

Fredrik Siem Taklo
Hans-Christian Ringstad
Hans Christian Haugan Finnson

Manulab

Manufacturing Laboratory
Development of pilot project and digital twin

May 2020

NTNU

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

Bachelor's thesis

2020



Fredrik Siem Taklo
Hans-Christian Ringstad
Hans Christian Haugan Finnson

Manulab

Manufacturing Laboratory
Development of pilot project and digital twin

Bachelor's thesis
May 2020

NTNU

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences



Norwegian University of
Science and Technology

IE303612 BACHELOR'S THESIS

Manulab



*Fredrik S. Taklo,
Hans-Christian Ringstad,
Hans Christian H. Finnson*

Abstract

This bachelor's thesis details the 2020 "Manulab" project, meant to assist in the development of NTNU's Manufacturing Laboratory in Ålesund, as well as creating a suitable pilot project for it. The Manulab is an Industry 4.0-compliant laboratory for experimenting with factory logistics and manufacturing methods, with equipment supplied by Omron and Amatec. Due to logistical deviations and the lockdown caused by the COVID-19 pandemic, the project was restructured with a focus on creating a digital twin of the lab using RoboDK.

Fredrik Siem Taklo (Candidate number: 10060)
Hans-Christian Ringstad (Candidate number: 10058)
Hans Christian Haugan Finnson (Candidate number: 10067)

Supervisors: Ottar Laurits Osen & Paul Steffen Kleppe

Number of pages: 125 / 331

Preface


This bachelor's thesis was written during the spring semester of 2020 by three students at NTNU Ålesund. The project's purpose was to assist in the development and pilot project of the *Manulab*, an Industry 4.0-compliant robotics laboratory for testing new methods of manufacturing and assembly.

We would like to express our thanks to everyone who helped us on the way, including:

- Our supervisors: Ottar L. Osen and Paul Steffen Kleppe
- The lab faculty: Anders Sætersmoen, Øyvind Andre Hanken and Markus Lyngstad
- Chief engineer André Tranvåg and his apprentices
- Irina Emily-Hansen and Ola Jon Mork
- Omron dept. Ålesund
- Omron Support Norway
- Amatec AS



Fredrik Siem Taklo



Hans-Christian Ringstad



Hans Christian Haugan Finnsen

Ålesund, Møre og Romsdal, Norway

18.05.2020

Contents

Contents	3
List of figures	5
List of tables	8
1 Introduction	9
1.1 Objective	11
1.2 Project Specification	13
1.3 Project Restructuring	13
2 Theoretical basis	14
2.1 Terminology	14
2.2 Lean	15
3 Method	19
3.1 Hardware	19
3.2 Software	23
3.3 Preliminary Work	25
3.4 Product Prototypes	31
3.5 Master Computer and Program	35
3.6 3D Printing	49
3.7 Laser Cutter	62
3.8 Mobile Robots	64
3.9 TM robot arms	70
3.10 GUI	77
3.11 Project restructuring: Digital Twin	85
4 Results	92
4.1 Products	92
4.2 Master Program	94
4.3 3D Printing	100
4.4 Laser Cutter	103
4.5 TM and LD robots	104
4.6 GUI	106

CONTENTS

4.7 Digital Twin	113
5 Discussion	115
5.1 Complications	115
5.2 Suggestions for future improvement	117
6 Conclusions	123
References	124
Appendices	126
A Manulab 2020 Git	127
B Gantt Chart	128
C Working hours and descriptions	130
D Progress Meeting Reports	143
E Manulab Preliminary Report	165
F Sysmac Master Program Source Code	182
G Movicon.NExT HMI Source Code	251
H OctoPrint Communicator Script Source Code	307
I AddTextToDXF Program Source Code	326

List of figures

1.1	Early concept art of the Manulab by <i>Offshore Simulator Centre AS</i>	9
1.2	Overhead map of the planned Manulab and its equipment. Edited from Mazemap.	10
1.3	Overhead map of the temporary test lab made for this thesis. Edited from Mazemap.	11
1.4	Concept art of the Manulab by OSC AS	12
2.1	Kanban storage illustration	17
3.1	TM Collaborative Robot	19
3.2	Omron LD AMR	19
3.3	Omron NY512-1300 IPC	20
3.4	Omron NYP17 Operator Panel	20
3.5	Prusa i3 MK3S 3D Printer	20
3.6	Prusa i3 MMU2S	21
3.7	Raspberry Pi 4B	21
3.8	Trotec S400 Laser Engraver	21
3.9	GCC External Control Interface Unit	21
3.10	Epilog Fusion M2 Laser Cutter	22
3.11	RobotiQ Grippers	22
3.12	Flowchart showing the main process of the lab	25
3.13	L101 (FabLab) before reconstruction	26
3.14	Early overhead view of room L108	27
3.15	State of the test lab in February	27
3.16	L108 taking shape	27
3.17	Network diagram of the LAN setup in room L108	28
3.18	LD-130 during network wiring	30
3.19	Render of first name plate prototype	31
3.20	The different iterations of name plate legs	32
3.21	Discarded open bracket design	33
3.22	Drawing of plate without personalised name.	33
3.23	Rendering of the finished keychain frame design	34
3.24	Overview of the master program process flow using states	36
3.25	Code for adding a new order to queue	37
3.26	Code for removing order from queue	38

3.27	doTask function in the main program, 2 lines per TM robot arm	42
3.28	Overview of the process to write information for the DXF script. . . .	43
3.29	Overview of the process to write commands to the printer.	44
3.30	Overview of the process to read cutter status.	45
3.31	Overview of the process to read printer status.	46
3.32	Gcode-name interpreter function	47
3.33	3D printers for testing in L108	49
3.34	MMU2S unit during assembly	50
3.35	Screenshot showing static IP settings on Raspberry Pi using SSH . .	52
3.36	The OctoPrint web client's graphic interface	53
3.37	API key location in OctoPrint settings	55
3.38	Input used to set up the OctoPrint Communicator script	56
3.39	3D Printer Connection API documentation vs. Python implementation	58
3.40	Flowchart of the processes in the 3D Printer Communications Script .	61
3.41	The three LD robots used for the project	64
3.42	Screenshot of Mobile Planner 5.1.6 GUI	66
3.43	Screenshot of LD robot map from the basement in Mobile Planner 4.7.7	68
3.44	Example program in TMFlow	71
3.45	TM landmark	71
3.46	TM landmark found with Vision	72
3.47	Lifting printing plate with vacuum gripper, first test	73
3.48	Lifting printing plate with hand gripper, first test	73
3.49	Main page of the GUI in Movicon.NExT as seen in the editor.	78
3.50	The top banner used on the main page and order page as seen in the editor.	78
3.51	Order page as seen in the editor.	79
3.52	A status page, here used as the main status page, as seen in the editor.	80
3.53	The second top banner used in the status pages and buffer controls as seen in the editor.	80
3.54	The bottom banner used in the status pages and buffer controls as seen in the editor.	80
3.55	The navigation buttons for the status pages and buffer controls in the GUI as seen in the editor.	81
3.56	The status slot for a Trotec S400 Laser Engraver	81
3.57	The status of the workstation of the mobile robot fleet.	81
3.58	The select...case statement setting the state variables in the tags list.	82
3.59	Buffer control used in the GUI as seen in the editor.	83
3.60	The button used to open the information pop-up window.	83
3.61	Information pop-up for the main page in the GUI as seen in the editor.	84
3.62	Information popup for the status pages in the GUI as seen in the editor.	84
3.63	Screenshot showing the RoboDK interface	85
3.64	RoboDK's custom mechanism designer	86
3.65	Screenshot from RoboDK printer plate retrieval simulation	88
3.66	Screenshot from RoboDK name plate assembly simulation	89
3.67	Screenshot of RoboDK simulation opening & closing the laser cutter .	90

3.68	Some of the reworked and simplified CAD models in Fusion360	91
4.1	The finished nameplate	92
4.2	Finished keychain accessory printed in PLA, without custom name	93
4.3	Overview of the finished program structure and its files	95
4.4	Test run of the ProcessGcodeName function	98
4.5	Private Prusa i3 MK2 setup used to complete printer communication. A Raspberry Pi 1 B+ is mounted to the bottom of the table.	100
4.6	Screenshot from ATC Europe's Mobile Robot Port Forwarding guide	105
4.7	Main page of the GUI in Movicon.NExT.	106
4.8	Order page of the GUI.	107
4.9	Main page for the status pages showing if the workstation has an error or not.	108
4.10	Status page for printer rack 1.	109
4.11	Status page for the LD-Robots.	109
4.12	Buffer control for the buffer storage.	110
4.13	The button used to open the information pop-up window.	111
4.14	Information pop-up window for the main page.	111
4.15	Information pop-up window for the status pages.	111
4.16	Final result of the RoboDK Manulab simulation station	113

List of tables

1.1	Initial Project Specifications	13
2.1	Loose estimates of cycle time	18
3.1	Static IP address reservations for L108	29
3.2	Octoprint Profile specifications used for the Prusa MK3S	53
3.3	Example of 3D Printer Status CSV	59
3.4	Example of Printer Commands CSV from IPC	59
3.5	Table showing possible OctoPrint-IPC commands	59
3.6	Specifications for base LD-90 and LD-130CT <i>Source: Omron^[1]</i>	65
3.7	Safety specifications for base LD-90 and LD-130CT <i>Source: Omron^[1]</i>	67
3.8	Specifications for TM5-900 and TM5M-900 <i>Source: Omron^[2]</i>	70

Chapter 1

Introduction



Figure 1.1: Early concept art of the Manulab by *Offshore Simulator Centre AS*

Manulab - short for *Manufacturing Laboratory* - is a national research infrastructure created by NTNU with funding from the Research Council of Norway^[3]. Comprising multiple fields of technology, these laboratories provide a platform for industries and academia to experiment with new means of production. Different labs have different focuses, ranging from welding to additive manufacturing, Industry 4.0 and medical technology. As of May 2020, the project consists of 11 labs located across Gjøvik,

Trondheim and Ålesund.

In cooperation with NTNU, Omron and Amatec, we have the honour of assisting in the development of such a lab in Ålesund. Inspired by Industry 4.0, it is meant to allow for flexible manufacturing of products while operating as a *dark factory*, meaning minimal human interaction. The following equipment is planned to be implemented:

- 3D printers and a laser cutter for manufacturing of parts.
- Three mobile robots for transportation.
One equipped with a robotic arm; the two others with conveyors.
- Two collaborative robot arms and a rotating table for jigs, meant for assembly of products.
- A four-axis parallel robot for i.e. speedy pick-and-place operations.
- Two industrial robotic arms capable of moving alongside a conveyor belt.
- A cabinet with a master IPC, HMI screen and network equipment to manage the lab.

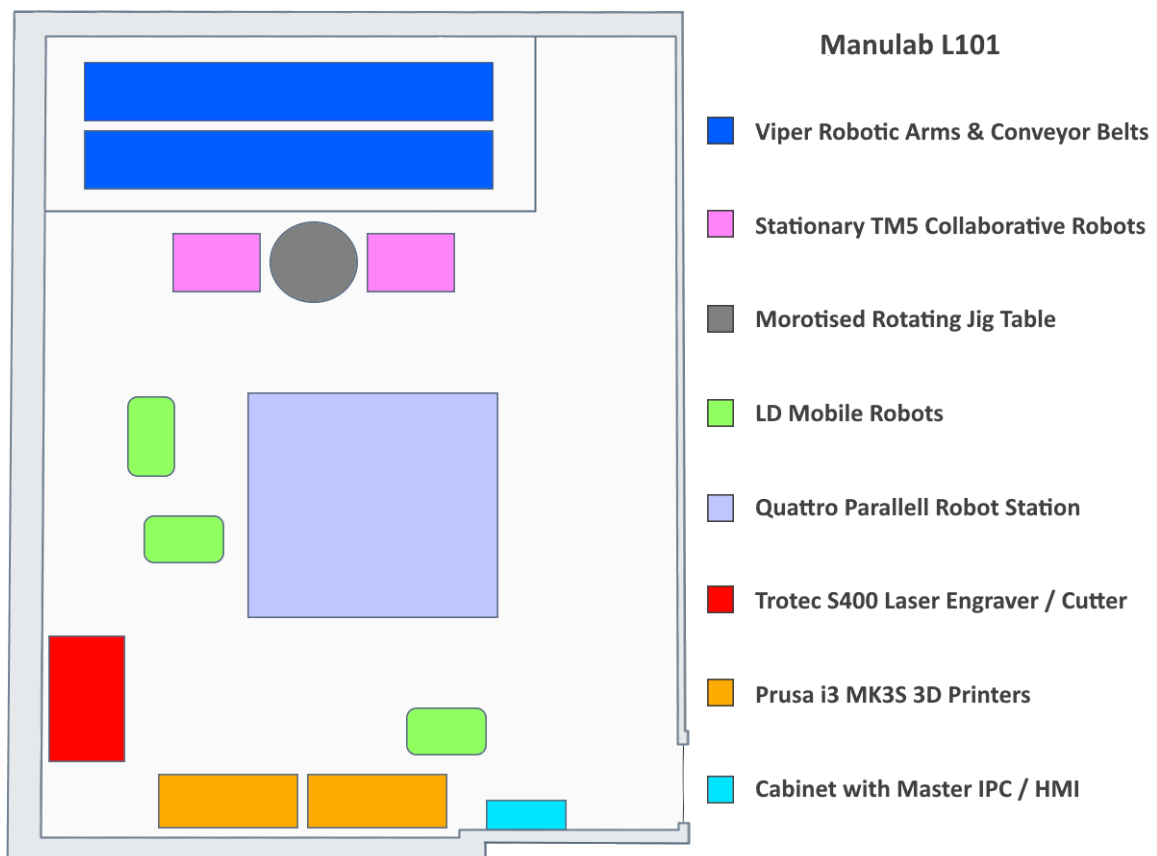


Figure 1.2: Overhead map of the planned Manulab and its equipment.
Edited from Mazemap.

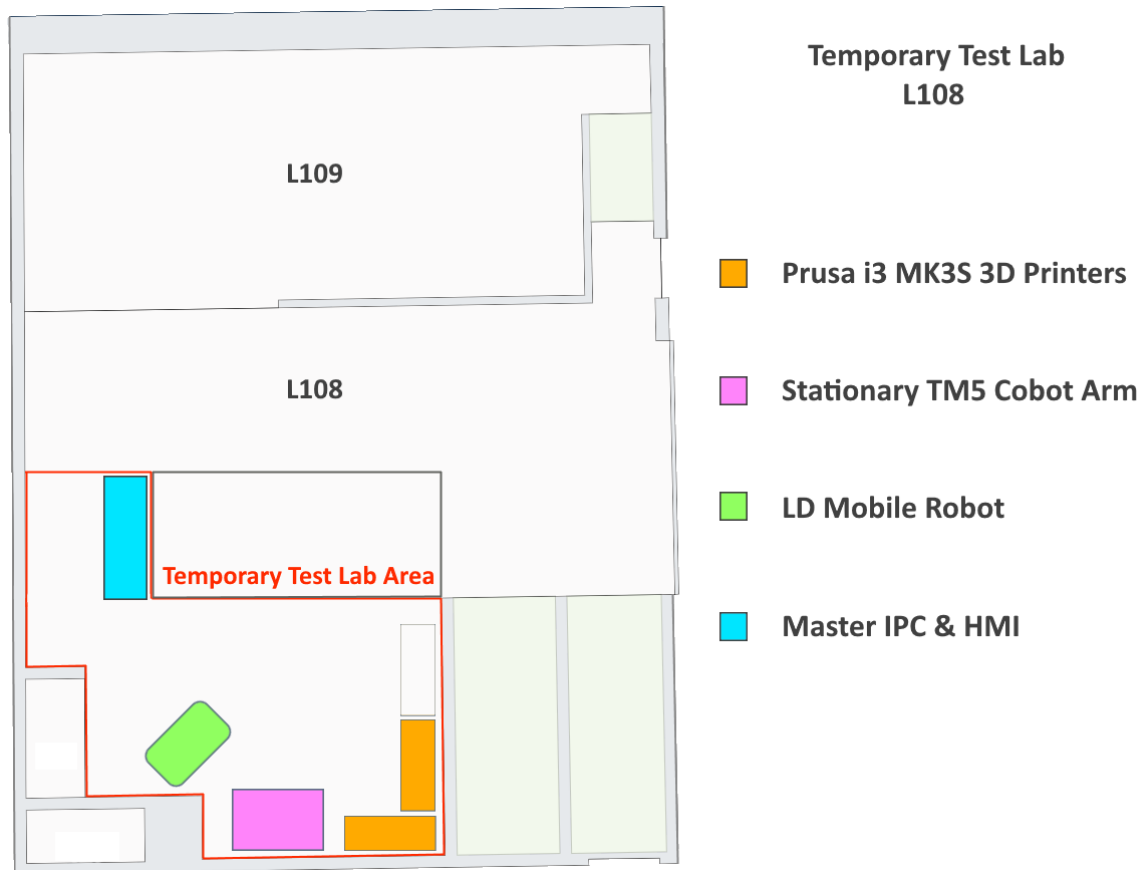


Figure 1.3: Overhead map of the temporary test lab made for this thesis. Edited from Mazemap.

1.1 Objective

The main objective of this pilot project is to enable automatic manufacturing of a range of products, minimising the need for manual labour wherever possible. To achieve this, various equipment and solutions will have to be tested to see what works best for enabling flexible manufacturing. The pilot project is designed with this purpose in mind.

The process begins with a customer or operator ordering a product from a "web shop" connected to an industrial PC. On this PC's internal controller, the order is parsed and tasks are delegated to the necessary machines to complete the product. Modified Omron LD Mobile robots take care of the logistics, bringing raw and processed materials to and from the machines, while two Omron TM5 collaborative robotic arms assist in the assembly process and work in tandem with the lab equipment.

As test cases, two prototype products were designed for manufacturing and assembly: a desktop sign and a keychain with customisable logos and names.



