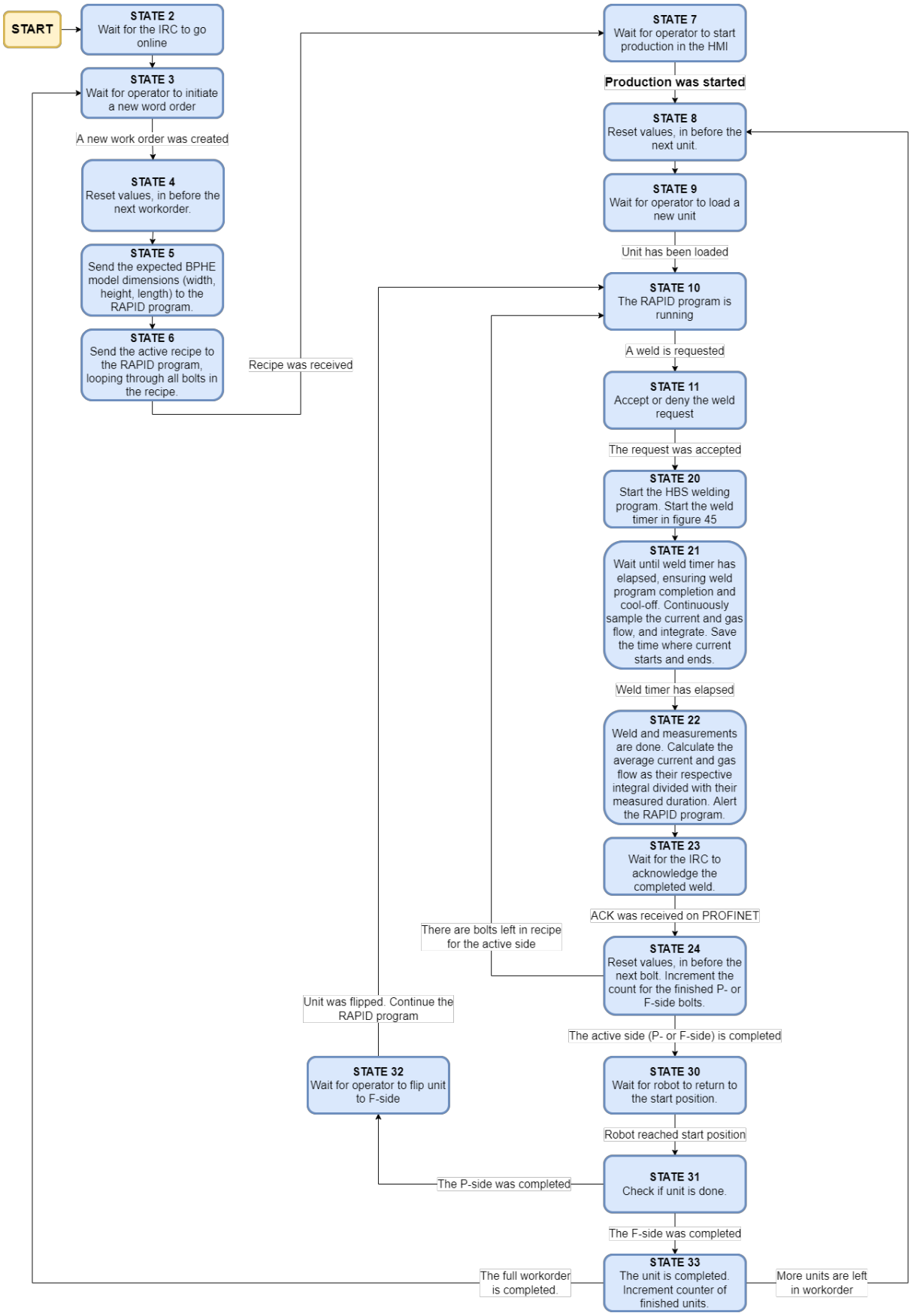
Reading instructions: begin at START. When a box appears with a **bold** name, it alerts you to jump to the separate function with that name. Rectangular parts: the original RAPID program. Rounded parts with thick borders: the additions made in the thesis work.

Appendix A: Flow chart of the pre-existing RAPID program




Measure;
Move TCP to pMHome
Cycle stop checkpoint
Read model width, height, length on PROFINET
Measure the three surface points
MeasurePosition, P1;
Cycle stop checkpoint
MeasurePosition, P2;
Cycle stop checkpoint
MeasurePosition, P3;
Cycle stop checkpoint
Calculate the surface plane equation from the three points, as a cross product.
Save the plane as a WorkObject: wHEPlane
Measure a point on each long side:
MeasureSide, P4;
Cycle stop checkpoint
MeasureSide, P5;
Cycle stop checkpoint
Measure the "extra" side point P6
MeasureSide, P6;
Cycle stop checkpoint
Calculate the vector between P4 and P6, then the angle between this vector and the fixed workstation table x-axis.
Any rotation around the z-axis can now be accounted for
Rotate wHEPlane on its z-axis with the calculated angle.
Method II is used?
Yes
No
Need five more side points measured
MeasureSide, P7
Cycle stop checkpoint
MeasureSide, P8
Cycle stop checkpoint
MeasureSide, P9
Cycle stop checkpoint
MeasureSide, P10
Cycle stop checkpoint
MeasureSide, P11
Cycle stop checkpoint
Need two more side points measured
MeasureSide, P8
Cycle stop checkpoint
MeasureSide, P10
Cycle stop checkpoint
All points have been measured. Center can now be calculated
CalculateCenter
Move TCP to pMHome
return to program
TRAP function: QuickStop
A quick-stop was ordered by the operator
Immedietly stop all robot movements using StopMove
Wait for operator to end the quick-stop
Allow movements again using StartMove
return to program
MeasureSide
Move probe close to measured point (fast)
The probe is pointing down. The TCP is below the surface plane, outside the piece footprint.
Perform SearchL (slow) until negative flank, towards side.
Probe touch detected. Save TCP point.
Perform SearchL (slow) until positive flank, away from side.
Probe release detected. Save TCP point.
The desired side point Px is calculated as the average of the two saved TCP points.
return to program
TRAP function: ResetOrder
A new work order was initiated by the operator
Clear any movement path currently active
Move the active tool safely up in the z-axis from the present position.
Move TCP to pMHome
Reset the full program execution with ExitCycle
return to START