Figure 1.4: Concept art of the Manulab by OSC AS

1.2 Project Specification

The project must...	The project should...	The project could...
Work as a production line, converting raw materials into a finished product	Work as initially intended, demonstrating a production process	Have a simulated or digital counterpart
Utilise the available machinery to demonstrate their capabilities	Be flexible enough to allow for changes to the production line	A GUI showing progress of the current tasks
Be secure for humans traversing the open workspace	Allow for creating different products simultaneously	Improve upon clients' specifications
Communicate between the different stations via LAN	Include suggestions for future improvements, and act as a good basis for that	If possible, be entirely autonomous from start to finish

Table 1.1: Initial Project Specifications

1.3 Project Restructuring

On March 12, the development of the project took an unexpected turn, as public facilities across Norway were closed off to combat the spread of the COVID-19 disease. Due to this, the project could no longer proceed as planned, and had to be restructured.

In coordination with advisors, our team decided to opt for a form of "digital twin" for the Manulab. While finishing the work that we could do from home, we would begin to gather CAD files for the equipment and then eventually start working on simulated counterparts for various processes in the lab using *RoboDK*^[4].

More about these complications and the restructuring itself can be found in section [5.1: Complications](#).

1.3.1 New Project Objectives

As our original goals could no longer be achieved, new objectives were set to be worked on until the thesis' deadline in May:

- Complete any software and documentation that is possible to do from home.
- Work with advisors to create a library of CAD files and kinematic models of lab equipment.
- Research RoboDK's capabilities and learn how to use it.
- Investigate possibilities for offline programming.
- Use RoboDK to simulate the lab where applicable.

Chapter 2

Theoretical basis

This chapter contains some of the underlying theory and terminology that the project is based upon.

2.1 Terminology

NTNU Norwegian University of Science and Technology

ARCL Advanced Robot Control Language

GUI Graphical User Interface

HMI Human Machine Interface

LAN Local Area Network

IP Address Internet Protocol Address

DHCP Dynamic Host Configuration Protocol

UDP User Datagram Protocol

PLC Programmable Logic Controller

IPC Industrial PC, for high dependability and precision

SSH Secure Shell, a protocol for secure communication over insecure networks

Gcode Numerical control programming language used in i.e. 3D printers

Structure Data type used in PLC programming containing several sub-variables, also referred to as an object

Modbus Communication protocol commonly used in industrial applications by PLCs.

Digital Twin Digital, simulated counterpart to a physical system

2.2 Lean

Seeing as the Manulab is in essence a production line, it was decided that integrating Lean manufacturing principles into the production process would be a good idea to improve the production flow of the Manulab.

2.2.0.1 Process mapping

One of the most important parts of Lean manufacturing is creating a process mapping of the production line. This helps visualise the process flow to make it easier to understand all the different sub-processes that take place throughout the production process. The process mapping for the Manulab is shown and discussed in section [3.3.1](#).

2.2.0.2 Project planning

Another important part of Lean manufacturing is having an organised and clean overview of work tasks, project progress, and who is responsible for what part of the process of developing the Manulab. In order to keep track of this, the web applications Asana and Instagantt were utilised throughout the project to create, maintain and follow a gantt chart for the project work. See appendix [B](#) to view this chart.

2.2.1 Pure and necessary waste

In production processes, there are generally two types of wastes; pure waste and necessary wastes. Pure waste benefits no one and adds nothing of value in the production process, while necessary wastes are still wasteful, but are required in order to ensure smooth operation of the production process. In a project as large and with as many sub-tasks as this, some necessary and unnecessary waste will inevitably be involved.

Some examples of pure waste that seem likely to occur are:

- The time for the process to complete product assembly may become worse if people get in the way of the LD robot while they're moving between stations, as it has to continuously reroute to avoid the people in its path. This does however not cause more than a few seconds delay unless they are actively blocked.
- When cutting parts for the acrylic plates, it is likely that there will be a lot of leftover acrylic material in the cutter when products have been cut from it which has to be thrown away. This may be solvable, as described in section [5.2.2.2](#).

On the other hand, there are some examples of necessary wastes that are likely to occur in the Manulab project.

- While a storage of spare parts is generally looked at as a waste from a Lean perspective, keeping a kanban storage to avoid the bottleneck that is 3D-printing

new parts on demand provides a significantly larger benefit to cycle time and production flow than it would've been if there was no kanban storage.

- The printing plates that the printers use will require periodic maintenance to remain usable over a reasonable amount of time. This causes downtime for the printers, which may negatively impact the lab depending on the current workload, however having enough printers makes this issue avoidable.
- When lifting the acrylic plate out of the laser cutter, it very likely will be required to have some kind of attachment point that the robot is able to lift from. This means that less products will be able to fit onto one acrylic plate, causing some waste to occur.

2.2.1.1 5S

In order to work towards the five principles of 5S: Sort (Seiri), Set in Order (Seiton), Shine (Seiso), Standardise (Seiketsu) and Sustain (Shitsuke); a few things will have to be taken into consideration for the Manulab.

- Kanban storage to ensure parts for assembling products are sorted, and with different tasks being performed at separate stations.
- Frequent cleaning and maintenance on the robots to ensure smooth and reliable operation.
- All equipment will have fixed positions. Robots will go to a pre-determined place where they will be if they have no work tasks, while network equipment and computers will have fixed and easily accessible positions where they will be accessible at all times.
- The lab in general has to be easily accessible and open for people to be able to move through without being obstructed by equipment or robots, for example to refill resources for production and perform maintenance on the various robots in the lab.
- Standardising all working stations, so that they communicate using the same protocol where applicable, use mechanical solutions that make them all equally capable of working with a variety of different products, and other methods to help reduce the time and effort required to make adjustments or restructure parts of the lab.
- Procedures for work and development in the lab have to be put in place and maintained.

Once optimal methods for fulfilling the other principles of 5S have been put in place, they will be upheld to fulfill the last principle.

2.2.2 Kanban storage

Due to the fact that 3D-printing components takes a lot of time, making it part of the main production process would make it a very time-consuming bottleneck. To help combat this, a kanban storage was added in as an integral part of the project in order to cut down on one of the 8 wastes of Lean.

From the operator panel in Movicon, an operator is able to set the buffer size in the kanban storage, in a range of 0 to 20. If the current amount of a given part in storage is smaller than the current buffer size, then the master computer immediately orders new printed parts until the combined total of current parts in storage and the parts being printed is equal to the buffer size.

Once an order is placed and a part is taken from kanban storage for production, a new part will immediately be printed to fill the storage back up to buffer size.

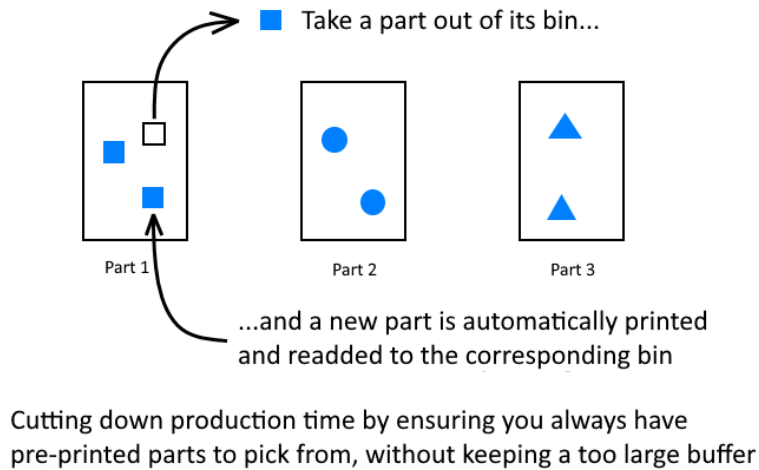


Figure 2.1: Kanban storage illustration

2.2.3 Production time

A loose estimate to the cycle time of the production process is included below. In its current version, the PLC program is only able to run one production at a time, meaning a longer time between each produced product. The program does make some efficiency improvements though, such as running the 2 minute long cutting process in the 2 minute time it's estimated that it takes to get a printed part from the kanban storage to assembly, and then driving over to the cutter.

Process	Time	Simultaneous process	Time
Process order in Movicon	1s		
Process order on PLC	1s		
Drive LD to kanban	15s	Perform laser cutting and engraving	2m
Pick part from kanban	45s		
Deliver part to assembly	15s		
Put part on assembly table	30s		
Drive LD to cutter	15s		
Open cutter and pick up part	1m30s		
Drive LD to assembly	15s		
Put acrylic piece on assembly table	30s		
Perform assembly	1m30s		
Pick up product	30s		
Drive product to delivery	15s		
Place product for delivery	30s		
Total:	7m02s		

Table 2.1: Loose estimates of cycle time

While it would be preferable to start a new production process from the point at which the assembly begins, it would require significant changes to the program which there simply was no time for, and is discussed in section 5.2.12. If this was done, only 4m17s would have to pass before the next product could start production, and thus the production line outputting one product for every 4m17s instead of 7m02s. Further improvements may be made, but then the program would have to move away from the state-based method it currently uses.

In addition to the times above, the printers take roughly 1 hour and 30 minutes to print two nameplate legs, and 45 minutes to print a keychain frame, with the retrieval of the printed part by the LD robot with a TM arm taking somewhere along the lines of 2m00s to 2m30s, depending on whether the TM arm is needed to move the printer plate onto the table for separating pieces, or if the LD-90's conveyor can handle it.

Considering the fact that the customer will be on site when placing the order from the GUI and upon the completion of the product, the **lead time** (time from ordering to delivering a product) is practically identical to the **cycle time** (time spent per product), as there would be no delay between assembly completion and the customer's ability to pick up the part. As there is no fixed customer demand, calculating the **takt time** (time needed per product) is not feasible at this point.

Chapter 3

Method

3.1 Hardware

Note: All images are sourced from their respective manufacturers.

3.1.1 TM Collaborative Robots

For the Manulab project, two Omron TM5-900 collaborative robot arms will be used, as well as one TM5M-900 arm attached to the top of an LD-130CT mobile robot. These "cobots" are equipped with force sensors in each joint, making them sensitive enough to be ISO/TS 15066-compliant. This means that they can be used to work together with humans, eliminating the need for security barriers.



Figure 3.1:
TM Collaborative Robot

3.1.2 Omron LD Autonomous Mobile Robots

The project utilises a fleet consisting of three LD Autonomous Mobile Robots. Two LD-90 robots will have a conveyor belt on their back to carry objects between working areas, while one LD-130CT will be carrying a TM5M-900 robot arm on its back, which will perform various tasks throughout the working area.



Figure 3.2:
Omron LD AMR

3.1.3 Omron NY512-1300 Industrial PC

The Omron NY512 IPC was used as the lab's master computer during development. Communication and logistics are handled by a Sysmac program running on the internal controller. The SCADA/HMI software Movicon.NExT runs on the embedded Windows PC and enables the end user to interface with the lab. A *S8BA Uninterruptible Power Supply* was attached for added security.



Figure 3.3: Omron NY512-1300 IPC

3.1.4 Omron NY532-Z300 Industrial Panel PC

The 15.4 inch *Omron NY532-Z300-112214T20 Industrial Panel PC* is the final lab's primary human-machine-interface. A controller in its own right, it can also be used to display the NY512's graphics. It enables the users of the lab to interact with it through a touch screen, or alternatively, by connecting a mouse and keyboard to operate the IPC like a normal Windows computer. A similar panel was borrowed from Omron dept. Ålesund to be used for testing HMI capabilities at the temporary lab in L108.



Figure 3.4: Omron NYP17 Operator Panel

3.1.5 Prusa i3 MK3S 3D Printer

An open source, fused deposition modeling (FDM) 3D printer, made by Prusa Research. It is based on the RepRap project. Equipped with magnetic PEI-coated steel sheets, filament sensors. Each printer is connected to a Raspberry Pi to enable remote network operation.



Figure 3.5: Prusa i3 MK3S 3D Printer

3.1.5.1 Prusa i3 Multi Material Upgrade 2S

An add-on to the Prusa i3 MK3 / MK3S that enables simultaneous printing using different filaments. One was constructed for use in the lab, but after testing it proved to be unfit for the project. Due to the extra maintenance required, as well as the amount of periodic errors it introduces, we propose that MMU2S installed in the lab should be operated manually for now.

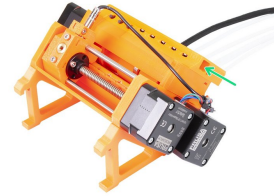


Figure 3.6:
Prusa i3 MMU2S

3.1.6 Raspberry Pi Model 3B+ & 4B

The Raspberry Pi is a small single-board computer widely used for its versatility and form factor. In this project, we've flashed them with Octoprint to enable remote control of the 3D printers, then made a program in Python that handles communication between them over HTTP using a REST API. Initially we only had access to the model 3B+, but the finished lab uses the model 4B.



Figure 3.7:
Raspberry Pi 4B
v

3.1.7 Trotec Speedy 400 Laser Engraver

The machine used for laser cutting and engraving in the lab itself. It has a CO2 laser with a power output of 120W. Requires a robust ventilation system to evacuate noxious fumes. For this project, it is meant to be controlled by the main IPC via PLC and operated by a robot.

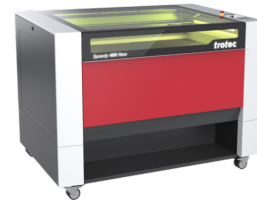


Figure 3.8: Trotec
S400 Laser Engraver

3.1.7.1 GCC External Control Board

Bought and installed by Amatec, this board enables remote control of the Trotec S400 via digital signals. It can start and stop jobs, turn the laser on and off, as well as provide status signals for the currently running job.



Figure 3.9: GCC
External Control
Interface Unit

3.1.8 Epilog Fusion M2 Laser Cutter

The Epilog Fusion M2 120W Laser Cutter was used for learning about laser cutting and prototyping of products, but could not be used in the development of the lab as it proved to be unsuitable for automation and remote operation.

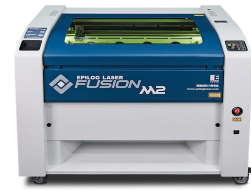


Figure 3.10: Epilog Fusion M2 Laser Cutter

3.1.9 RobotiQ Grippers

For the TM robotic arms, three types of grippers from RobotiQ were used. Two conventional grippers (Hand-E and 2F-140) and one vacuum gripper (EPick). The EPick has a built-in compressor, meaning no external pneumatic source is needed. The grippers connect to the TM control box's serial ports and can be accessed in the TMFlow software after installing so-called *TMComponents* packages from RobotiQ's web sites.



Figure 3.11: RobotiQ Grippers

3.1.10 Network Switch

Several network switches were used to connect the various equipment in the project, both in the temporary development lab and the Manulab itself. Initially a *Netgear ProSafe GS108E* was used, then later supplemented with a *TRENDnet TPE-S50 Power over Ethernet-switch*. The latter was also supposed to power the new Raspberry Pi 4B units as they were implemented into the printer rack.

3.1.11 Wireless Router

Wireless routers were used for a variety of reasons, acting as DHCP servers for computers connecting to the network as well as the first-time-setup of the Raspberry Pis. They also handle communication between the cabled lab equipment and the mobile robots.

3.2 Software

3.2.1 Omron Sysmac Studio

Sysmac Studio is a software suite for programming various Omron controllers and equipment. In accordance with the IEC 61131-3 standard, it supports standard PLC languages like structured text and ladder, and also contains HMI programming tools, servo controls and more.^[5]

Its main purpose in this project is to program the IPC and PLCs that are responsible for the main logic of the laboratory. They will delegate tasks and manage communication with other devices alongside the SCADA-software *Movicon.NExT*.

3.2.2 PyCharm

Pycharm is an integrated development environment (IDE) made specifically for the Python programming language. It provides debugging options, version control with Git, as well as other useful tools. In this project, it was used to program a script for customising DXF-files for laser cutting, as well as scripts for adding communication and various functions to Raspberry Pi-connected 3D-Printers.

3.2.3 TM Flow

TM Flow is used to program the movement of the TM collaborative robot arms. It's a flowchart-based software, and robot movement is made either by manually adding in function blocks, or automatically creating blocks using the buttons on the robot arm itself. Through this software, the robot's movements, gripper, vision camera, timers, communication, and so on can be controlled.

For the project, a set of pre-made programs will be created on the robot arms to handle their various tasks. Depending on the value they're given by the master program, they will then run a specific segment of the program that fits the command that is given.

3.2.4 Omron Mobile Planner

Mobile planner is used for controlling the LD autonomous mobile robots. Using this software, you get a graphical map of the area the robots are moving around, where you are able to set up various goals, assign tasks to the various robots, create macros, and changing the configuration of the individual robots.

In the case of the LD robots, various goals throughout the Manulab working area will be created. Depending on the progress on the current production, the master program will then assign a robot to go to a specified goal and do a given task.

3.2.5 Autodesk Fusion 360

Our chosen CAD-based software to use for 3D modelling is Autodesk Fusion 360. While there are a lot of options for CAD, Fusion 360 was chosen because of its simplicity, functionality and the group's earlier experiences with the program.

3.2.6 Prusaslicer

Prusa Research's in-house slicing software, based on the open-source Slic3r project. Converts 3D models into g-code that can run on a 3D printer. In this project, STL-objects from Fusion 360 are sliced for the Prusa i3 MK3S.

3.2.7 AutoCAD

For the 2D-drawings of the plates for the laser cutter the computer-aided design (CAD) software AutoCAD was used. It can be used to edit and draft 2D geometry and 3D models, annotate texts and dimensions and add images^[6].

3.2.8 Movicon.NExT

Supplied and recommended by Omron for use in the Manulab project, Movicon.NExT was chosen as the main HMI application. It is a high-level SCADA program that enables different proprietary hardware to be connected to the same GUIs, web servers and more.^[7]

3.2.9 RoboDK

After the university closed for students due to the COVID-19 outbreak, it was suggested by the supervisors to try to make a digital twin of the Manulab to simulate it, with RoboDK^[4] acting as their recommended platform. RoboDK allows for simulating industrial robots and kinematics, and was chosen for its relative simplicity so that future students could quickly pick it up and build upon it. The software should be able to simulate most of the lab's mechanisms, but requires reworking most CAD-models to simpler geometries so as not to become too heavy to run.

3.3 Preliminary Work

3.3.1 Main Process Overview

The main process starts with a web store, where an end user decides on a product to order. The IPC processes the order, checks for stored parts and gives the necessary commands to manufacture the product. The robots, laser cutter and 3D printers are all given orders in this manner. The autonomous Omron AMRs handle the logistics between stations, including transport of raw materials to the laser cutter and finished parts from the 3D printers to a kanban-style buffer storage.