Appendix B: PLC program state diagram



Appendix C: PLC program code

```
//This is the main PLC program. It will run continuously.

IF "btnPress_cycle" OR "btnPress_quick" OR "btnPress_reset" THEN
    "InputManager"(); //a button was pressed. Check if it may perform its action
END_IF;

"Timers_DB"(); //update timers

//Measure current
"ReadAnalogInputs"(analog_in := "current_analog_in",
                   scale_lower := 0, //current value when sensor signal MIN
                   scale_upper := 1500, //current value when sensor signal MAX
                   out_value => "Data".current_now);

//Measure gas flow
"ReadAnalogInputs"(analog_in := "gas_analog_in",
                   scale_lower := 0,
                   scale_upper := 30,
                   out_value => "Data".gaslevel_now);

//Read the analog inputs from PN. Must SWAP bytes in each WORD.
"Data"."lastWeld_x" := WORD_TO_UINT(SWAP_WORD("lastWeld_x_analog")) - "Calc".word_signOffset;
"Data"."lastWeld_y" := WORD_TO_UINT(SWAP_WORD("lastWeld_y_analog")) - "Calc".word_signOffset;

"state_to_irc" := SWAP_WORD(INT_TO_WORD("Data".program_state)); //Send program state to IRC

//Control the custom HMI text of the "Production button", as it is dep on BOTH hmi_cycleControl AND program_state
IF "hmi_cycleControl" THEN
    "HmiData".cycleBtnTextControl := 10; //use text "Cycle stop"
ELSE
    IF "Data".program_state <= 8 OR "Data".program_state = 90 THEN
        "HmiData".cycleBtnTextControl := 0; //use text "Start production"
    ELSIF "Data".program_state = 32 OR "Data".program_state = 9 THEN
        "HmiData".cycleBtnTextControl := 1; //use text "Resume production", is waiting for flip
    ELSE
        "HmiData".cycleBtnTextControl := 11; //use text "Leave cycle stop"
    END_IF;
END_IF;

//the finite-state-machine
CASE "Data".program_state OF

    2: //Wait for IRC to start

        IF "irc_is_online" THEN
            "hmi_resetOrder" := FALSE;
            "HmiData".warningActive := FALSE; //remove when has dropped warning thing cleanly
            "Data".program_state := 3;
        END_IF;

    3: //Wait for "Reset Order"
        ;
    4: //Reset before each WORKORDER

        "hmi_recipe_sent" := FALSE;
        "hmi_cycleControl" := FALSE;
        "hmi_quickStop" := FALSE;
        "ActiveWorkOrder".unitsDone := 0;

        "Calc".weld_wait_ms := T#3000ms; //Weld execute+cooloff time
        "Calc".analog_StartCheck := 20000; //After what analog value (0 to 27648) the current will count as live
        "Calc".word_signOffset := 10000;
        "HmiData".cycleBtnTextControl := 0;
        "ActiveRecipe".currentlyOnPside := TRUE;
```

```
    //Calculate the amount of P- and F-side bolts in the active recipe:
    "ActiveRecipe".activeBolts_Pside := BOOL_TO_INT("ActiveRecipe".P_bolt1_active) + BOOL_TO_INT("ActiveRecipe".P_bol
    "ActiveRecipe".activeBolts_Fside := BOOL_TO_INT("ActiveRecipe".F_bolt1_active) + BOOL_TO_INT("ActiveRecipe".F_bol

    IF "robot_inStartPos" THEN
        "Data".program_state := 5;
    END_IF;

5: //Update the model dimensions to the IRC (is needed as "expected dimensions" when measuring later)

    "hmi_model_width" := SWAP_WORD(INT_TO_WORD("ActiveRecipe".modelDim_width));
    "hmi_model_height" := SWAP_WORD(INT_TO_WORD("ActiveRecipe".modelDim_height));
    "hmi_model_length" := SWAP_WORD(INT_TO_WORD("ActiveRecipe".modelDim_length));

    //Reset before recipe send
    "hmi_recipe_sent" := FALSE;
    "hmi_pos_sent" := FALSE;
    "ActiveRecipe".sendCount := 0;

    "Data".program_state := 6;

6: //Send the full recipe to the IRC

    IF "ActiveRecipe".sendCount < 8 THEN
        "ActiveRecipe".sendCount := "ActiveRecipe".sendCount + 1;
        "SendBoltToIRC"(boltNbr := "ActiveRecipe".sendCount,
                        retVal => "boltActive");
        IF "boltActive" THEN
            "hmi_pos_sent" := TRUE;
            "Data".program_state := 90; //go to recipe side-handler
        END_IF;
    ELSE
        "hmi_recipe_sent" := TRUE;
        "hmi_resetOrder" := FALSE;