Since 3D printing is a clear bottleneck in the manufacturing process, the main program will aim to keep a set amount of parts in storage for smoother operation. This number is determined by a user on the IPC.

When all necessary parts of the product are completed, they are sent to the appropriate robot stations for assembly. If the product’s geometry does not allow for assembly, it is delivered to a pick-up point, from which the end user can retrieve it.

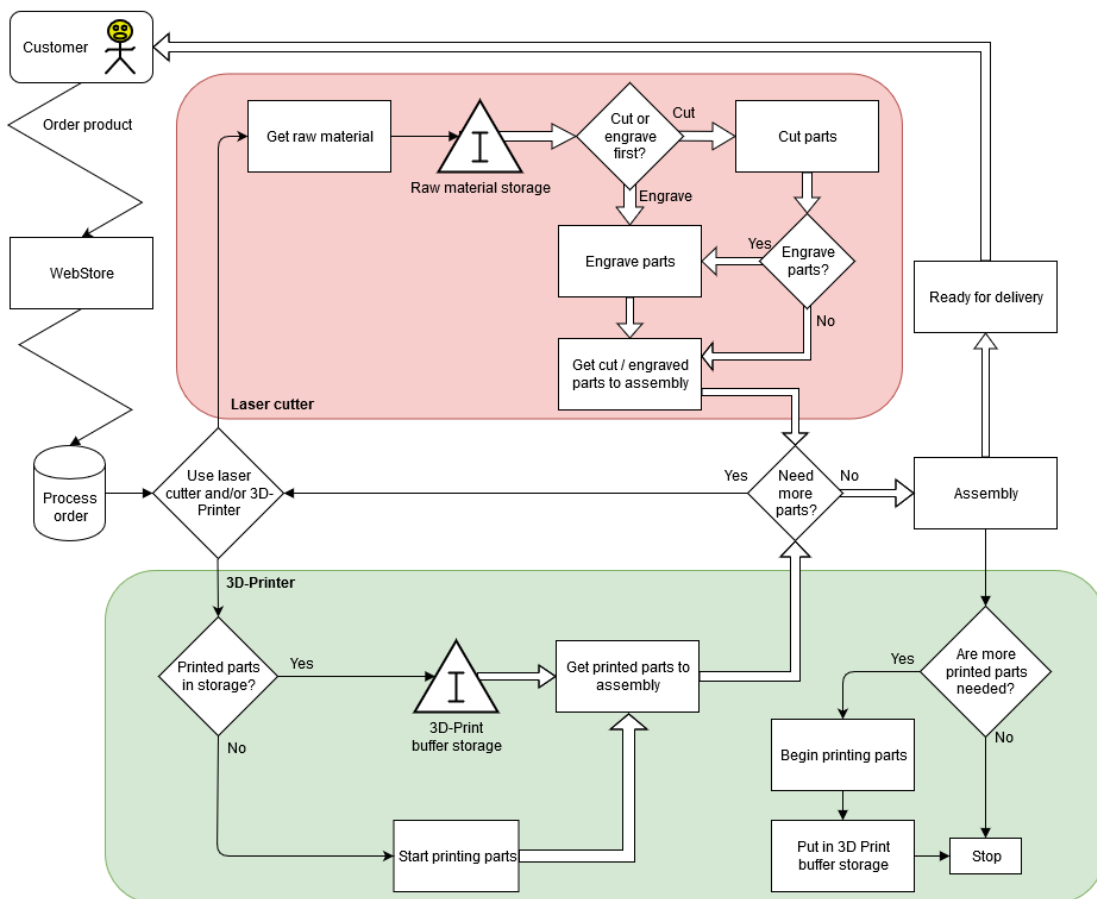


Figure 3.12: Flowchart showing the main process of the lab

3.3.2 L108 Test Lab

Up until 2020, room L101 was used for the so-called *FabLab*, housing a variety of 3D printers and a laser cutter among other things. Before the Manulab could be set up, the room itself had to be entirely renovated from top to bottom. A new epoxy floor was to be laid down, and new wiring and pipes installed. This was to be carried out by The Norwegian Directorate of Public Construction and Property (*Statsbygg*).



Figure 3.13: L101 (FabLab) before reconstruction

Demolition of the old lab started in January, as this thesis was entering its early stages. This was a major challenge, as it left us without facilities to develop the project. Thankfully, chief engineer André Tranvåg was generous enough to clear out part of the old plastics lab in room L108. This would serve as the testing grounds for whatever equipment that could be acquired or moved throughout the semester, up until the March 12 lockdown occurred.

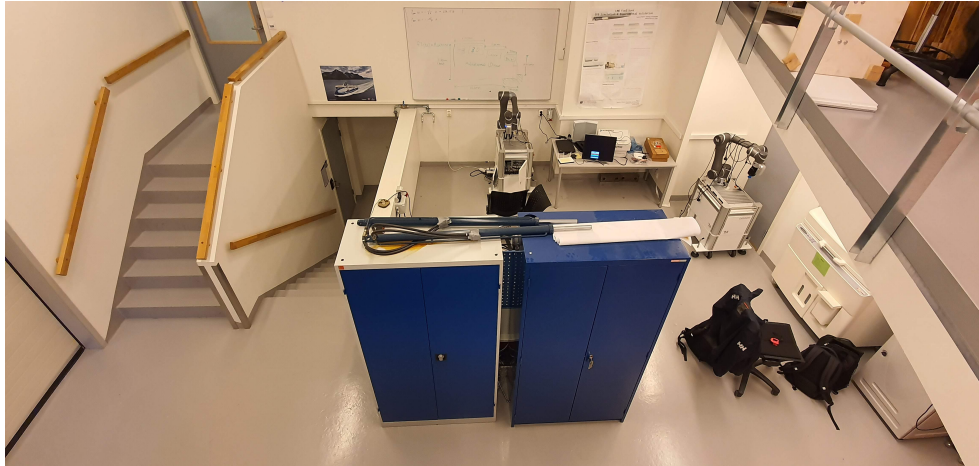


Figure 3.14: Early overhead view of room L108

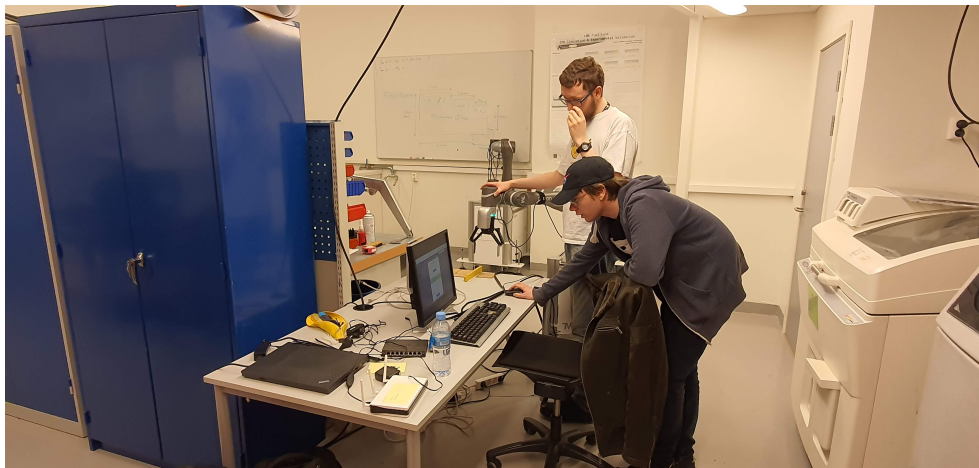


Figure 3.15: State of the test lab in February



Figure 3.16: L108 taking shape

3.3.2.1 Network and Communications Setup

Twisted pair category 5 (Cat5) cables with RJ45 plugs were made on-site and used to connect the various equipment through switches and routers borrowed from the lab faculty. By March 12, the network had taken shape as seen in figure 3.17.

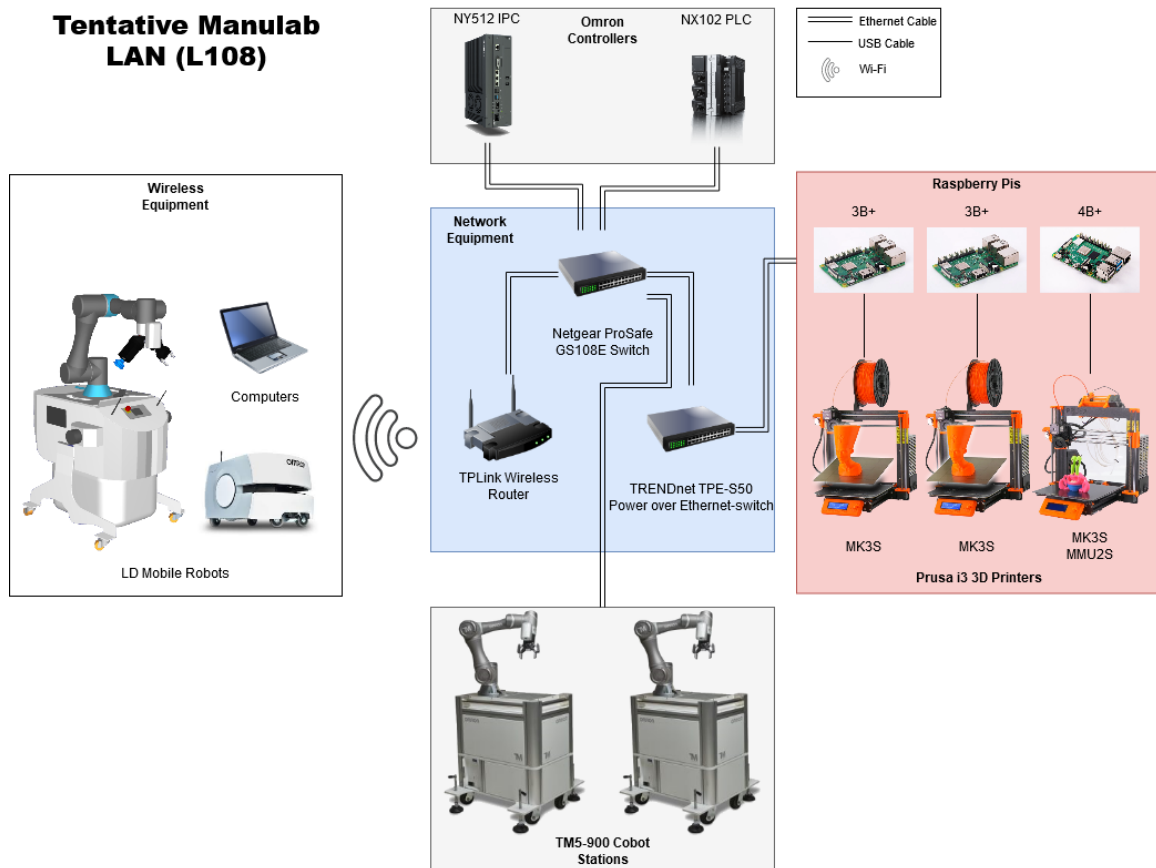


Figure 3.17: Network diagram of the LAN setup in room L108

More information about the hardware can be found in their respective sections, or summarised in section 3.1. Every connected piece of equipment was given its own static IP address, as can be seen in table 3.1.

IP Address (192.160.250.XXX)	Equipment
000	
001	Sysmac
002	Windows
003 - 006	HMI reserved
007 - 031	PLC reserved
032 - 050	Laser reserved
51	Cobot LD
052 - 070	LD robots reserved
071 - 090	Quattro Parallel Robot reserved
091 - 110	Viper Robots reserved
111 - 199	Free
200 - 240	Raspberry Pis (3D Printers)
241 - 254	Router DHCP Range

Table 3.1: Static IP address reservations for L108

3.3.2.2 TM Collaborative Robots setup

Communication between the main IPC and the TM Collaborative Robot Arms was carried out over Modbus TCP using the aforementioned Cat5 cables. Further details can be found in section 3.5.4.1.

The various RobotiQ grippers connected to the TM robots are wired to the TM control boxes' serial COM ports through a RS485 to RS232 converter^[8]. They have their own internal controllers that control the grippers using Modbus RTU. On RobotiQ's product web sites, there are packages that can be imported into TM Flow to make using the grippers a plug-and-play experience with pre-defined functions.

3.3.2.3 LD Autonomous Mobile Robots setup

Being mobile, most LD robots connected to the lab network did not require any additional wiring. Lacking both space and Omron's Enterprise Manager for fleet control, the robots in the temporary test lab had to be accessed using a household router. This limited the mobile robots fleet's capabilities. See section 3.8.3 for details.

3.3.2.4 Setup of modified LD-130CT with TM5M-900 attached

The modified LD-130CT mobile robot with a TM5M-900 cobot arm attached to it did require some wiring. Initially, we had planned for buying a wireless access point and using that to directly control the TM5M over Modbus TCP from the IPC. However, Amatec had intended for the LD-130CT to use its internal switch to relay data to the cobot through port forwarding. The ethernet cable from the LD to the TM5M had not been wired, so the robot had to be reassembled on February 25. Serial communication for the RobotiQ Hand-E gripper was also wired at this time.

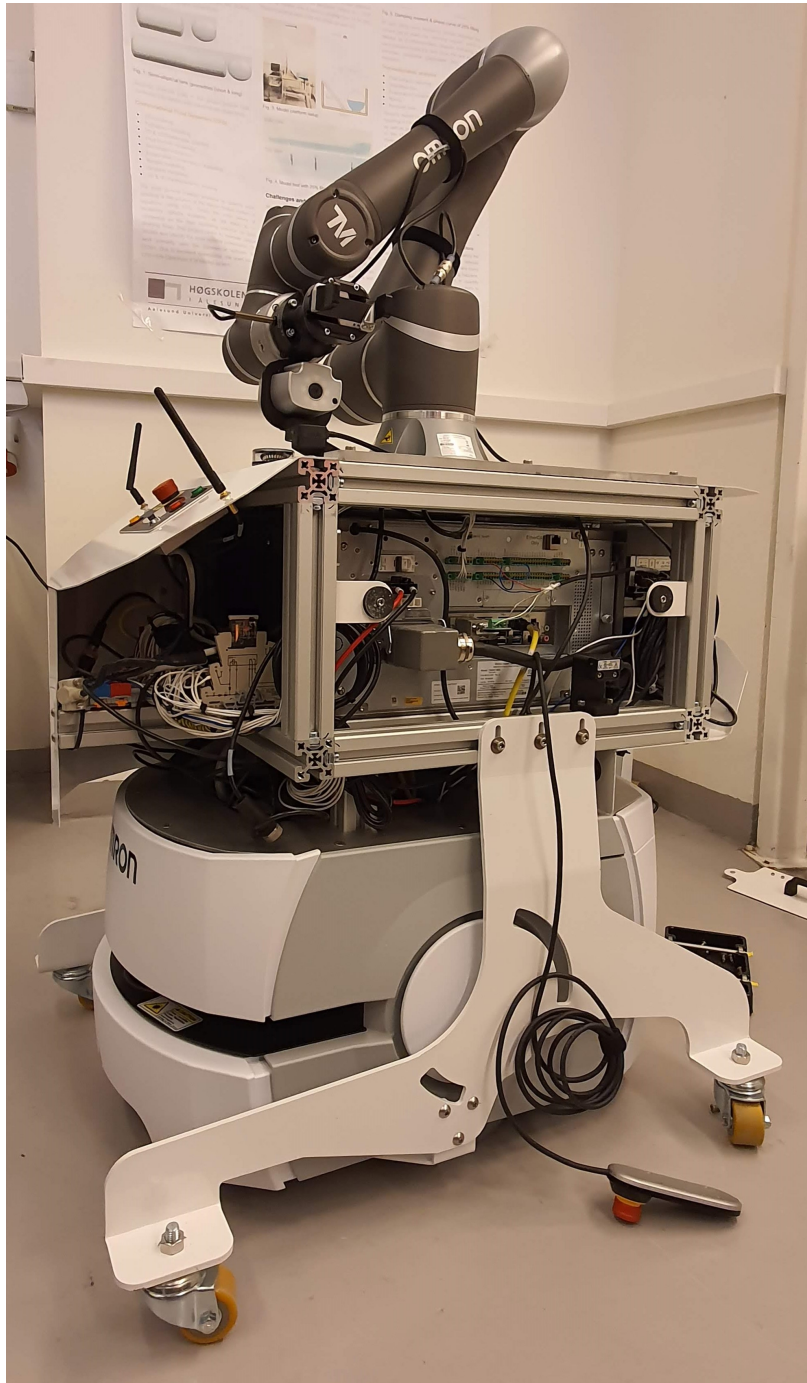


Figure 3.18: LD-130 during network wiring

For the LD-cobot internal network setup, a guide written by ATC Europe was provided by Omron and Amatec. The result of this setup can be found in section [4.5.3](#).

3.4 Product Prototypes

For a production line to be of any purpose, it needs to produce something. One of the first things we did for the project was to brainstorm several ideas for products that could be used to showcase the different machinery and stations comprising the Manulab. 3D models and CAD-files were then designed and continuously improved upon throughout the project.

3.4.1 Product 1: Desktop Name Sign

The desk name plate was the first idea for a product that utilised both 3D printing and laser cutting. Two symmetrical, 3D printed brackets work as feet, while a 4 millimetre thick acrylic glass plate is cut and engraved to fit into them. The idea behind the name plate is a simple one; it allows for visitors of the Manulab to bring home a souvenir that they can customise with their own name. The design is intended to be something a person would want to put on their desk, while also being simple enough for two robotic arms to assemble it.

The first iteration of the design was simple and had 4 parts. The name plate itself, two brackets which hold the name plate at an angle, and a supporting plate at the bottom between the two brackets. This design proved to be too loose and couldn't hold the plates together reliably, so the design of the leg pair was quickly changed.



Figure 3.19: Render of first name plate prototype

3.4.1.1 Brackets

In the second iteration of the design, the leg pairs were designed to have a hole where a plate with a fitting cut-out could simply slide in and stay in place without twisting or falling out. Upon testing, this variant of the design proved to be so stable that the support plate on the ground was made redundant, and was subsequently removed from the design to cut down on production complexity, time and resources.

Some further tweaks were then done to the leg design to improve it. Hard edges were changed to slopes to make it easier to slide the plate into the slots, the hole was lowered further into the leg structure to make it less easy for the acrylic plate's legs to break if stressed, and the curved angle on the front was made sharper to avoid the plate falling forwards when pushed. With the support plate gone, making the legs in a way that the same leg could be used for both left and right side of the plate also became possible.



Figure 3.20: The different iterations of name plate legs

Once a final design was determined, optimising the print time and material usage per printed leg was investigated. A leg with a hollow opening in the bottom part and an NTNU logo on each side was made, but upon closer inspection in PrusaSlicer it was discovered that this actually resulted in more time and filament being used per print. The cause of this appears to be that creating the internal support structure for the inner walls of the opening consumed 15 cm more filament than keeping that section of the leg solid, so this design variant was scrapped, but the NTNU logo carried over to the final design.



Figure 3.21: Discarded open bracket design

3.4.1.2 Plate

As the design for the legs went through its various iterations, the plate also needed to be redesigned to fit the new shape. In the second iteration, holes were added to the legs that would let the plate slide into the leg to provide better stability. Later, the legs were moved closer to the centre, as shown in figure 3.22, to give better support to the newest iteration of legs by letting the outside remain flat against the side of the leg, making the product look more pleasing to the eye.

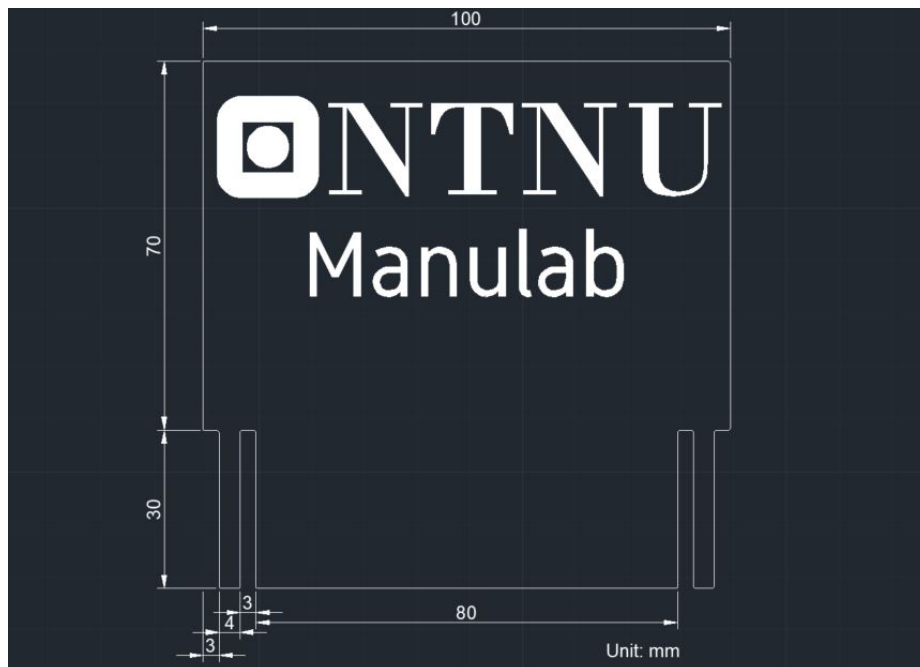


Figure 3.22: Drawing of plate without personalised name.

3.4.2 Product 2: Keychain Accessory

For a second prototype, another small and easy to assemble object was required to be made by the lab, to show the flexibility it can offer. After some brainstorming, a two-piece keychain accessory was determined to be one of the best options. The idea was to have a simple frame that a robot could then push an acrylic plate with custom name on it into.

In order to utilise as much of the lab as possible, this product design and assembly job was designed with the Quattro robot in mind, although the standard TM robots would also feasibly be capable of doing this in case the Quattro robot did not become available in time. The frame was designed to be symmetrical to simplify the assembly process. This way the acrylic piece could be inserted in any direction as long as the tag faced upwards.

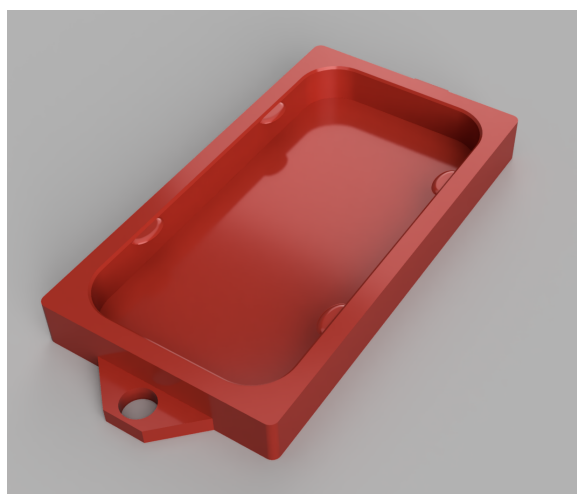


Figure 3.23: Rendering of the finished keychain frame design

Once the model was complete, a frame was 3D printed and an acrylic name tag was laser-cut and engraved. Upon initial testing, the plate was not held securely enough in place, so small overhanging edges were added to the inside of the frame to secure it in place, with slightly smaller notches added to the acrylic piece. This made it difficult to insert the name tag, but it was well-secured in its place on successful insertion.

To counter the difficult insertion process, alternative filaments were considered for the printed piece. Some research led to discovering a more springy material known as NinjaFlex^[9]. This material was both flexible and incredibly durable, and sounded promising for the keychain frame. Printing with the material did however prove troublesome, and separating it from the printing plate was quite problematic, but the finished result was a flexible keychain frame. Inserting the acrylic piece proved much easier with this material, however it was also more prone to falling out. A more semi-flexible material would therefore have been better, but there wasn't time to test this before the COVID-19 contingency measures began.

3.5 Master Computer and Program

To facilitate and handle communication between all the different robots and other parts of the Manulab, as well as managing the process and order of tasks that all the different parts of the Manulab will have to do, a central computer to act as a master was required to meet the requirements of the task at hand. This computer needed to be capable of connecting to and controlling a variety of different types of hardware, and run a program that strings it all together. An Omron NY5 IPC was determined to be used for the project, due to its integrated PLC and the capability to run Windows software ontop of it, making it very flexible for what kinds of tasks it is capable of doing.

3.5.1 Master program

In order to manage the production process while all the equipment of the Manulab is connected, a master program is required to keep track of communication, the tasks at hand, and to send out instructions in the correct order to keep all the equipment in the Manulab working as it should in the right order. The source code for this entire program can be found in appendix F.

In order to process orders made by a customer in the Movicon GUI in a reliable way, using a state-based process using IF-statements for the master program was determined to be the best course of action. The most significant reason behind this decision was to prevent issues from arising when the program would have to wait to perform the next action, which would mean everything before and after the current code would've been run again and again. This would have caused a lot of issues, and would have to be resolved with a lot of time spent in FOR- or WHILE-loops without progressing, which is generally not recommended for PLC processes. Using a state-based method would also make debugging much easier, because the state variable would show where in the program it currently was if an error occurs.

In order to set it up a simple and straightforward state-based process, the states were split up into a set of primary tasks, each of them with a sub-tasks per individual piece of equipment in the Manulab. The program starts in a setup state where it connects to all the equipment in the lab, then if no crucial equipment is missing, go to an idle state while it awaits an order from Movicon. Once an order is received, a set of states coordinating the production process is run until completion, then it loops back to idle and will either await another order, or start working on any queue that may have built up in the meantime.

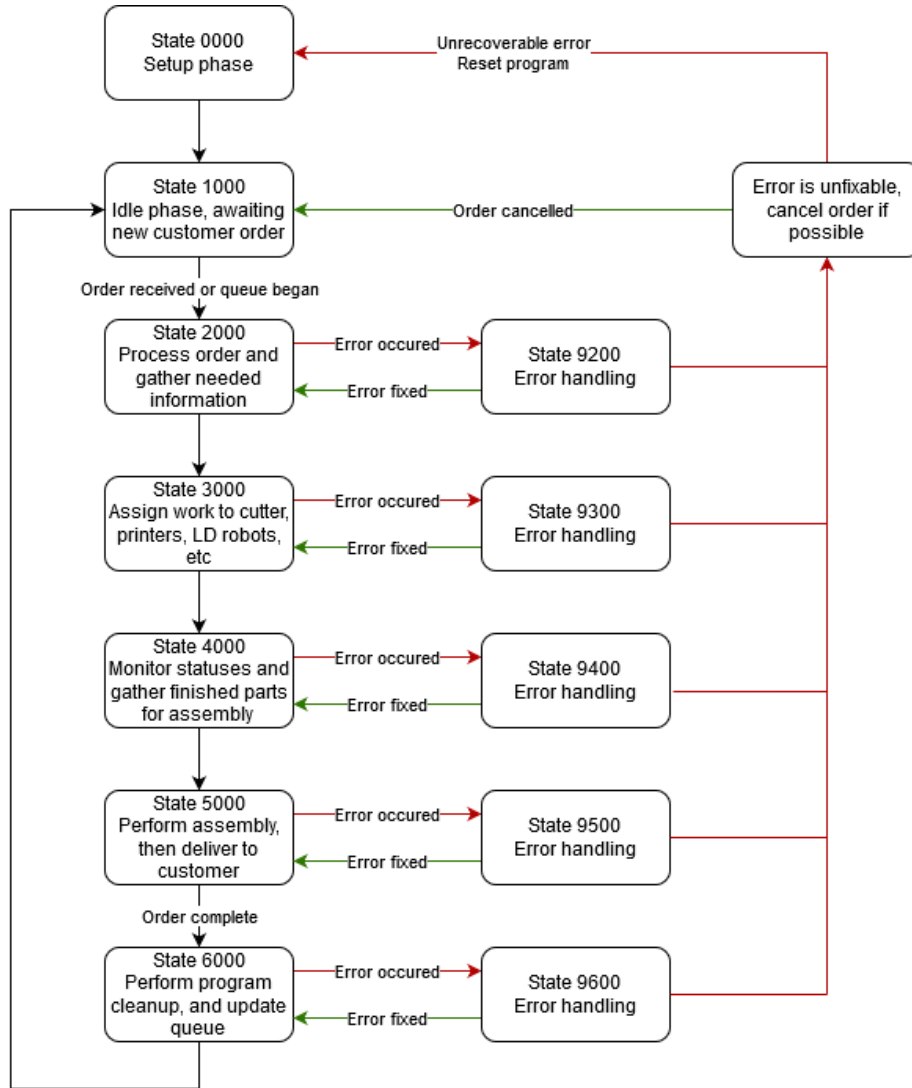


Figure 3.24: Overview of the master program process flow using states

Each main state consists of a set of sub-states, and in general the states follow the an XYZ0 pattern where X is the main process, Y is the current device being used, and Z are sub-steps for that device. For example, main state 3000 consists of four sub-states; 3000, 3100, 3200 and 3300. The initial 3000 state handles any necessary setup for the rest of the whole 3000 main state. 3100 handles any work related to the laser cutter. 3200 handles work related to the 3D printers, and 3300 handles tasks related to the LD robots. Each of these have sub-states of their own where tasks are performed one at a time, in order to prevent problems when running the given state. 3110 is the first step of 3100, 3120 is the second, and so on.

At the end of the program, error handling is performed for each individual state at any given point where it could occur, generally following a 9XYZ pattern that reflects where the error occurred. If an error occurred in step 2210, then the error-handling is done in 9221. This includes handling invalid product IDs, if the process of connecting

to any machine in the lab fails, parts not being in stock, and any other potential issue that could arise.

3.5.1.1 Queue system

The master program also needed to be able to handle a queue system if a new order was received while the program was busy processing another order. When the master is then done processing the current order and returns to idle, it can start working through the queue if there is one. There are a few ways that this could be done, but an array containing order information seemed to be the simplest way to accomplish this.

Whenever a new order is received from Movicon and starts processing, the order info is copied to a local variable and reset, freeing the original variable up for new orders immediately while the production processes data from the local variable. The queue system then works on the side independently from the master state by checking for new orders (`customerOrder.orderReceived = TRUE`) while the program is busy processing another request. If a new order is received while the program is not in its idle state, it's copied to the first available spot in the queue array, behind any other queued order.

Once the program has finished processing the original order and goes back to Idle (state 1000), it checks the queue. If an order exists in the first spot in the queue array, it is then processed. When the queued order is completed, all orders in the queue are moved down one spot in the array before the program goes back to Idle and checks again. As such, the oldest order is always the first to be processed. The queued order is not removed from the array until the order is finished in case something fails and the program has to start processing again.

```
86 // Add order to the back of the queue when a new order is received while the program is busy processing another.
87 // Reset customerOrder after completing to open up for a new order if one comes
88 // Note, this always runs in the background.
89 IF NOT(masterState = 1000) AND customerOrder.orderReceived THEN
90     customerQueueLength := customerQueueLength + 1;
91     customerQueue[customerQueueLength] := customerOrder;
92     customerOrder.orderReceived := FALSE;
93     customerOrder.productID := 0;
94     customerOrder.name := "";
95 END_IF;
```

Figure 3.25: Code for adding a new order to queue

```

330 IF masterState = 6050 THEN
331   //If the current process was started from a queue, then move all spots in the queue down one level and reset the last.
332   FOR index := 1 TO customerQueueLength-1 DO
333     customerQueue[index] := customerQueue[index+1];
334   END_FOR;
335   customerQueue[customerQueueLength].orderReceived := FALSE;
336   customerQueue[customerQueueLength].productID := 0;
337   customerQueue[customerQueueLength].name := "";
338   customerQueueLength := customerQueueLength - 1;
339   startedFromQueue := FALSE;
340
341   //Queue process complete, continue with order cleanup
342   masterState := 6100;
343 END_IF;

```

Figure 3.26: Code for removing order from queue

3.5.2 Background processes

After a while of coding, the main master program started to get fairly cluttered with both states and stateless tasks in the same program. To help reduce this and decrease cycle time in the master program file, a separate program file for background processes was introduced to the main program. This background program stores all processes that need to be periodically done regardless of current state that the master is in, and runs at a reduced cycle time to reduce strain on the PLC. The tasks this program is responsible for are:

- The status of all connected robots, printers and cutters in the Manulab are read and updated. From here, they are then processed and transmitted to Movicon to update the status panel with the latest information. This information is transferred via the `moviconSendStatus` global variable.
- The list of currently active prints is updated by iterating over the status array for all printers, checking for printers that are either currently printing or in the finished state, and then processing the gcode file name with the `ProcessGcode-Name` function to get values for product ID and number of parts being printed in each individual printer.
- With all printer statuses updated, the program then checks for printers showing the 'finished' status. If one is found, it's retrieved by an LD robot, added to kanban storage and the current number of finished parts and parts being printed are updated.
- The kanban buffer set by an operator in Movicon is checked for out-of-range values. Any buffer outside a range of 0-20 is either rounded up to 0 or down to 20.
- Current number of the different types of parts in kanban storage is checked and compared to the current buffer size for each individual part set by the operator in Movicon. If the current total of parts is smaller than the buffer, then prints are ordered until the added total of parts in storage and parts being printed is equal to the buffer size for the given part.

- Iterate over printer status array to see if either the printer bed or printer nozzle has reached a temperature that is too high. If so, the printer is told to make an emergency shutdown. This is done as a security measure to prevent any potential hacker from damaging the equipment by ordering it to reach too high temperatures.
- Periodically reset a few variables and functions.

3.5.3 LD functions

In order to coordinate the LD robots with the rest of the lab, and especially between the LD-130CT and the TM5M-900 on its back, some logic needs to be put in the central program. The LDs need to be told to do specific jobs at specific times, or wait for ready signals such as when the TM robot arms are finished performing their jobs and have placed the object on a conveyor LD for delivery.

In order to communicate between Sysmac Studio and the LD robots, a library of functions and function blocks developed in-house and supplied by Omron Support was used. This "ARCL" library, short for Advanced Robotics Control Language, contains function blocks for connecting to LD robots, assigning tasks, docking to recharge the battery, making robots perform certain actions like playing audio files or following a person, and more. The functions available in the library are the primary source of assigning jobs for the LD robots.

These functions are contained in a series of ladder programs called commsLD, with commands and information being transferred between it and other parts of the program by using the LDCommands global variable.

3.5.3.1 Connecting to LD robots

By using the ARCL.Connect function block, the LD robots are connected to the main program in Sysmac Studio by inserting the IP address of the robot, the port, password and connection ID. Seeing as the delays for the lab prevented access to the Enterprise Manager used for fleet control, and thus prevented testing with it, the program instead connects directly to each individual LD robot via their individual IP addresses rather than just the Enterprise Manager as this was the only thing that was possible to test before the COVID-19 outbreak.

In order to be able to complete all the tasks required by the lab, the LD-130CT with the TM5M-900 arm is required to be connected and functional, and at least one LD-90 with conveyor too. If either the LD-130CT or both of the LD-90s fail to connect, an error signal is sent back to the master program to report that the LDs were not successfully connected, forcing the program into an error state.

3.5.3.2 LD tasks

The LDs need to perform a handful of work activities throughout the lab, mostly related to transport. There are 4 of these tasks run in the main program, each of them triggered with a bool variable for start located in a global struct that is used to store LD commands, aptly named LDCommands. Once a task has been completed, a second bool variable is turned true to signify to the rest of the program that the task has been completed. The tasks are as follows:

getFromPrinter Navigate to the specified printer by retrieving the X and Y position of the printer from its information. The LD-130CT is then given the X position so it can drive up to the correct column of printers on the rack, as well as giving the TM5M arm the Y position to raise the up to the correct row. While the TM5M arm is picking the part and printing board off the printer, a second LD-90 with a conveyor is then told to pull up behind the LD-130CT, which the arm then places the printer plate and part on. Seeing as the TM arms are unable to separate the prints from the printer plates themselves, the part and printer plate is then delivered to a working station where a human separates the part before putting it into the kanban storage.

getFromKanban Depending on the part needed, the LD-130CT drives up to the kanban storage near the needed part, the TM5M picks the part up after being given the product ID, then it drives to the assembly table and drops off the part within view range of the TM5 arms used for assembly.

getFromCutter The LD-130CT robot drives up to the cutter, the TM5M opens the lid, picks up the required part using either a vacuum gripper or a standard gripper, then carries it to assembly. This may or may not require a second LD equipped with a conveyor depending on the size of the item to pick up, but cannot be tested due to lack of access to hardware.

deliverProduct The finished product is picked up by the LD-130CT, then carried over to an area where finished products are placed. The TM5M arm then places the finished product there.