        "Data".program_state := 7;
    END_IF;

7: //Wait for PRODUCTION START...

    IF "hmi_cycleControl" THEN
        "Data".program_state := 8;
    END_IF;

8: //Reset before each new UNIT

    IF "ActiveWorkOrder".unitsDone = 0 THEN
        "hmi_cycleControl" := 0; //wait for the first unit
    END_IF;

    //Decide if should start on P- or F-side
    IF "ActiveRecipe".activeBolts_Pside > 0 THEN
        //We have at least one bolt on P-side, so start on P-side
        "ActiveRecipe".currentlyOnPside := TRUE;
    ELSE
        //We have no bolts on P-side, so this recipe is F-side only, so start there.
        "ActiveRecipe".currentlyOnPside := FALSE;
    END_IF;

    "ActiveRecipe".finishedBolts_Pside := 0;
    "ActiveRecipe".finishedBolts_Fside := 0;
    "hmi_want_flip" := FALSE;
    "hmi_cycleControl" := FALSE;
```

```
    "Data".program_state := 9;

9: //Wait for new BPHE to be loaded/placed

    IF "hmi_cycleControl" THEN
        "Data".program_state := 10;
    END_IF;

10: //Weld progam running. PLC waiting for weld request from IRC.

    IF "irc_want_weld" THEN
        //irc is in welding position, gas is flowing. IRC wants a weld.
        "Data".program_state := 11;
    END_IF;

11: //Weld request received

    "Data".program_state := 20; //accept the weld request
    "current_timer_reset" := FALSE;
    "Data".current_avg := 0;
    "Data".gas_avg := 0;
    "Data".time_measured := T#0ms;

20: //Start the weld

    "HmiData".AlarmTriggerBits := 0;
    "weld_power_hbs" := TRUE; //activate the welding program on the HBS
    "weld_timer_on" := TRUE; //start the weld timer
    "Calc".samplesDone := 0;
    "Data".program_state := 21;

21: //Wait until weld is done (weld+coolof), and actively MEASURE all weld params

    IF "current_analog_in" > "Calc".analog_StartCheck AND "Calc".current_isactive = FALSE THEN
        "current_timer_on" := TRUE; //Current has begun, so start timer
        "Calc".current_isactive := TRUE;
    END_IF;

    IF "current_analog_in" < "Calc".analog_StartCheck AND "Calc".current_isactive = TRUE THEN
        "current_timer_on" := FALSE; //Current has stopped, end timer
        "Calc".current_isactive := FALSE;
    END_IF;

    //Wait until weld timer is done
    IF "weld_timer_done" THEN
        "weld_timer_on" := FALSE; //reset timer
        "weld_power_hbs" := FALSE; //turn off the welding program
        "Data".program_state := 22;
    END_IF;

22: //Weld is done. Parameters has been measured.

    //Calculate the average current and gas, from the integrals
    "Calc".time_measured_int := TIME_TO_INT("Data".time_measured); //dont merge with below. HMI needs int.

    IF "Calc".time_measured_int > 0 THEN
        "Data".current_avg := "Calc".current_integral / "Calc".time_measured_int;
    ELSE
        "Data".current_avg := 0;
    END_IF;

    "Data".gas_avg := "Calc".gas_integral / "Calc".samplesDone;
    "weld_done" := TRUE; //alert the IRC
    "Data".program_state := 23;
```

```
23: //Wait for IRC acknowledgement

    IF "irc_want_weld" = FALSE THEN
        "weld_done" := FALSE;
        "Data".program_state := 24;
    END_IF;

24: //Weld is done! Reset between welds.

    "current_timer_on" := FALSE;
    "current_timer_reset" := TRUE;
    "weld_timer_on" := FALSE;
    "weld_power_hbs" := FALSE;
    "weld_done" := FALSE;

    "Calc".weld_wait_ms := T#3000ms; //Weld execute+cooloff time
    "Calc".current_isactive := FALSE;
    "Calc".samplesDone := 0;
    "Calc".current_integral := 0;
    "Calc".gas_integral := 0;
    "Data".current_now := 0;
    "Data".gaslevel_now := 0;

    //Put out alarm if any weld parameter was outside the tolerances
    IF "HmiData".abortIfOutsideTol THEN
        IF "Data".current_avg > "Tol".currentTarget * (1 + "Tol".currentTol / 100) THEN
            "HmiData".AlarmTriggerBits := "HmiData".AlarmTriggerBits OR LREAL_TO_UINT(2 ** 0); //Current TOO HIGH
        ELSIF "Data".current_avg < "Tol".currentTarget * (1 - "Tol".currentTol / 100) THEN