It is worth noting that some of these tasks may or may not need a few changes depending on how they would perform during testing. Such as replacing the LD-130CT's tasks with an LD-90 in some cases.

3.5.4 TM functions

Like with the LD robots, connecting to and using the TM robot arms also required a library of function blocks supplied by Omron Support, known as the OEN Toolbox. The IPC that the program was initially developed on was incompatible with this library due to outdated firmware, as described in chapter 5.1.4, but by using a second NX102 connected to the IPC, testing was possible. Once finished, the IPC used in the actual Manulab would be compatible with this library.

Like with the LD robot, the functions for communicating with the TM robots are contained in the ladder program called `commsTM`, while information and commands are transferred between it and the rest of the program with the `TMCommands` global variable.

3.5.4.1 Connecting to TM cobots

Connecting to- and communicating with the TM cobots is done via a series of function blocks for setting up Modbus TCP communication. In order to do this, Omron supplied a pre-made program and a toolbox of functions used for communicating with a variety of their robots and equipment. As previously mentioned, this did not work on the main IPC, but testing on a second NX102 was successful and communication with one of the TM robot arms was established, allowing for some basic testing.

This process works by setting up a structure for each individual TM arm and filling in configuration information, such as IP addresses, ports and which of the various status variables to transfer. After this is done, the 3 different TM arms are connected to the PLC program via Modbus TCP, and upon completion, a "connection successful" bool is set to TRUE. If all 3 arms return the connection successful signal within 30 seconds, the process reports back to the main program that all arms are connected so the setup process can continue. If not, and not all are connected within 30 seconds, connection error is returned instead.

3.5.4.2 TM tasks

Once connection is established, the process of telling the TM cobots which tasks to do from Sysmac is a fairly straightforward process. There are 3 separate functions in the `doTask` process, one for each TM cobot with corresponding values; `armLD`, `arm1` and `arm2`. Separate functions are used to prevent conflicting variables being used.

When a particular job needs to be done by a particular robot, a corresponding "job-Start" bool for a given TM cobot is set to TRUE to start the function, an integer with a job number is given, and if required, the product ID for the current order. The given job number is transferred to the robot via a variable that goes through Modbus directly to the TM robot, which determines which part of the program on the robot is run. Upon completing the job, a bool value is returned from the robot to Sysmac as TRUE, which then leads to the function returning a "jobDone" boolean, set to TRUE, to the master program to signify that the job has been completed and that the production process can continue.



Figure 3.27: doTask function in the main program, 2 lines per TM robot arm

3.5.5 Communication with printers and cutter

With the Manulab consisting of many parts and components not normally built to communicate with a PLC, parts of the communication could not be done through normal inputs and outputs in Sysmac like the robots can. To achieve communication here, alternative methods had to be used. Two candidates were eventually determined to be the best course of action; either using UDP to perform communication directly, or use a Python script in combination with expanding upon the CSV process that was already used by the DXF script to extract and insert information to create cutter jobs with custom names engraved on them.

While UDP was the best option of the two for proper communication, CSVs were picked as the temporary solution in the start for a variety of reasons.

- As mentioned, the framework for using CSV files to transmit data was already in place for the DXF script, and as such it was rather quick and simple to set it up for data transmission for other devices such as printers. This allowed testing with communication what was available at the time to begin almost momentarily.
- Octoprint sends information about printers as JSON objects, and attempting to parse these in the PLC program would quickly become a lengthy and very complicated process according to an advisor from Omron.
- Parsing JSONs is much more easily handled by a python script -like the one described in chapter 3.6.3- which runs locally on the IPC; and using local CSV files was the easiest method we were able to find to transfer information between Sysmac and Python scripts.
- TCP was off the table due to unreliable support for it in Sysmac Studio according to the same advisor.

While a long-term plan was to eventually replace the CSV process with a proper UDP communication protocol and JSON parsing directly in the PLC program, other tasks were continuously prioritised over performing this rework, especially since the CSV method worked well enough, reflecting the Minimum Viable Product role of the project. As a result, this change never happened. A UDP socket program was however included in the program as a starting point for future expansion, but due to the aforementioned difficulty of performing JSON interpreting in Sysmac, this was never used.

In order to read and write information to the CSV files, the program uses a set of functions that are used to read and write files stored on an SD memory card. While this would be a problem on a standard PLC, where this kind of communication would not be usable due to having to physically move the SD card from location to location, the Industrial PC does not have this flaw. It instead has a folder locally on the Windows portion of the IPC used as a virtual SD card, which the PLC portion of it accesses instead. This is then within reach of the Python scripts that either read from or write to the file when doing JSON parsing. These scripts are however impossible to start from the PLC itself, but Movicon has the capability to do so. As such, when a Python script needs to be started to perform its task, a command is sent by the PLC to Movicon, using the `moviconSendCommand` structure variables, telling it to perform this action.

All of the functions listed below are contained in the `commsCSV` ladder program, with start and done signals being transmitted via the `csvBools` global variable.

3.5.5.1 Write DXF for cutting

In order to create DXF files for cutting that allow for user customisable names on the plate or keychain, the Python script requires to read information from somewhere. Different methods for solving the problem of how to transfer information from Sysmac to a Python script was investigated, and in the end, it was determined that the simplest course of action is using Sysmac to write to a CSV file on a virtual SD Card stored on the IPC, which the script can access.

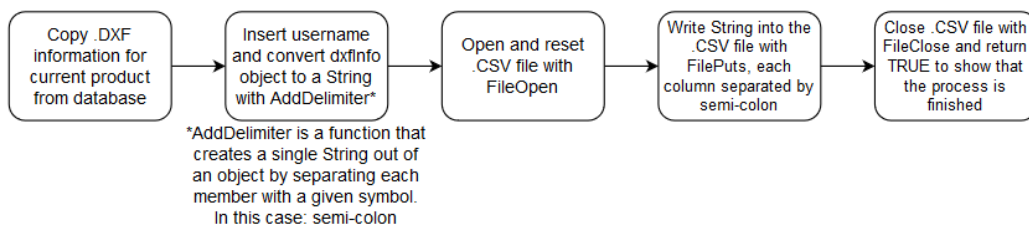


Figure 3.28: Overview of the process to write information for the DXF script.

After the process is complete, a signal is sent to Movicon to start the Python script, because Sysmac is unable to do this itself.

It's worth noting that due to never getting access to the controller that would interface with the laser cutter, it was impossible to determine how communicating with this would be done, and thus a function for writing directly to the cutter was never made. A process similar to reading and writing to printers would most likely have been used if it was available though.

3.5.5.2 Write command to 3D printer

As previously mentioned, the easiest and quickest course of action to communicate with hardware that couldn't normally connect to a PLC was determined to be using Python scripts that read and write information from CSVs handled by the PLC program. The most noteworthy case of this was communicating with the 3D printers, because Octoprint writes JSONs that are troublesome to parse in Sysmac, but simple to handle in Python. The process for writing commands to the printers for the Python script is as follows:

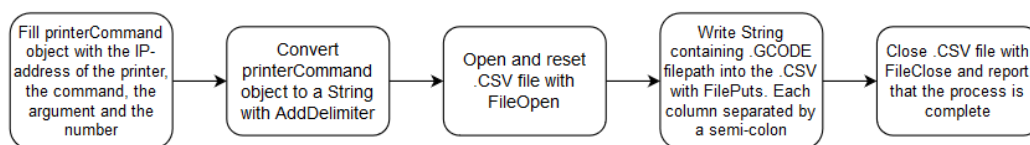


Figure 3.29: Overview of the process to write commands to the printer.

Like with the Write DXF function, a signal is then sent to Movicon telling it to start the Python script to handle communication to and from

The printerCommand structure that is used supports a few commands, but more can be added in the future.

print Prints a 3D model, the most common command. Argument is used for the filepath of the .GCODE file containing the 3D model of the given product (which is retrieved from a database), and Number is used for the number of the parts being printed.

printFinished Sent to the printer after the print has been picked up to tell it that it's ready to be used for new prints. No argument or number used.

shutdown Sent to the printer to force a shutdown if a dangerous scenario happens, such as nozzle temperature or printing bed temperature getting too high as a result of a malfunction, or an accidental or malicious attempt to harm the printer from overheating.

3.5.5.3 Read cutter status

With the write functions for printers and cutter as a working proof of concept for using CSVs together Python for communication, using the same framework for reading status was investigated as well to keep one standard for communicating with these

devices throughout the program. After some trial and error, it was proven to work just as well as reading, but the process was more complicated due to the fact that the function will only read from a given starting point and until the end of the line.

After opening the file, a starting point has to be set before Sysmac would read the current line, represented by a byte offset in the input file. Because the text prior to the line where the required information is stored has a fixed length, this byte offset was set so that the starting point is at the beginning of the third line in the CSV. A function block is then used which reads the rest of the line until newline, saves it as a string, and then converts the string to a cutterStatus structure using the SubDelimiter function. Lastly, a boolean value is set to true to show that the read was completed successfully and that the program can continue to run from there.

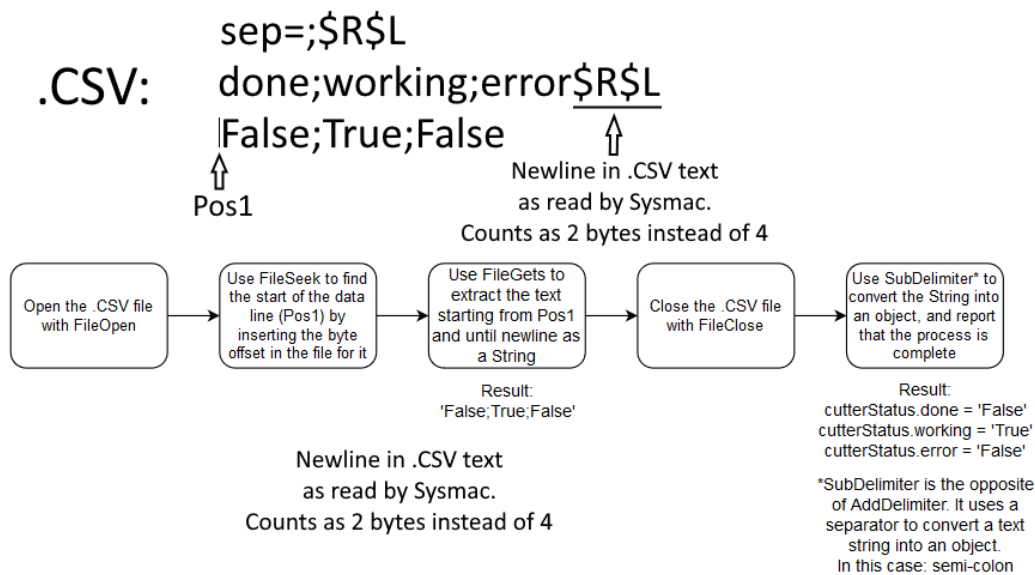


Figure 3.30: Overview of the process to read cutter status.

3.5.5.4 Read 3D printer status

Reading status information from all the printers quickly became the most complicated part to create for this type of data transmission. The status information was given across several rows of text, each row corresponding to one printer, each rowing having varying length, and similarly to the code used to read the cutter status, the function block used to read CSV files can only read one line at a time, starting at a pre-determined position in the file based on an offset variable. To account for the 20 or so printers in the Manulab, an array of printerStatus objects has to be used to store the status of each individual printer to avoid variable clutter.

To read information to add to this array, the program goes to the pre-determined starting position set by the offset variable where the first row with information begins, before calling a ReadLine function block that extracts the text from the starting position and until newline into a string output. The length of the string is then

determined and added to the offset, so that the next ReadLine function will begin at the start of the next row. Once this is done, the line is cut up using SubDelimiter, where each respective value are assigned to the appropriate variable in the struct. Because of how Sysmac reads newlines from the CSV, a \$R\$L is automatically added to the end of each row, and thus at the end of the last object in each printer status object. To avoid issues, this newline text is removed from the end of the last variable in the structure. After all this is done, an index value is increased by one and the process starts over to read the next line into the next array spot. This repeats until the index becomes larger than the number of elements in the array. Finally, a boolean value is toggled to show that the read is complete and successful.

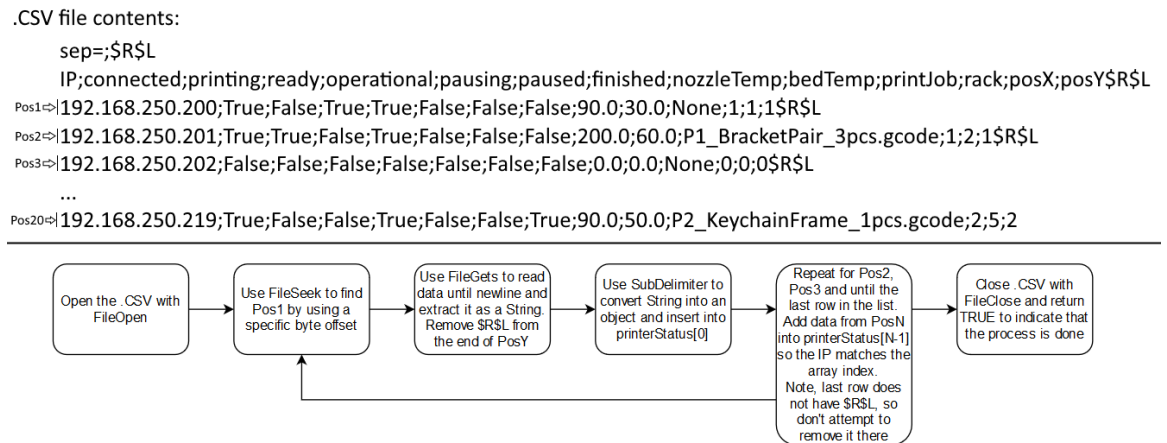


Figure 3.31: Overview of the process to read printer status.

3.5.6 Functions and function blocks

As the program started to take shape, it became apparent that there was a lot of reused code throughout the program. To ensure that changes were easier, to reduce the overall complexity and to improve the general quality of the code, the most reused pieces of code were put in separate functions where possible.

3.5.6.1 FindAvailablePrinter

One particular piece of code that became very frequently used as the program was being made is iterating over the printer status array to find an available printer to use. For the sake of simplicity and to ensure the earlier points are followed, this was put into its own function, with the capability of returning a -1 if no printer is available.

It works by having a FOR loop iterate over the array containing the status of all printers, and finds the first printer in the array that is both listed as ready to accept a new print, and isn't in the finished state where the object hasn't been retrieved from the printer yet. The printer number in the array is then returned by the function. Should no available printer be found, the function returns a -1, and it is thus advised

to follow usage of the function with an IF statement to cover what to do if there is and isn't an available printer.

3.5.6.2 ProcessGcodeName

To avoid hardcoding several IF checks throughout the program to determine what printer is printing what, a function had to be introduced to automatically process the name of a .GCODE file to extract product ID and the number of parts being printed at the specified 3D printer. This does come with the downside of requiring the file names to follow a specific, pre-determined format, and thus needing an IF statement check to confirm this, but it would enable it to handle any value and length of both product ID and number of parts as long as the .GCODE name follows the format. If it doesn't, -1 is returned to signify that something is being printed but what it is can't be determined.

Before running the function itself, the name from the input is checked to see if it matches the expected file name format of 'P(ID)_ProductName_(number)pcs.gcode'. In order for the function to accept the name, it needs to fulfil the criteria below. If it fails at any one of them, "-1;-1" is returned because the function will be unable to retrieve the required information from the name.

- First letter is "P"
- Last 9 letters are "pcs.gcode"
- The name has no more and no less than 2 underscores

If the name passes the check and is accepted, the function starts processing the gcode name to retrieve ID and number of parts being printed, following the process below.

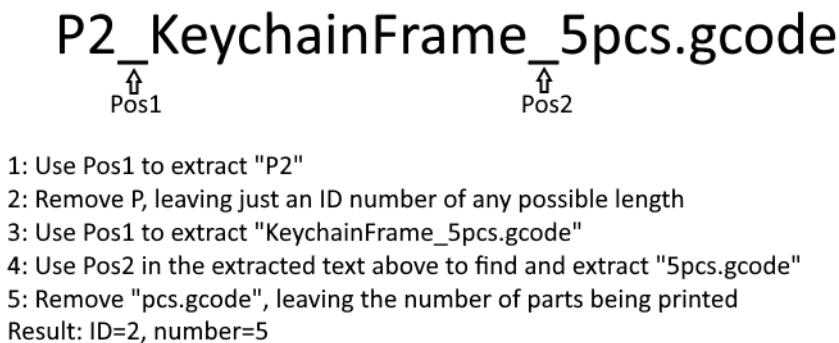


Figure 3.32: Gcode-name interpreter function

Once the ID number and number of parts being printed have been extracted from the name, the extracted numbers are run through two more checks to ensure they are actually numbers before being returned and converted to integers.

- Check for letters by seeing if there is any difference in the string when changed to uppercase and lower case. If there are letters, the ID and/or number is set to -1.
- Check for commas and periods. If any are found, they're removed from the number before the process continues.

Once all this is done, the ID and number are concatenated into a "(ID);(number)" string and returned by the function. While outputting them as separate variables in a structure would be preferred, a limitation with Sysmac prevents functions from returning structures and arrays. Instead, the returned string can be run through a SubDelimiter function to be converted into a structure.