            "HmiData".AlarmTriggerBits := "HmiData".AlarmTriggerBits OR LREAL_TO_UINT(2 ** 1); //Current TOO LOW
        END_IF;

        IF "Data".gas_avg > "Tol".gasTarget * (1 + "Tol".gasTol / 100) THEN
            "HmiData".AlarmTriggerBits := "HmiData".AlarmTriggerBits OR LREAL_TO_UINT(2 ** 2); //Gas TOO HIGH
        ELSIF "Data".gas_avg < "Tol".gasTarget * (1 - "Tol".gasTol / 100) THEN
            "HmiData".AlarmTriggerBits := "HmiData".AlarmTriggerBits OR LREAL_TO_UINT(2 ** 3); //Gas TOO LOW
        END_IF;

        IF "Calc".time_measured_int > "Tol".timeTarget * (1 + "Tol".timeTol / 100) THEN
            "HmiData".AlarmTriggerBits := "HmiData".AlarmTriggerBits OR LREAL_TO_UINT(2 ** 4); //Time TOO LONG
        ELSIF "Calc".time_measured_int < "Tol".timeTarget * (1 - "Tol".timeTol / 100) THEN
            "HmiData".AlarmTriggerBits := "HmiData".AlarmTriggerBits OR LREAL_TO_UINT(2 ** 5); //Time TOO SHORT
        END_IF;
    END_IF;

    //Increment finished bolt count
    IF "ActiveRecipe".currentlyOnPside THEN
        "ActiveRecipe".finishedBolts_Pside := "ActiveRecipe".finishedBolts_Pside + 1;
    ELSE
        "ActiveRecipe".finishedBolts_Fside := "ActiveRecipe".finishedBolts_Fside + 1;
    END_IF;

    //Check if P side is done
    IF "ActiveRecipe".currentlyOnPside AND "ActiveRecipe".finishedBolts_Pside = "ActiveRecipe".activeBolts_Pside THEN
        //This was the last bolt on the P side
        IF "ActiveRecipe".currentlyOnPside AND "ActiveRecipe".activeBolts_Fside > 0 THEN
            //there are bolts on F-side too, so need FLIP
            "hmi_want_flip" := TRUE;
        END_IF;
        "Data".program_state := 30;
    ELSIF "ActiveRecipe".currentlyOnPside = FALSE AND "ActiveRecipe".finishedBolts_Fside = "ActiveRecipe".activeBolts
        //This was the last bolt on the F side
        "Data".program_state := 30;
        "hmi_want_flip" := FALSE;
```

```
        ELSE
            //There are more bolts on the active side
            "Data".program_state := 10;
        END_IF;

    30: //Wait until robot has returned to the start position

        IF "robot_inStartPos" THEN
            "hmi_want_flip" := FALSE;
            "Data".program_state := 31;
        END_IF;

    31: //The current side (P or F) was completed

        IF "ActiveRecipe".currentlyOnPside THEN
            //the P side was done, wait for FLIP, then perform F side, if F bolts exist
            IF "ActiveRecipe".activeBolts_Fside > 0 THEN
                "hmi_cycleControl" := FALSE; //USING THE CYCLE-STOP FOR THE FLIP-WAIT TOO
                "ActiveRecipe".currentlyOnPside := FALSE;
                "Data".program_state := 32;
            ELSE
                //P side was done, and there are NO F-side bolts. Unit is completed.
                "Data".program_state := 33;
            END_IF;
        ELSE
            //The F side was done, so ALL IS DONE!
            "Data".program_state := 33;
        END_IF;

    32: //P-side was completed, so wait for OP to flip the HE

        IF "hmi_cycleControl" = TRUE THEN
            "Data".program_state := 10; //"Resume" was pressed
        END_IF;

    33: //UNIT IS FULLY COMPLETED. Check active workorder, if more units are needed.

        "ActiveWorkOrder".unitsDone := "ActiveWorkOrder".unitsDone + 1;
        IF "ActiveWorkOrder".unitsDone = "ActiveWorkOrder".numberOfUnits THEN
            //The last unit was completed. The full work order is completed!
            "hmi_cycleControl" := FALSE;
            "hmi_resetOrder" := FALSE;
            "Data".program_state := 3;
        ELSE
            //There are units left in the workorder
            "Data".program_state := 8;
        END_IF;

    90:
        //RECIPE SEND: wait for IRC ACK
        IF "irc_pos_received" THEN
            "hmi_pos_sent" := FALSE;
            "Data".program_state := 91;
        END_IF;
    91:
        //RECIPE SEND: wait for IRC reset before next bolt
        IF "irc_pos_received" = FALSE THEN
            "Data".program_state := 6;
        END_IF;
    ELSE
        ;

END_CASE;
```