3.6 3D Printing

3D printers are an integral part of the Manulab, allowing for additive manufacturing of parts and products using various filaments. As mentioned in section 3.1 the lab houses several Prusa i3 MK3S 3D printers, which are placed in two two-story racks. These printers print on detachable magnetic steel sheets, and can be connected to other equipment via USB. One of the main tasks of this thesis was to find a way to incorporate these 3D printers into the lab's greater whole. To accomplish this, the following problems needed to be addressed:

- Calibrating and setting up the printers as needed
- Testing filaments and modifications to see what is suited for automatisation
- Connecting the printers to the lab network
- Adding necessary functionality for the printers to work with the IPC
- Finding a method to transport and replace printer steel sheets automatically

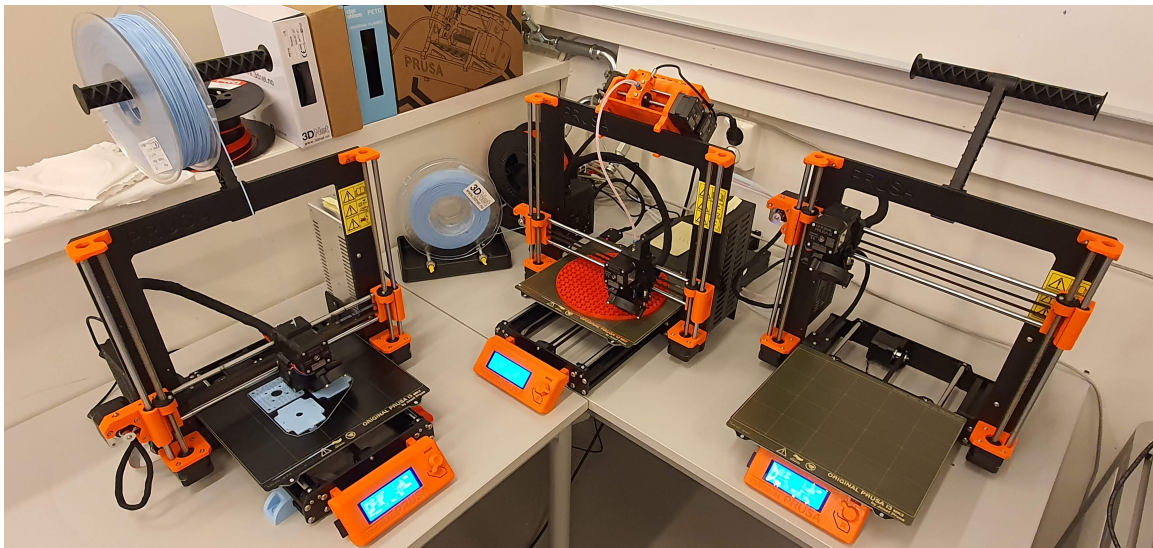


Figure 3.33: 3D printers for testing in L108

The Prusa i3 MK3S contains an Einsy RAMBo printed circuit board, which handles the printer's motor and temperature controls among other things. Beyond parsing G-code and controlling the printing process, it has limited functionality. To make the printers more versatile, it was decided that Raspberry Pi single-board computers would be used to control and connect them to the lab network. In addition to the Prusa printers, Raspberry Pi 4 Model B's had been ordered, but not yet arrived when work on the thesis began. To get around this, early testing was done on the university's Ultimaker 2+ printers, and a Prusa MK3S and Pi model 3B+'s were borrowed from supervisors.

3.6.1 Mechanical setup, testing and modification

Before working on various software solutions, the 3D printers themselves had to be operating smoothly. On January 22, the first Prusa i3 MK3S printer was borrowed, then two more arrived on February 6, pre-calibrated from Prusa's Lab in the Czech Republic. Alongside the latter two, the university had ordered a *Multi-Material 2S Upgrade kit* (MMU2S) for our team to assemble and test.

3.6.1.1 Multi-Material 2S Upgrade

The MMU2S is an upgrade for the Prusa i3 MK3S that allows for printing with up to 5 rolls of filament at the same time. Primarily intended for printing using multiple colours, it can also be used for interesting applications like printing water soluble supports. It uses a filament spool buffer and a pulley system to feed the filament through teflon tubes to the extruder. To assemble it, the extruder needed to be reassembled and the PCB rewired. Including basic calibration, this took just short of two days while following the web manual^[10].

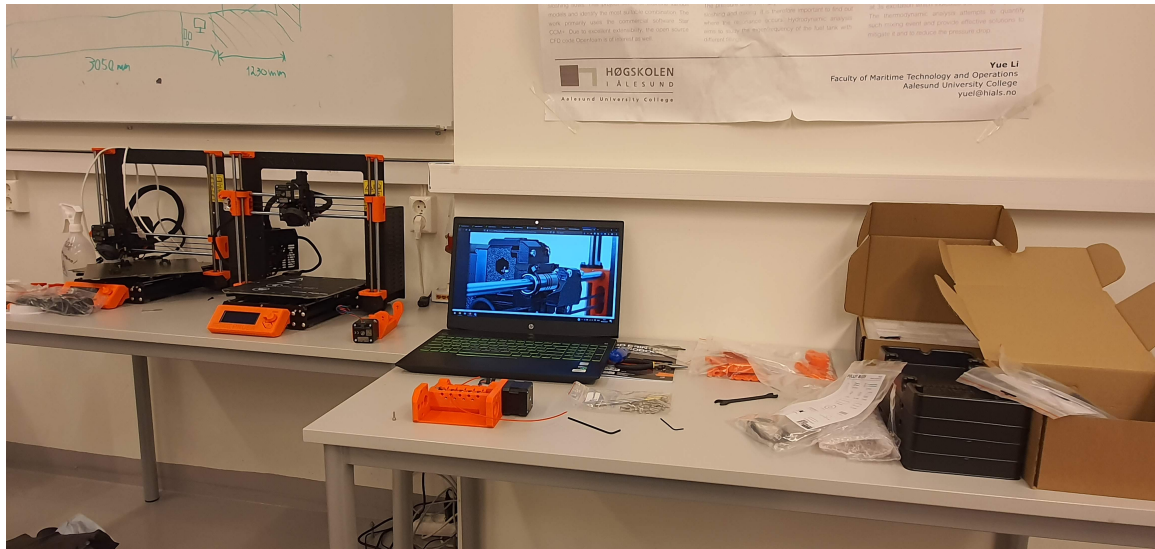


Figure 3.34: MMU2S unit during assembly

3.6.1.2 Calibration

The borrowed printer had some fundamental issues out of the box, assumed to be due to problems during transportation. While testing the printer using PLA filaments, the layer height seemed to result in adhesion problems and warping occurred frequently. The extruder would also get close enough to prod the heatbed during XYZ-calibration and mesh bed leveling. To counter this, the magnetic P.I.N.D.A. probe sensor was adjusted a bit lower, making the height difference between the extruder nozzle and the probe around the width of a zip-tie. The printer was then calibrated according to official guidelines^[11], and after running the first layer calibration a few times, a

sweet spot was found. It was now printing around as reliably as the factory-calibrated printers.

The MMU2S-enabled printer also required recalibration after assembly. It also introduced new problems that are further detailed in the Results section 4.3.2.

3.6.1.3 Slicing: converting 3D models to G-code

As mentioned in section 3.2, PrusaSlicer was used to convert three dimensional geometries into code that can be understood by the printer. It provides a simple 3D GUI where the user can import 3D objects of certain file types, then arrange them and slice them into G-code. Custom supports and various thermal and mechanical settings can also be adjusted to fine-tune the printing process depending on the material used.

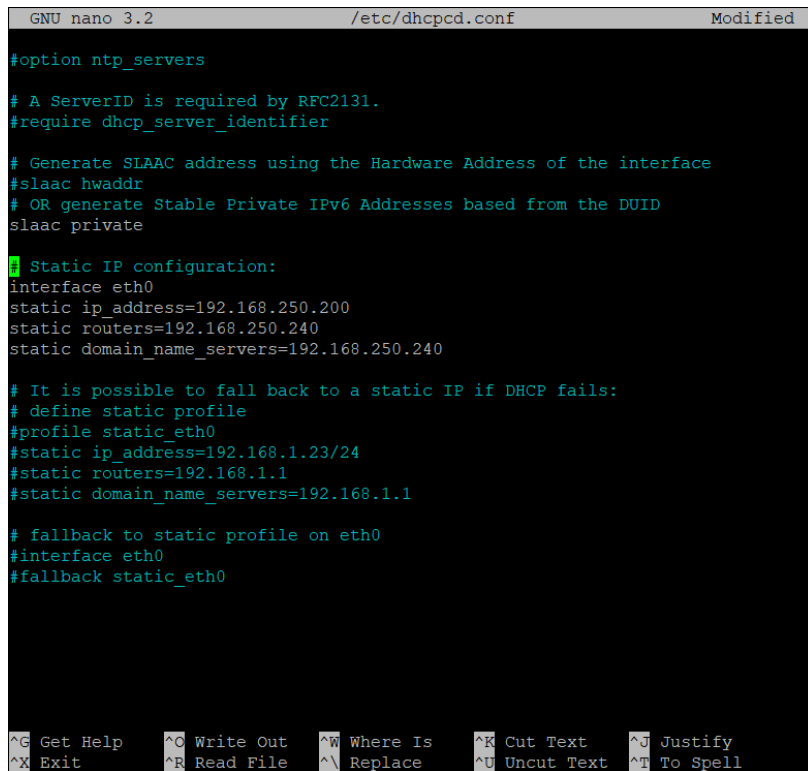
For most of the prints made during the project, the Prusament PLA and PETG presets provided good results. When working with supports or printing with flexible filaments, custom settings were added depending on the part's geometry.

3.6.2 Communication and network setup

Each printer is assigned its own Raspberry Pi. Each Pi interfaces with its printer via USB and connects to the network through an RJ45 ethernet port. To control the printers, the open source application OctoPrint was installed on the Pis. It provides a lot of extra functionality, including a web interface, custom G-code and webcam support. By using the Windows application balenaEtcher, we were able to flash a Linux distribution called OctoPi onto the Pis' SD cards. It is based on Raspbian and comes with a pre-configured version of OctoPrint.

To be able to address the printers separately, each Pi has been given its own static IP address. To do this requires some modification on the Linux side of things. Using a program called PuTTY, the system was accessed remotely from a PC in a Secure Shell (SSH) by doing the following:

1. Identify the dynamic IP address given to the Pi by the router
2. Enter said IP address into PuTTY and log into OctoPi
3. Access the IP config file using `sudo nano /etc/dhcpd.conf`
4. Comment the DHCP-related sections and uncomment the *interface eth0* part
5. Enter the printer's desired IP address as well as the router's. Make sure that all equipment is on the same subnet as the IPC.



```
GNU nano 3.2 /etc/dhcpd.conf Modified
#option ntp_servers

# A ServerID is required by RFC2131.
#require dhcp_server_identifier

# Generate SLAAC address using the Hardware Address of the interface
#slaac hwaddr
# OR generate Stable Private IPv6 Addresses based from the DUID
slaac private

# Static IP configuration:
interface eth0
static ip_address=192.168.250.200
static routers=192.168.250.240
static domain_name_servers=192.168.250.240

# It is possible to fall back to a static IP if DHCP fails:
# define static profile
#profile static_eth0
#static ip_address=192.168.1.23/24
#static routers=192.168.1.1
#static domain_name_servers=192.168.1.1

# fallback to static profile on eth0
#interface eth0
#fallback static_eth0

^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify
^X Exit         ^R Read File    ^\ Replace      ^U Uncut Text   ^T To Spell
```

Figure 3.35: Screenshot showing static IP settings on Raspberry Pi using SSH

In the temporary lab that was set up in room L108, the Pis were wired to a network switch connected to the main router. The IP range for the Pis were set outside of the router’s DHCP range. See section 3.3.2.1 for more details about the wider LAN setup. An alternative approach would be to reserve the addresses in the router’s settings, but this approach is not recommended as results could vary depending on the hardware used.

3.6.2.1 OctoPrint

Communication with the network-connected instances of OctoPi is carried out over HTTP, similarly to a common web page. When entering the Pi’s IP address or host name into a web browser, a graphical user interface is loaded. From here, users have numerous tools at their disposal for controlling and monitoring the printer. Access levels can be set up for security and the connected printer’s specifications entered for safety, among other things.

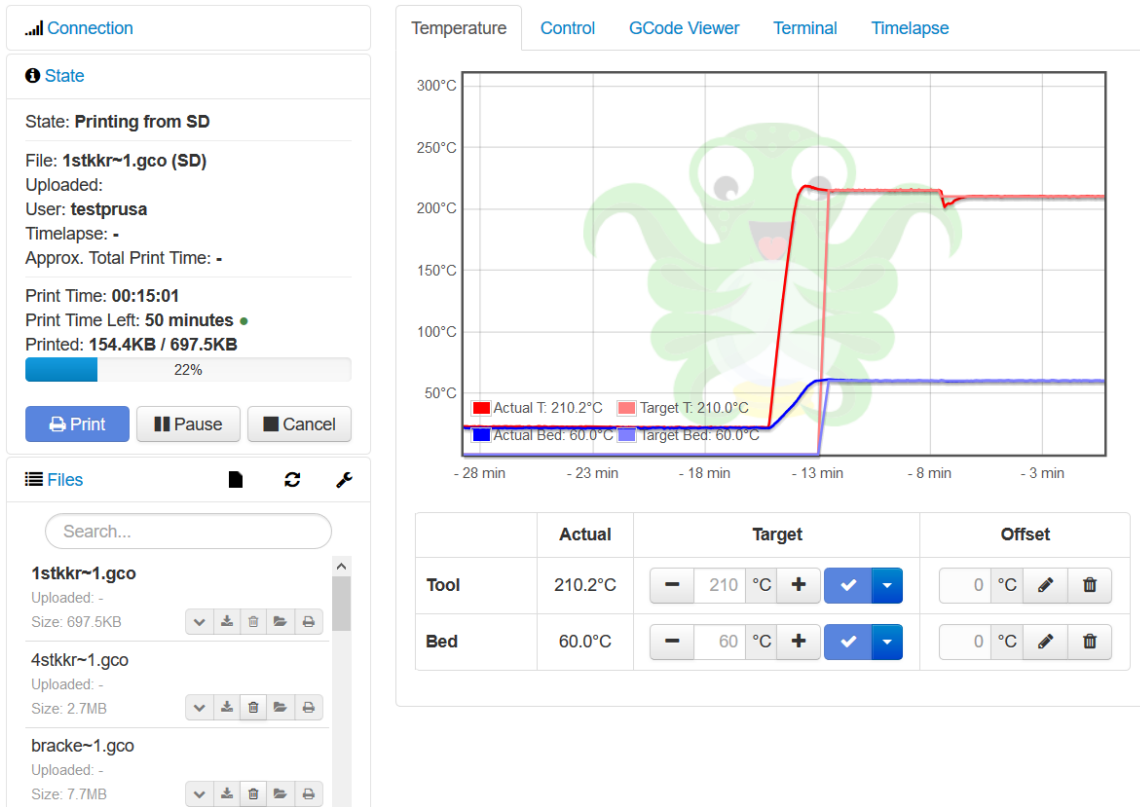


Figure 3.36: The OctoPrint web client’s graphic interface

Profiles for the Prusa i3 MK3 were added in the settings, as they are identical to the MK3S. Said profile was documented by Gina Häußge, AKA foosel, the creator of OctoPrint. [12]

Model	Form Factor	Origin	Heated Bed	Width	Depth	Height
Prusa i3 MK3	rectangular	lower left	yes	250mm	210mm	200mm
Custom Bound-ing Box	Max X	Max Y	Max Z	E	Nozzle	Extruders
X: 0/250 Y: -4/210 Z: 0/200	6000 mm/min	6000 mm/min	200 mm/min	300 mm/min	0.4 mm	1

Table 3.2: Octoprint Profile specifications used for the Prusa MK3S

In the OctoPrint settings, there is also support for adding custom G-code scripts that are run at certain conditions. One of these are a script running upon the completion of a print job. For the TM5M-900 robotic arm to be able to extract the printer’s steel sheet, custom code was added to raise the extruder and move the heatedbed outwards. This is effectively done using one line of code for positioning:

```
G90
G1 X10 Z170 Y200 F2000.0
```

Note that these values are possibly not final. Being a numerical language, the different "G-codes" correspond to different functions. G90 sets the coordinate system to absolute values, while G1 drives the axes X,Y & Z to the specified point at feedrate F. Since G-code was initially created for CNC machines, feed rate corresponds to speed in this case.

3.6.2.2 Scrapped OctoPrint-IPC communication solutions

The functions detailed thus far shows only a small part of what OctoPrint brings to the table in terms of functionality. However, it doesn't solve the main challenge of integrating the 3D printers into the Manulab. For the lab to run autonomously, the main IPC needs to be able to communicate with the printers on its own. Several approaches were considered regarding how to approach this. In the beginning, there were plans for creating Windows macros that emulated mouse clicks to start printing through the OctoPrint web client. This was undesirable, as it would've resulted in an unstable and inflexible solution.

3.6.2.3 cURL

Further researching ways to communicate with OctoPrint over HTTP led to discovering cURL. cURL is a command-line tool for transmitting data over multiple network protocols. As an example, entering something like

```
curl -H "X-Api-Key: 12345678" -X GET 192.168.1.10/api/printer
```

will retrieve information about the printers connected to the Pi in JSON format and write said info to the console. This approach was a step in the right direction, but using only a command shell it was challenging to process the information in a way that was suitable for automation, not to mention communicating with the IPC.

3.6.2.4 REST API

While trying to better understand how cURL requested and received formatted data from OctoPrint, it was discovered that OctoPrint had its own so-called REST API^[13]. REST stands for Representation State Transfer and describes a set of rules for how to create stateless web services. Commonly used to interact with IoT-devices, it allows for clients (like the IPC) to communicate with a server (like the Pi) over HTTP. In essence, the client sends an HTTP GET or POST request, then the server performs the associated function and responds with a JSON-formatted text object.

Handling and parsing all of this data would have been a cumbersome ordeal in IEC 61131-3-compliant languages. It was therefore decided that a script for handling communication was to be programmed in Python using the PyCharm IDE.

3.6.3 OctoPrint Communicator Python Script

OctoPrint Communicator^[14] is the name that was given to the script handling the communication between the printer-connected Raspberry Pis and the IPC. Initially it was intended to handle printer communication using cURL and sending network data directly to the IPC over UDP, but as development proceeded, things took a different turn.

Similarly to the Laser Cutting AddTextToDXF program described in section 3.7.1, this script runs locally on the IPC and communicates with Sysmac by reading and writing CSV files. This approach was chosen due to its simplicity, the fact that Sysmac already had support for working with CSVs in a folder that simulates an SD card in Windows, and because transferring data via .CSV files was the only simple and reliable method found to transfer data between Sysmac and the Python script. For more information about IPC communication, see section 3.5.5.

3.6.3.1 Requests and user input

To achieve proper HTTP communication, the Requests-library for Python was used. A considerable amount of time was spent on trying to formulate requests that fit the criteria set by the OctoPrint REST API. For security reasons, OctoPrint's API key needs to be supplied alongside the Pi's IP address upon certain requests. The API key can be found in the OctoPrint settings as shown in the figure below.