Method SampleWeldParameters: called every sample during the welding program

```
//This will be called every lms during the welding unit program

IF "Data".program_state = 21 THEN
    //Is in state 21, i.e weld is ongoing. Integrate the current and gas parameters!
    "Calc".current_integral := "Calc".current_integral + 1 *"Data".current_now;
    "Calc".gas_integral := "Calc".gas_integral + 1 * "Data".gaslevel_now;
    "Calc".samplesDone := "Calc".samplesDone + 1;
END_IF;
```

Method InputManager: called when any button on the HMI is pressed

```
//To allow for a complex yet dynamic input system from the HMI (multiple actions used for a single button, etc) the hmi
//buttons only activate dummy-signals btnPress_x, that in turn cause this function to run.
//Here, the hmi input is checked towards the active program state, to allow or ignore the input. For example,
//so that "Start production" may not be triggered if the chosen recipe is empty, or workorder unit count is zero.

"state" := "Data".program_state;

IF "btnPress_cycle" THEN
    //check if it may do its action in the current state
    IF (("state" = 7 AND "ActiveRecipe".activeBolts_Pside + "ActiveRecipe".activeBolts_Fside > 0 AND "ActiveWorkOrder".nu
         OR ("state" >= 9 AND "state" <= 30) OR "state" = 32) AND "hmi_quickStop" = FALSE THEN
        //allow
        "hmi_cycleControl" := NOT ("hmi_cycleControl");
    END_IF;

    IF "state" = 7 AND "ActiveRecipe".activeBolts_Pside + "ActiveRecipe".activeBolts_Fside <= 0 THEN
        "HmiData".AlarmTriggerBits := "HmiData".AlarmTriggerBits OR LREAL_TO_UINT(2 ** 6); //faulty or empty recipe input
    END_IF;

    IF "state" = 7 AND "ActiveWorkOrder".numberOfUnits <= 0 THEN
        "HmiData".AlarmTriggerBits := "HmiData".AlarmTriggerBits OR LREAL_TO_UINT(2 ** 7); //faulty or empty workorder in
    END_IF;

ELSIF "btnPress_quick" THEN
    //check if it may do its action in the current state
    IF (("state" = 10 OR "state" = 11 OR "state" = 30) AND "hmi_cycleControl") OR "hmi_quickStop" THEN
        //allow
        "hmi_quickStop" := NOT ("hmi_quickStop");
    END_IF;

ELSIF "btnPress_reset" THEN
    //check if it may do its action in the current state
    IF "irc_is_online" AND (("state" = 3 AND "robot_inStartPos") OR "state" = 7 OR "state" = 9 OR "state" = 10 OR "state"
        OR "state" = 32 OR "state" = 90 OR "state" = 91) THEN
        //allow
        "hmi_resetOrder" := TRUE;
        "Data".program_state := 4;
    END_IF;

ELSE
    ;
END_IF;

//Reset dummy tags (dont allow simultan presses, so reset them all)
"btnPress_cycle" := FALSE;
"btnPress_quick" := FALSE;
"btnPress_reset" := FALSE;
```

Appendix D: Plant Simulation program code examples

Method Init: called when the simulation initiates

```
HEXbuffer1.ExitLocked := True
Done1buffer.ExitLocked := True
HEXbuffer2.ExitLocked := True
Done2buffer.ExitLocked := True
ProbeBuffer.ExitLocked := False -- pickup probe from the get go
Gunbuffer.ExitLocked := True
SwitchGripperBuffer.ExitLocked := True

TotalHEXoutput := 0

.UserObjects.WeldGun.create(GunBuffer) -- spawn the WeldGun MU for robot2 to use
.UserObjects.MeasuringProbe.create(ProbeBuffer) -- spawn the MeasuringProve MU for robot2 to use
.UserObjects.HEXgripper.create(SwitchGripperBuffer) -- dummy MU to allow robot1 approach

Turner1buffer.SideUp := "-"
Turner2buffer.SideUp := "-"

if SpawnRandomHEXs.Value then
    Source1.ExitLocked := False
    print("WARNING: HEXs are spawned locally in StudBoltCell for testing purposes. Toggle SpawnRandomHEXs to change.")
else
    Source1.ExitLocked := True
end
```

Method PalletEnters: called when a pallet enters the cell

```
if ?.name = "PalletBuffer1" then

    -- new pallet entered, save its arrival time
    ?.EntryTime := EventController.SimTime

    -- 3D scanning delay
    scanning1 := True
    wait time_3dscan_sec
    scanning1 := False

    -- reset pallet hex count
    HEXdone1 := 0
    HEXamount1 := ?.MU(1).NumMU

elseif ?.name = "PalletBuffer2" then

    -- new pallet entered, save its arrival time
    ?.EntryTime := EventController.SimTime

    -- 3D scanning delay
    scanning2 := True
    wait time_3dscan_sec
    scanning2 := False

    -- reset pallet hex count
    HEXdone2 := 0
    HEXamount2 := ?.MU(1).NumMU

else
    print("error PalletEnters")
end

-- give the initial hex to be picked attributes to remember the place it was picked from
?.MU(1).PE(?.MU(1).nextXpick, ?.MU(1).nextYpick).MU(1).createAttr("pickedFromX", "integer")
?.MU(1).PE(?.MU(1).nextXpick, ?.MU(1).nextYpick).MU(1).pickedFromX := ?.MU(1).nextXpick
?.MU(1).PE(?.MU(1).nextXpick, ?.MU(1).nextYpick).MU(1).createAttr("pickedFromY", "integer")
?.MU(1).PE(?.MU(1).nextXpick, ?.MU(1).nextYpick).MU(1).pickedFromY := ?.MU(1).nextYpick

if ?.name = "PalletBuffer1" then

    -- move the next hex pick out of the pallet. entering HEXbuffer1, it will call robot1
    ?.MU(1).PE(?.MU(1).nextXpick, ?.MU(1).nextYpick).MU(1).move(HEXbuffer1)

    -- move pallet grid pickup location
    ?.MU(1).nextXpick := ?.MU(1).nextXpick + 1
    if ?.MU(1).nextXpick > ?.MU(1).XDim then
        ?.MU(1).nextXpick := 1
        ?.MU(1).nextYpick := ?.MU(1).nextYpick + 1
    end

elseif ?.name = "PalletBuffer2" then
```

```
    -- move the initial hex pick out of the pallet. entering HEXbuffer2, it will call robot1
    ?.MU(1).PE(?.MU(1).nextXpick, ?.MU(1).nextYpick).MU(1).move(HEXbuffer2)

    -- move pallet grid pickup location
    ?.MU(1).nextXpick := ?.MU(1).nextXpick + 1
    if ?.MU(1).nextXpick > ?.MU(1).XDim then
        ?.MU(1).nextXpick := 1
        ?.MU(1).nextYpick := ?.MU(1).nextYpick + 1
    end

end
```

Method HEXbufferExit: called after Robot1 picks up a hex from a pallet

```
-- allow a single hex to be picked up by robot1, then lock exit to not trigger another pickup
?.ExitLocked := True

-- create hex attributes needed for welding and replacing it to its pallet later
@.createAttr("BelongsToPallet", "integer")
@.createAttr("needP", "integer")
@.createAttr("needF", "integer")
@.createAttr("sbcycletime", "time")

-- save the time the hex was picked from the pallet, so its own cycle time can be calculated later
@.sbcycletime := EventController.SimTime

-- check the hex buffer that was triggered, save ref to parent pallet
var HEXbuffer : object
var PalletBuffer : object

if ?.Name = "HEXbuffer1" then
    HEXbuffer := HEXbuffer1
    PalletBuffer := PalletBuffer1
    @.BelongsToPallet := 1
elseif ?.Name = "HEXbuffer2" then
    HEXbuffer := HEXbuffer2
    PalletBuffer := PalletBuffer2
    @.BelongsToPallet := 2
end

-- search through JobList for the hex .WO
.Models.Model.Joblist.setCursor(1,1)
var find := .Models.Model.Joblist.find({1,1}..{1,*}, @.WO)
if not find then
    print("ERROR didnt find WO of 1")
    EventController.Stop
end

-- save the JobList bolt configuration for the hex .WO
@.needP := .Models.Model.JobList["BOLTS_P", .Models.Model.Joblist.CursorY]
@.needF := .Models.Model.JobList["BOLTS_F", .Models.Model.Joblist.CursorY]

-- if all hex on the pallet has been picked up, stop picking up more
if PalletBuffer.MU(1).nextYpick > PalletBuffer.MU(1).Ydim then return end

-- give the next hex to be picked attributes to remember the place it was picked from
PalletBuffer.MU(1).PE(PalletBuffer.MU(1).nextXpick, PalletBuffer.MU(1).nextYpick).MU(1).createAttr("pickedFromX", "integer")
PalletBuffer.MU(1).PE(PalletBuffer.MU(1).nextXpick, PalletBuffer.MU(1).nextYpick).MU(1).pickedFromX := PalletBuffer.MU(1).nextXpick
PalletBuffer.MU(1).PE(PalletBuffer.MU(1).nextXpick, PalletBuffer.MU(1).nextYpick).MU(1).createAttr("pickedFromY", "integer")
PalletBuffer.MU(1).PE(PalletBuffer.MU(1).nextXpick, PalletBuffer.MU(1).nextYpick).MU(1).pickedFromY := PalletBuffer.MU(1).nextYpick

-- move the next hex from the pallet to the hexbuffer, to trigger robot1 to move it
PalletBuffer.MU(1).PE(PalletBuffer.MU(1).nextXpick, PalletBuffer.MU(1).nextYpick).MU(1).move(HEXbuffer)

-- move pallet grid pickup location
PalletBuffer.MU(1).nextXpick := PalletBuffer.MU(1).nextXpick + 1
if PalletBuffer.MU(1).nextXpick > PalletBuffer.MU(1).XDim then
    PalletBuffer.MU(1).nextXpick := 1
    PalletBuffer.MU(1).nextYpick := PalletBuffer.MU(1).nextYpick + 1
end
```