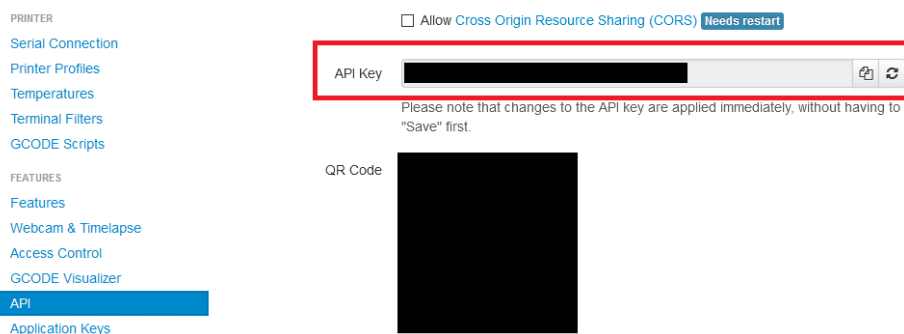


Figure 3.37: API key location in OctoPrint settings

In the beginning, the IP addresses and API keys corresponding to the printers were hard coded into the script, and the responses from the HTTP requests were written to the console as raw text strings. This was absolutely suboptimal, as it meant that any modifications to the printer setup would require rummaging through and editing the source code. To make it simpler for the user to set up the lab, a CSV file called *ListOfPrinters.csv* was created in which necessary info for printer operations can be added manually. Eventually, a text file called *config.ini* was also added to add further control over things like file paths and HTTP timeout thresholds.

```

1  # For configuring the OctoPrintCommunicator script.
2  # Customize file paths and tweak settings to suit the machine you're deploying it on.
3
4  [Paths]
5  ListOfPrinters = ListOfPrinters_HOME.csv
6  PrinterStatus = PrinterStatus.csv
7  PrinterCommands = PrinterCommands_HOME.csv
8  Log = Log.txt
9
10 [Settings]
11 # Connect all printers to Pis on script startup. It is recommended to use the connect-command for reconnections.
12 StartupAutoConnect = True
13 # If verbose is set to true, print status and info to console
14 Verbose = True
15 # HTTP Timeout threshold in seconds
16 HTTP_timeout = 2
17 # Time between program cycles in seconds
18 CycleTime = 4

```

(a) config.ini

	A	B	C	D	E	F	G	H
1	ipAddress	apiKey	username	password	rackID	xPos	yPos	comment
2	IPADDRESS1	APIKEY1	USERNAME1	PASSWORD1	printer rack ID	X-position from bottom left	Y-position (height) from bottom	comment space
3	IPADDRESS2	APIKEY2	USERNAME2	PASSWORD2	printer rack ID	X-position from bottom left	Y-position (height) from bottom	comment space

(b) ListOfPrinters.csv

Figure 3.38: Input used to set up the OctoPrint Communicator script

3.6.3.2 JSON parsing

As mentioned above, the server responds to HTTP requests by sending a JSON data object in return. JSON stands for *JavaScript Object Notation* and is a method for storing key-value pairs of data in human-readable text^[15]. When sending HTTP requests to OctoPrint, the responses are sent as raw text in which the data sets are nested using curly brackets. The following is a formatted example of the response to an HTTP GET request to *ipaddress/api/printer/*.

```
"sd":
  "ready":true,
"state":
  "flags":
    "cancelling":false,
    "closedOrError":false,
    "error":false,
    "finishing":false,
    "operational":true,
    "paused":false,
    "pausing":false,
    "printing":false,
    "ready":true,
    "resuming":false,
    "sdReady":true,
    "text":"Operational",
"temperature":
  "bed":
    "actual":21.6,
    "offset":0,
    "target":0.0,
  "tool0":
    "actual":21.8,
    "offset":0,
    "target":0.0
```

Listing 1: JSON response from Octoprint /api/printer

Although originally developed for JavaScript, JSON is a popular format with several languages having libraries for it. From the get-go, Python 3 has a package for encoding and decoding JSON objects simply named *json*. Using this, information from the responses were extracted and used for things like updating a CSV file containing the printers' current status.

JSON objects are also assembled and sent to OctoPrint to make HTTP POST requests, which are used for most printer-related operations. Below is an example of how to implement the POST requests from the REST API using the Requests library in Python. Note that *self.post* corresponds to the *requests.post* method, but has been slightly modified in the script to address error handling and HTTP timeout thresholds.

```
POST /api/connection HTTP/1.1
Host: example.com
Content-Type: application/json
X-API-Key: abcdef...

{
  "command": "connect",
  "port": "/dev/ttyACM0",
  "baudrate": 115200,
  "printerProfile": "my_printer_profile",
  "save": true,
  "autoconnect": true
}
```

(a) REST API

```
def connectToPrinter(self):
    """
    ~~~~~
    Connect to the 3D printer over USB. Default values are used.
    Returns response code as integer.
    Note that establishing connection may take several seconds.
    ~~~~~
    """
    url = "http://" + self.ipAddress + "/api/connection"
    headers = {"Content-Type": "application/json", "X-API-Key": self.apiKey}
    json = {"command": "connect"}
    r = self.post(url, headers=headers, json=json)
    if r is not None:
        return r.status_code
    else:
        errorStr = str(self.ipAddress) + " connectToPrinter response: No connection to Pi"
        self.logger.error(errorStr)
        if self.verbose:
            print(errorStr)
```

(b) Python

Figure 3.39: 3D Printer Connection API documentation vs. Python implementation

3.6.3.3 The OctoPrintClient Class

At the heart of the script, inside *octoprintcommunication.py*, lies a class called *OctoPrintClient*, which is used to represent each instance of communication between the IPC and a printer. Objects of this class are initialized upon starting the script from *__main__.py*, where the ListOfPrinters CSV file is read and a Python List structure called *opcs* is populated by them.

This class stores the information needed to operate each printer, like IP addresses and API keys. It also holds onto some other useful data for operating the lab, like which printer rack it belongs to and where in said rack it is placed. Using the Requests library, it handles all communication over HTTP with the Raspberry Pis.

3.6.3.4 Output: Printer Status

In the main part of the script there is a method called *updatePrinterStatus* which is run once every cycle. It calls the methods *getPrinterStatus* and *getCurrentPrintJob* for each OctoPrintClient object. These use GET requests to fetch information about the printer's current behaviour, similarly to what can be seen in the section about JSON above.

The responses are then parsed and data that is needed for the IPC to operate the lab extracted. Together with some of the data retained by the client objects, they form a CSV in which each row represents one printer

IP	Connected	Printing	Ready	Operational	Pausing	Paused	Finished	NozzleTemp	BedTemp	PrintJob	RackID	Xpos	Ypos
1.2.3.4	TRUE	FALSE	TRUE	TRUE	FALSE	FALSE	TRUE	22.6	22.5	test.gcode	1	1	1
5.6.7.8	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	185.0	50.0	test.gcode	1	2	1

Table 3.3: Example of 3D Printer Status CSV

3.6.3.5 Input: IPC Printer Commands

Similarly to the Printer Status CSV, there is a file containing commands written by the IPC named *PrinterCommands.csv*. Each row represents a printer, and can be issued a command word and eventual arguments, much like a method in a programming language. For example, the IPC can issue the command "print" with the argument being the file path of the G-code to be printed.

IP_Address	Command	Argument
1.2.3.4	print	/api/files/local/example.gcode
5.6.7.8	retrievedPrint	

Table 3.4: Example of Printer Commands CSV from IPC

There are also commands and variables meant to facilitate printer sheet retrieval, as well as shutting down the script remotely. As of May 5th 2020, the list of available commands is as follows:

IP	Command	Argument	Function
Printer IP	print	/api/files/path/to/gcode	Starts printing the selected G-code if printer is available. Needs to point to /api/files/ to be valid.
Printer IP	connect		Attempt to connect the specified Pi to its printer.
Printer IP	printRetrieved		Resets the status variable "Finished" to FALSE. Used by the IPC to control print sheet retrieval.
shutdown			Terminates and closes the script. Also works with "exit".

Table 3.5: Table showing possible OctoPrint-IPC commands

Note about handling CSV files: CSV stands for *Comma-Separated Values*, but when running spreadsheet programs in languages that use the comma as a decimal point, the separator value can also be a semicolon. This does not matter to the script, as

it is able to infer which is being used while parsing. However, when exporting the Printer Status CSV, it uses semicolons to be read by the IPC.

3.6.3.6 Deploying the script

The script is designed to be platform agnostic, as long as the machine running it has Python 3.8 or later installed. It is recommended that the Python executable is added to the system's environment variable path. This enables running Python scripts directly from the system's command line. It is usually an option available in the installer. After this is done, Python can be run from a command line by simply writing *python* before writing any commands.

Place the folder containing the script and its files where you want it to run, then modify the *ListOfPrinters.csv* and *config.ini* files accordingly.

It may be necessary to install certain packages before running the script. These packages are listed in *requirements.txt*. Provided the machine is connected to the internet, they can be installed from the command line using:

```
python pip install -r requirements.txt
```

To run the script itself, simply write:

```
python path/to/the/script/__main.py__
```

When *__main.py__* is run, the printer list and config files will be read before starting the main part of the script. What follows is dictated by whatever commands are issued by the IPC's Printer Commands CSV.

3.6.3.7 3D Printing main process overview

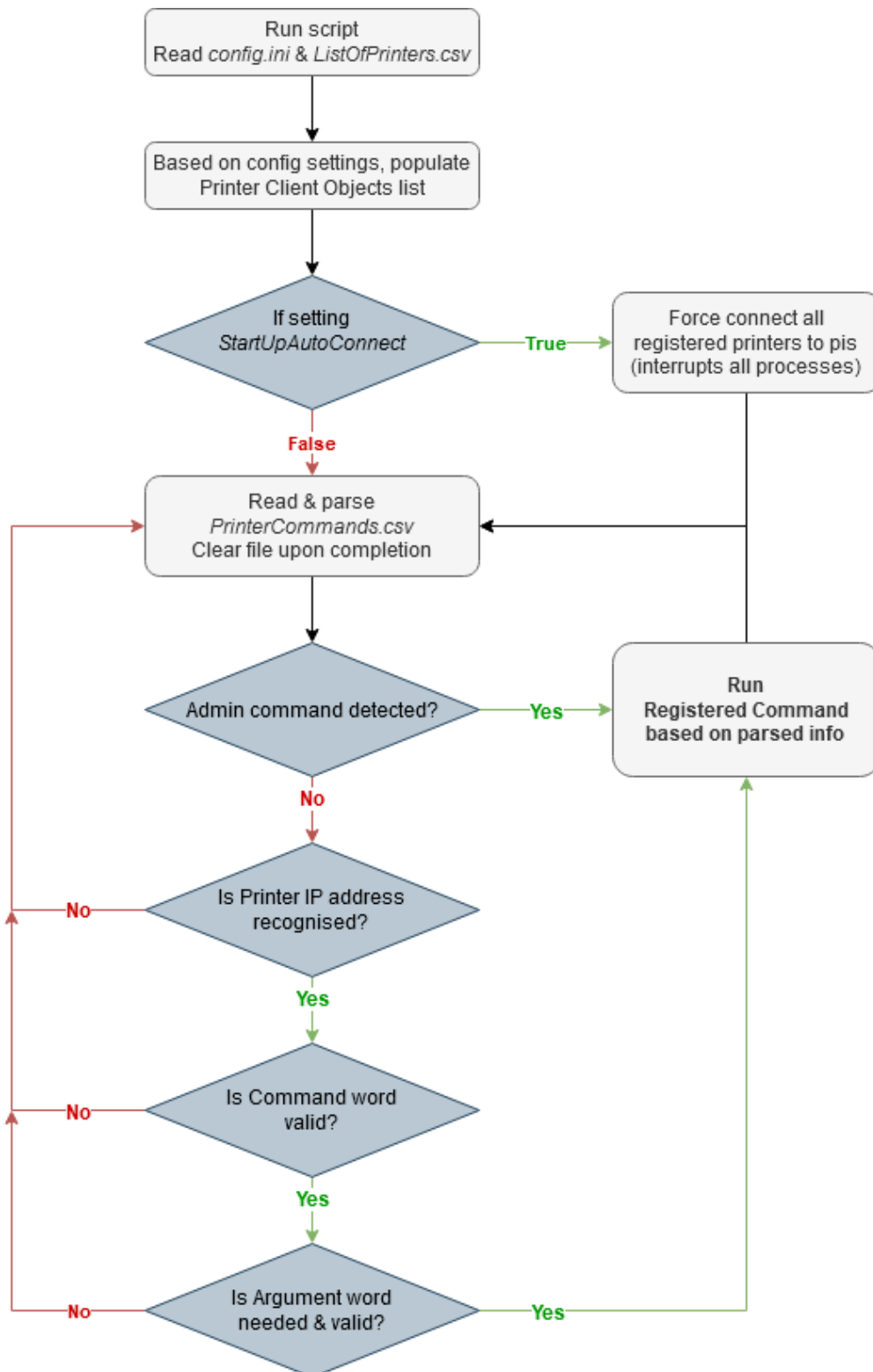


Figure 3.40: Flowchart of the processes in the 3D Printer Communications Script

3.7 Laser Cutter

Laser engraving and cutting constitutes another important part of the Manulab, allowing for high-precision cutting and engraving of parts. Inside the lab there are plans for one such machine: a Trotec S400 Laser Engraver. Another main task of this thesis was to find ways to connect the laser cutter to the other lab equipment, as well as interacting with it by using the LD-130CT mobile robot's arm. The current plan is to use the mobile robots for transporting new acrylic plates to the machine and extracting finished pieces for assembly.

Some of the challenges that need to be addressed to integrate laser cutting and engraving are:

- Creating a program for adding personalised names to a DXF-file before cutting and engraving
- Establishing some method of communication between the laser cutter and industrial equipment
- Uploading, starting and stopping jobs

3.7.1 AddTextToDXF

From the GUI, the end user will be able to order a personalised name plate. A text string is entered into the web client containing their name, which is then sent in a data structure to the Sysmac master program. The master program then exports a CSV to the "*Virtual SD Card*"-folder on the IPC, containing data to be read by the AddTextToDXF script.

3.7.1.1 Functionality

The script's primary function is to add a logo from a PNG image file to a DXF template, as shown in fig 3.22, then add the string sent from the web client under the logo and create a new DXF file. Handling of DXF files is done by utilising the Python package *ezdxf*.

In theory, the script should be able to communicate with other programs as well. The only requisite is that they follow the same formatting used for writing and reading to the CSV file. Error handling is implemented into the script to prevent it from crashing due to i.e. formatting issues, and a log file of errors is written to the folder containing the script.

3.7.1.2 Exporting Python script to Windows Executable file

As the script is responsible for a central task running continuously in the background, it should be made simple and robust. Because of this, it was decided to export the script to a Windows Executable program file (EXE). By creating an EXE it should be able to run regardless of which version of Python is installed on the machine.

As the PyCharm IDE has no inherent way to produce executable files, packages / libraries were installed to do so. These work by bundling the essential parts of the Python interpreter together with the selected script. From the Windows Command Prompt (CMD), the *pip* package manager was used to install *Pyinstaller*. Navigating to the folder containing the script, as well as creating the EXE file from it, is done by using the following commands in CMD:

```
cd C:/path/to/script/  
pyinstaller AddTextToDXF.py
```

Note that running Python commands directly from the command prompt requires Python to be included to the *Windows Path Environment Variables*. This is done by default when installing Python 2 or 3.

3.7.2 Communication

To implement the laser cutter into the greater lab, some form of communication with the master IPC is needed. This is necessary for the laser cutter to be operative without human interaction.

Like most of the non-industrial equipment in the lab, the Trotec S400 does not have the inherent ability to communicate with other machines beyond uploading new jobs from a computer. Therefore an external control board by GCC is to be installed by Amatec before delivery. This control board enables starting and stopping the machine through digital signals, as well as reading basic status signals from it. The current plan is to do so via an Omron NX1-series PLC.

3.8 Mobile Robots

As mentioned in section 3.1, the LD Mobile Robots produced by Omron and delivered by Amatec are the main workhorses for transport and are vital to provide automation to the Manulab production line. Two LD-90 robots have a conveyor belt attached to the top of them, and will primarily be used for component and product transport. The third LD robot, an LD-130CT, has an Omron TM collaborative robot arm attached to it, which allows it to be very flexible in what tasks it is able to manage and will thus act as a mobile serviceman for the production line. To prevent collisions, they have laser sensors mounted at the front, rear and on the sides.



Figure 3.41: The three LD robots used for the project
Source: Amatec

3.8.1 Specifications

Item		LD-90	LD-130CT	Note
Materials		Polycarbonate		
Dimensions (L x W x H) Without equipment		699 x 500 x 383 mm		
Weight (with battery) Without equipment		62 kg	81 kg	
Environment	Ambient temperature	5 to 40 °C		
	Ambient humidity	5 to 95% (non-condensing)		
	Operating environment	Indoor usage only, no excessive dust, no corrosive gas		Direct sunlight may cause safety laser false positive
	IP rating	IP20		
Payload	Maximum weight	90 kg	130 kg	
Mobility	Maximum speed	1350 mm/s	900 mm/s	
	Maximum rotation speed	180°/s	100°/s	
	Stop position accuracy	± 100 mm position, ± 2° rotation		± 10 mm position, ± 5° rotation with option High Accuracy Positioning System ± 25 mm position, ± 1° rotation with option Cell Alignment Positioning System

Table 3.6: Specifications for base LD-90 and LD-130CT *Source: Omron^[1]*

3.8.2 Programming

The robots are, as mentioned in section 3.2, programmed using a software provided by Omron known as Mobile Planner. This software is used to save the map that the LD robots initially scan, and from there allows the user to set up various goals and tasks on the map. The LD robot will then navigate to the given point on the map and perform the given task when told to by the PLC program using the ARCL function blocks described in the LD functions chapter of section 3.5.3. Configuration settings for the robot are also accessed through Mobile Planner, such as the minimum allowed distance from an object, enabling and disabling the ARCL server for remote control via PLC, connecting it to a WiFi hotspot, and more.