(part of) Method Robot1update: called when there is work for Robot 1 to do

```
-- check if robot1 should and can do a pick n place
-- some spagetti code factor present, didnt realise until later how much duplicate code I used, didnt have time to compress,
-- it would be alot more complex to read so not worth it either

while true

    -- DEFINE THE FOUR JOBS ROBOT1 CAN DO
    var needjob_fillturner1 := Turner1buffer.Empty and Done1buffer.Empty and (not Hexbuffer1.Empty or not HExbuffer2.Empty)
    var needjob_fillturner2 := Turner2buffer.Empty and Done2buffer.Empty and (not Hexbuffer1.Empty or not HExbuffer2.Empty)
    var needjob_emptyturner1 := not Done1buffer.Empty
    var needjob_emptyturner2 := not Done2buffer.Empty

    -- CHECK IF ROBOT1 CAN EMPTY TURNER1
    if needjob_emptyturner1 then
```

```
        -- the hex on turner1 is done, move it back to its pallet
        Done1buffer.ExitLocked := False -- trigger pick n place
        waituntil not Robot1.Empty -- wait until pick
        Done1buffer.ExitLocked := True

        if Robot1.MU(1).BelongsToPallet = 1

            Robot1.setDestination(hexdrain1)
            waituntil Robot1.Empty -- wait until place
            hexdone1 := hexdone1 + 1
            hexdrain1.MU(1).move(PalletBuffer1.MU(1).PE(hexdrain1.MU(1).pickedFromX, hexdrain1.MU(1).pickedFromY))
            if hexdone1 = hexamount1 then
                PalletBuffer1.MU(1).createAttr("SBdone", "boolean")
                PalletBuffer1.MU(1).move(AGVbuffer1)
            end

        elseif Robot1.MU(1).BelongsToPallet = 2

            Robot1.setDestination(hexdrain2)
            waituntil Robot1.Empty -- wait until place
            hexdone2 := hexdone2 + 1
            hexdrain2.MU(1).move(PalletBuffer2.MU(1).PE(hexdrain2.MU(1).pickedFromX, hexdrain2.MU(1).pickedFromY))
            if hexdone2 = hexamount2 then
                PalletBuffer2.MU(1).createAttr("SBdone", "boolean")
                PalletBuffer2.MU(1).move(AGVbuffer2)
            end

        else
            print("error robot2update")
        end

    -- CHECK IF ROBOT1 CAN EMPTY TURNER2
    elseif needjob_emptyturner2 then
```

Method Robot2update: called when there is work for Robot2 to do

```
-- check if robot1 should and can do a pick n place

-- like robot1update, it is called once and then automatically when new job is possible
while true

    if Turner1buffer.sideUp = "P" and Done1buffer.Empty and not Turner1buffer.Empty then --and not skipT1B then
        -- the hex on turner1 needs P welding (measure -> pickup gun -> weld -> pickup probe)
        Robot2seq(1, "P")
        if Turner1buffer.needF > 0 then
            Rotate1 -- it was a P+F recipe
        else
            Finish1 -- it was a P recipe, so it is DONE
        end

    elseif Turner1buffer.sideUp = "F" and Done1buffer.Empty and not Turner1buffer.Empty then --not skipT1B then
        -- the hex on turner1 needs F welding (measure -> pickup gun -> weld -> pickup probe)
        Robot2seq(1, "F")
        Rotate1 -- must end on P side, robot1 can only pick place on P side
        Finish1 -- it is DONE

    elseif Turner2buffer.sideUp = "P" and Done2buffer.Empty and not Turner2buffer.Empty then
        -- the hex on turner2 needs P welding (measure -> pickup gun -> weld -> pickup probe)
        Robot2seq(2, "P")
        if Turner2buffer.needF > 0 then
            Rotate2 -- it was a P+F recipe
        else
            Finish2 -- it was a P recipe, so it is DONE
        end
    elseif Turner2buffer.sideUp = "F" and Done2buffer.Empty and not Turner2buffer.Empty then
        -- the hex on turner2 needs F welding (measure -> pickup gun -> weld -> pickup probe)
        Robot2seq(2, "F")
        Rotate2 -- must end on P side, robot1 can only pick place on P side
        Finish2 -- it is DONE
    else
        -- nothing to do for robot1, wait until a hex appears anywhere close to robot1, so this method doesnt run in inf loop until the
        waituntil (Turner1buffer.sideUp = "P" and Done1buffer.Empty and not Turner1buffer.Empty) or (Turner1buffer.sideUp = "F" and Don
    end

end
```

Method Rotate: called to rotate a hex by 180 degrees smoothly over time

```
-- rotate the HEX in Turner1buffer by 180 degrees smoothly over 20 steps, taking time_rotatehex
var steps := 20
var angle := Turner1buffer.MU(1)._3D.Rotation[1]

for var i := 0 to steps
    if angle >= 360 then angle := angle - 360 end
    if angle <= -360 then angle := angle + 360 end
    Turner1buffer.MU(1)._3D.Rotation := [angle, 0, 1, 0]
    angle := angle + 180/steps
    wait time_rotatehex / steps
next

if Turner1buffer.SideUp = "P" then Turner1buffer.SideUp := "F"
elseif Turner1buffer.SideUp = "F" then Turner1buffer.SideUp := "P"
else print("error Rotate1") end
```

Method ChangeGripper: called to make Robot1 change gripper

```
-- simulate the cycle time of switching between 100 and 120 gripper by approaching the gripper then waiting for time_switchgripper

SwitchgripperBuffer.ExitLocked := False -- trigger the robot to approac it
waituntil not Robot1.Empty -- wait until pick
SwitchgripperBuffer.ExitLocked := True
wait time_switchgripper -- simulate time to switch between 100mm and 120mm grippers

if GripperActive = 100 then GripperActive := 120 elseif GripperActive = 120 then GripperActive := 100 else print("error ChangeGripper") end
Robot1.setDestination(SwitchGripperBuffer, false) -- place it where you took it
```

Method AGVdelivery: called on delivery and exit of pallets by AGV

```
-- called when a HEXpallet has been dropped by an AGV into a AGVbuffer,
-- or when a HEXpallet has been bolted and is ready to be taken away by an AGV

if ?.Name = "AGVbuffer1" then

    HEXamount1 := 0
    HEXdone1 := 0
    if ?.MU(1).hasAttribute("SBdone") then
        return
    end
    ?.MU(1).move(PalletBuffer1)

elseif ?.Name = "AGVbuffer2" then

    HEXamount2 := 0
    HEXdone2 := 0
    if ?.MU(1).hasAttribute("SBdone") then
        return
    end
    ?.MU(1).move(PalletBuffer2)

else
    print("error AGVdelivery" + to_str(?.Name))
end
```

Method Robot2sequence: called to make Robot2 measure and weld

```
-- the robot2 procedure: measure with already held probe -> place probe -> pickup gun -> weld -> place gun -> pickup probe

param turnerNum : integer, recipeSide : string
var nbolts := -1

if turnerNum = 1 then

    -- do seq on turner1 hex
    nbolts := -1
    if recipeSide = "P" then
        nbolts := Turner1buffer.MU(1).needP
    elseif recipeSide = "F" then
        nbolts := Turner1buffer.MU(1).needF
    end

    Robot2.setDestination(PlaceToolHere1) -- move prove to hex
    waituntil Robot2.Empty -- wait until place
    wait time_probemeasureside -- measure the side
    PlaceToolHere1.MU(1).move(Robot2) -- pickup probe again
    waituntil not Robot2.Empty -- wait until pick
    Robot2.setDestination(probeBuffer) -- go to weld gun
    waituntil Robot2.Empty -- wait until place
    GunBuffer.ExitLocked := False -- pickup gun
    waituntil not Robot2.Empty -- wait until pick
    wait time_probegunswitch -- switch between weld guns
    Robot2.setDestination(PlaceToolHere1) -- move weld gun to hex
```

```
    waituntil Robot2.Empty -- wait until place

    -- weld each bolt for time_weldbolt x numbolts, and count them
    for var i := 1 to nbolts
        wait time_weldbolt
        if recipeSide = "P" then boltsdoneP1 := boltsdoneP1 + 1  else boltsdoneF1 := boltsdoneF1 + 1 end
    next

    PlaceToolHere1.MU(1).move(Robot2) -- pickup gun again
    waituntil not Robot2.Empty
    Robot2.setDestination(gunBuffer) -- place gun back to buffer
    waituntil Robot2.Empty -- placed back gun
    ProbeBuffer.ExitLocked := False -- pickup probe again, is now back to Robot2 default position before a new weld sequence
    waituntil not Robot2.Empty
    wait time_probegunswitch

elseif turnerNum = 2 then

    -- basically duplicate code but for turner2 and its buffers, kept seperate to not cause too complex pointer/references code

    -- do seq on turner1 hex
    nbolts := -1
    if recipeSide = "P" then
        nbolts := Turner2buffer.MU(1).needP
    elseif recipeSide = "F" then
        nbolts := Turner2buffer.MU(1).needF
    end

    Robot2.setDestination(PlaceToolHere2)
    waituntil Robot2.Empty -- wait until place
    wait time_probemeasureside -- measure the side
    PlaceToolHere2.MU(1).move(Robot2) -- pickup probe again
    waituntil not Robot2.Empty -- wait until pick
    Robot2.setDestination(probeBuffer)
    waituntil Robot2.Empty -- wait until place
    GunBuffer.ExitLocked := False -- pickup gun
    waituntil not Robot2.Empty -- wait until pick
    wait time_probegunswitch
    Robot2.setDestination(PlaceToolHere2)
    waituntil Robot2.Empty -- wait until place

    for var i := 1 to nbolts
        wait time_weldbolt
        if recipeSide = "P" then boltsdoneP2 := boltsdoneP2 + 1  else boltsdoneF2 := boltsdoneF2 + 1 end
    next

    PlaceToolHere2.MU(1).move(Robot2) -- pickup gun again
    waituntil not Robot2.Empty
    Robot2.setDestination(gunBuffer) -- place gun back to buffer
    waituntil Robot2.Empty -- placed back gun
    ProbeBuffer.ExitLocked := False -- pickup probe again, is now back to Robot2 default position before a new weld sequence
    waituntil not Robot2.Empty
    wait time_probegunswitch

else
    print("error robot2seq")
end
```

Method HEXdrained: called when Robot1 places back a hex to its original pallet position

```
-- hex drained! calc its cycle time, update total hex output, and calc avgtimeperhex
@.sbcycletime := EventController.SimTime - @.sbcycletime
TotalHEXoutput := TotalHEXoutput + 1
AvgTimePerHex := num_to_time(time_to_num(EventController.SimTime) / TotalHEXoutput)

if PrintHEXcycleTimes.Value then
    -- if asked to get print info on each hex
    if @.needP = 0 and @.needF = 0 then
        print("HEX FINISHED from WO " + to_str(@.WO) + ": NO BOLTS needed, so it was ignored on pallet after QR was read.")
    else
        print("HEX FINISHED from WO " + to_str(@.WO) + ": received " + to_str(@.needP) + " P bolts, " + to_str(@.needF) + " F bolts
    end
end
```

The following page is left intentionally blank.

“An optimist will tell you the glass is half-full; the pessimist, half-empty; and the engineer will tell you the glass is twice the size it needs to be.” - Oscar Wilde

“Engineers like to solve problems. If there are no problems handily available, they will create their own problems.” - Scott Adams

“Whenever you are asked if you can do a job, tell ’em, “Certainly, I can!” Then get busy and find out how to do it.” - Theodore Roosevelt