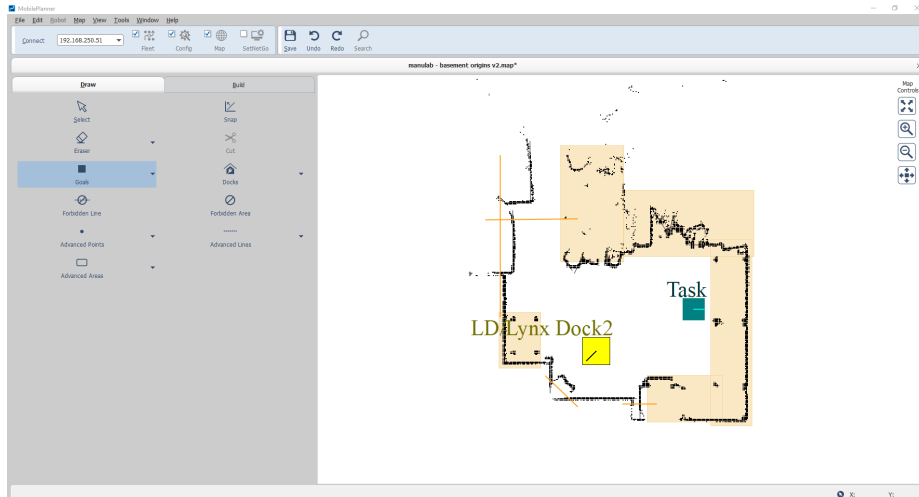


Figure 3.42: Screenshot of Mobile Planner 5.1.6 GUI

3.8.3 Communication

Communication to the LD robots are performed over WiFi, and are accessed via their individual IP addresses, whether through Mobile Planner or ARCL function blocks in Sysmac Studio. During initial setup when they were connected to the Manulab’s WiFi hotspot, they were given an IP in the range of 192.168.250.50 and up to 192.168.250.70, to ensure there is space for future expansion. If the LD robots are by themselves, they are accessed individually, but if an Enterprise Manager is present for fleet control, only that needs to be accessed, as it is made for simultaneous control and management of up to 100 LD robots^[16].

In order to connect to the LD robot through ARCL, an operator first has to enter the robot’s configuration to enable the ARCL server, and assign an IP address, port, password and connection ID. These are then inputted into the ARCL.Connect function block in Sysmac, and upon running that, a connection should be established and saved.

3.8.4 Safety

All LD robots come with built-in sensors and safety features that prevent them from harming themselves, its environment or any potential humans in their paths. Laser sensors at the front and rear scan for obstacles horizontally, while the sensors on the sides scan for obstacles vertically. These are used to ensure that the LD robot keeps a minimum distance from any obstacle or wall in the room, where the distance can be changed in the robot’s configurations via Mobile Planner. If the LD robot detects an obstacle, it simply changes its course and goes around it if possible.

Item	LD-90	LD-130CT	Note	
Safety features	Safety scanning laser	1 at front Class 1 PLd safety per ISO13849-1 15 m maximum range 240\degree field of view		
	Emergency stop	1 at operational panel	1 at operational panel 1 proximity-based for TM arm 1 proximity-based for operator panel to TM control box and TM IO	
	Rear sonar	2 at rear, 2 m range		Each pair includes one emitter and one receiver working together
	Front bumper	1 at front of platform, 2 pairs of sensors		
	Low front laser	1 at front of platform Class 1 4 m maximum range 126\degree field of view		
	Side laser	2 on horizontal tubes of HMI post Class 1 4 m maximum range 270° field of view		
	Indicators	Light disc in each side		
Speaker	3.5 in., 80 W max.			

Table 3.7: Safety specifications for base LD-90 and LD-130CT *Source: Omron*^[1]

In order for the LD-130CT with the robot arm to remain compliant with safety regulations, a two-piece sensor system was put in place where one sensor is attached to the end of the arm, and the other at the front of the LD robot. In order for the LD robot to be able to drive, the TM arm needs to be positioned in a way where the sensor pair can detect each other; if not, the LD robot effectively has an emergency stop pressed and will be completely unable to drive. A similar sensor pair safety system is used on a panel on the side of the LD robot, which is used to access the IO and control box of the TM5M robot arm. If this panel is open, the robot cannot drive.

3.8.5 Setting up and testing the LD robots

Once the LD robots were present and operational despite several delays out of our control, as described in section 5.1.5, they were integrated into the temporary lab structure in L108. They were assigned an IP range of 192.168.250.50 and up (see table 3.1) since this was available, and were moved around the basement room to create a map of the area to work in using the onboard sonar. Borders were also put up on the map in places the LD robot shouldn't go to avoid collisions as a backup measure if the sensors failed, as well as preventing the robot from trying to go through non-existent openings in the map.

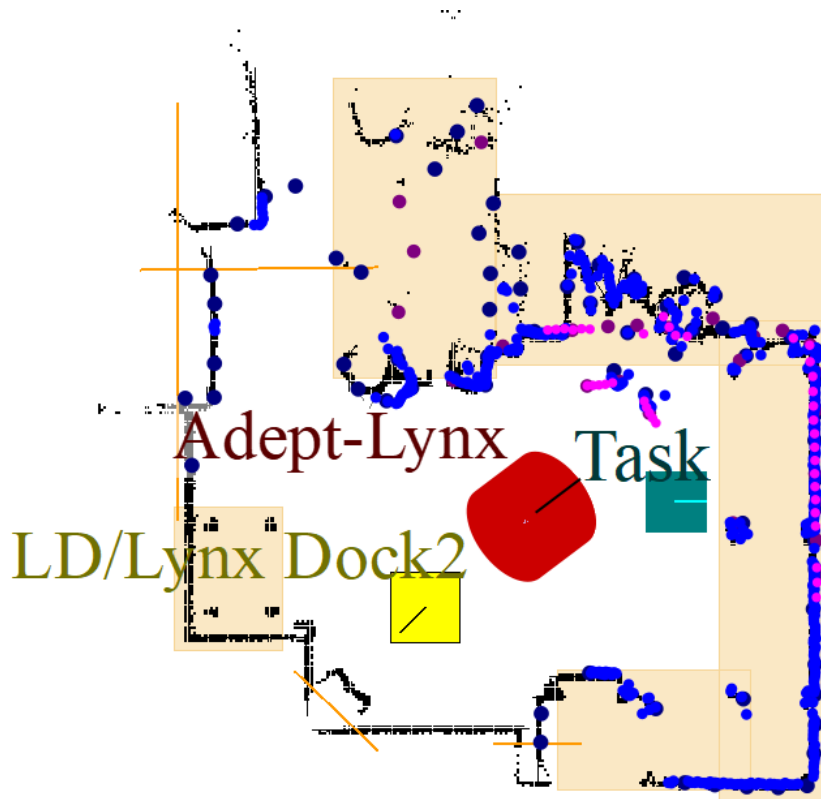


Figure 3.43: Screenshot of LD robot map from the basement in Mobile Planner 4.7.7

After the map was created, a few goals were set up to test the LD robot's ability to navigate to- and perform tasks. Various positions in the rooms were tested, but the LD robot had problems reaching them and would get stuck. Some investigating showed that the side sensors were picking up false readings, causing the robot to refuse to move, and they had to be temporarily disabled. In addition to that, the LD robot had a safety setting that did not allow it to get within 1 meter of a wall, however with the limited amount of space in the basement, that would leave the LD robot with very little space to move on. To counteract this, the setting was reduced to 20 centimeters for the time being.

With the issues related to safety boundaries fixed and temporarily circumvented, the robot was now capable of reaching almost any position in the room that wasn't very close to another object or a forbidden zone. Upon arriving, the assigned test tasks were successfully performed by the robot, such as rotating in its position, using voice synthesizer, and moving to other positions in the lab. A docking position was also set up, and the robot successfully managed to automatically dock to recharge its batteries.

3.8.6 Robot tasks

The primary task for the LD robots is delivering parts from one place in the Manulab to another, with the main routes being from cutter to assembly, from kanban storage to assembly, from assembly to delivery, and from printers to kanban storage. The LD-130CT with the TM arm will also perform general utility tasks throughout the lab that a person would normally have to do, like picking parts out of storage, opening the cutter's lid, and so on.

Seeing as the LD-90s with conveyor belts aren't capable of getting parts onto them by themselves, and the LD-130CT has difficulties carrying parts because the TM arm needs to be angled in a certain way to allow the LD to drive, part delivery will likely have to be a cooperative effort where the LD-130CT's TM arm puts parts onto the conveyor belt of an LD-90. Depending on the relative height of the destination, the LD-90 would either roll the part onto the table itself, or either the LD-130CT's TM arm or a stationary TM arm would have to pick the parts off the conveyor belt.

As described in section 3.5.3, when it comes to picking parts from the right 3D printer when delivering to kanban, the plan is that each printer will have two values, a vertical and a horizontal value. The LD-130CT robot is given the horizontal value to place itself in front of the right column of printers, while the TM arm on its back is given the vertical value to determine how far up it has to go to pick up the printer plate.

Due to the LD-130CT being incapable of driving when the security sensor does not register the accompanying sensor on the TM robots arm, it becomes incapable of interrupting the TM5M's task and driving away.

3.9 TM robot arms

In order to perform tasks such as assembly, picking up and carrying parts, opening and closing the laser cutter, and more; three TM5 arms from Omron were supplied to the Manulab. Two of them are fixed to a position next to the assembly table, while the third is attached to the back of a LD robot to make it capable of performing various tasks throughout the lab.

These TM5 robot arms are known as collaborative robots, or cobots for short, which means that they can work around humans without requiring a safety enclosure by moving at lower speeds and stop their movement if they detect too much resistance to their movement. This allows them to stand freely in the lab without being a potential danger to any bystander.

3.9.1 Robot capabilities

Item	TM5-900	TM5M-900
Weight	22.6 kg	
Controller weight	13.5 kg	14.5 kg
Max payload	4 kg	
Max reach	900 mm	
Mounting	Wall, table, ceiling	
Typical speed	1.4 m/s	
Joint range	Joint 1, 6	$\pm 270^\circ$
	Joint 2, 4, 5	$\pm 180^\circ$
	Joint 3	$\pm 155^\circ$
Joint speeds	Joint 1, 2, 3	$180^\circ/\text{s}$
	Joint 4, 5, 6	$225^\circ/\text{s}$
Repeatability	± 0.05 mm	
IP	IP54 (robot arm), IP32 (control box), IP40 (robot stick)	
Operating temperature	0 - 50 °C	
Power supply	100-240 VAC, 50-60 Hz	22-60 VDC
I/O interface	3x COM, 1x HDMI, 3x LAN, 4x USB2.0, 2x USB3.0	
Communication	RS232, Ethernet (master), Modbus TCP/RTU (master & slave)	
Integrated camera	5M pixels, colour	

Table 3.8: Specifications for TM5-900 and TM5M-900 *Source: Omron^[2]*

3.9.2 Programming

In order to program the TM robot arms, a program called TMFlow is used. As mentioned in section 3.2, it is a flowchart-based software where robot arms are programmed by adding tasks one after another. Points are added that save the robot's current position and orientation across all joints, and creating several in a row will cause the robot to follow a path through all the points, with different movement options being available between each point.

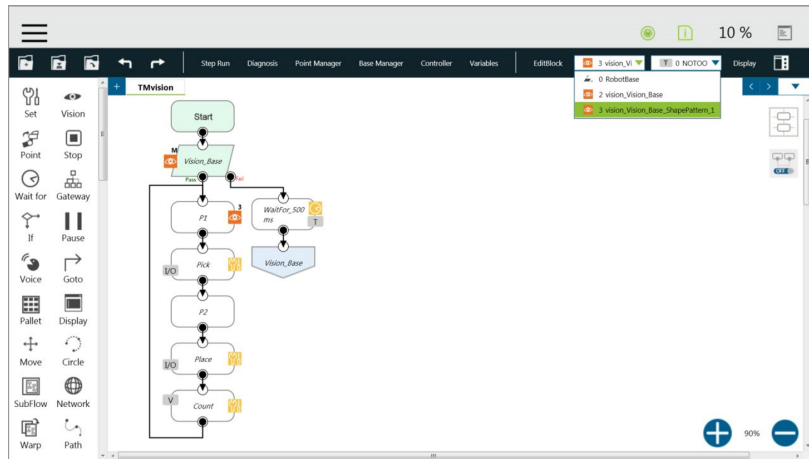


Figure 3.44: Example program in TMFlow

Source: *TMFlow Manual*^[17]

3.9.3 Vision system

The TM cobots also come with an integrated vision system, as soon on the silver-coloured camera housing attached near the robot's tool. This vision system allows the robot to recognise patterns that it has been taught, adjust its angle based on it, as well as generate a new coordinate system relative to the object. By doing this, it ensures the robot always has the same relative angle and movement to the object, no matter what original angle or position the object has, and as such is a vital tool to make the robots function reliably in the lab. There is also an option to use a so-called TM landmark, which the robot can automatically recognise and use to perform the prior actions for.

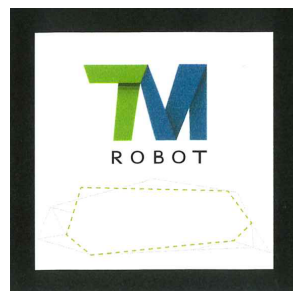


Figure 3.45: TM landmark

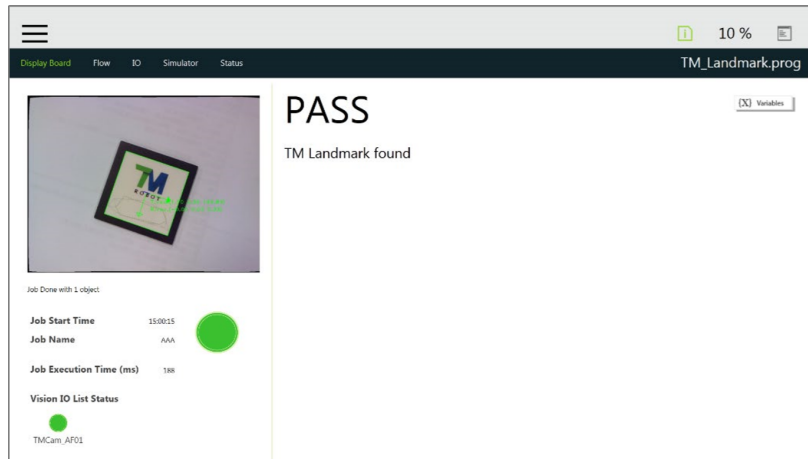


Figure 3.46: TM landmark found with Vision

Source: *TMFlow Manual*^[17]

3.9.4 Early testing

To start out, some early testing was done with the robot arms to test their capabilities for the tasks that would be required by them in the project once we had access to the main Manulab room.

The first tests done by the robot arms revolved around the 3D printers, attempting to lift the printing plate out of the Prusa printers. The first test of this was done with a vacuum gripper with a single suction cup, which while proving to be possible, was fairly unstable. A second test was later done by using a vacuum gripper with two suction cups which showed far more promise, proving both a more reliable lift and was able to be more stable when carrying the plate. It was also later tested whether a standard 2 finger gripper could perform this task, which while possible and perfectly stable, wasn't quite as effective. It required two different actions to lift the plate, there was uncertainty with the accuracy, and lifting the plate again after placing it down on a flat surface, the conveyor, was impossible using two fingers.

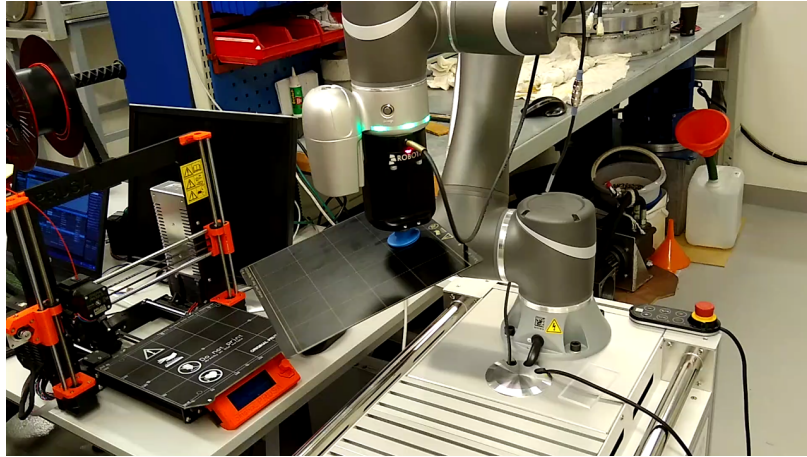


Figure 3.47: Lifting printing plate with vacuum gripper, first test

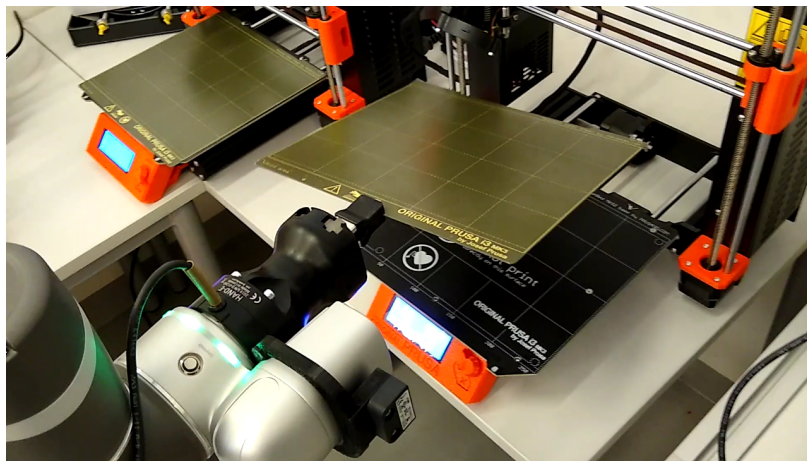


Figure 3.48: Lifting printing plate with hand gripper, first test

3.9.5 Communication

In order to communicate from the PLC program and to the TM robots, a library of functions known as the OEN Toolbox is used, which was supplied by Omron Support. It was then possible to establish connection between the PLC and the robot arms over ethernet using Modbus. This allows for transmitting a large amount of variables of various types back and forth between the two programs. In this case, a job number and a product ID is transmitted from the PLC to the robot, and a "job complete" bool is transmitted back from the robot to the PLC.

3.9.6 Robot tasks

As previously mentioned, there are three TM arms in the current iteration of the Manulab. Two TM5-900 that are in fixed positions near assembly, and one TM5M-900 attached to the back of an LD-130CT. These robots work on a general principle

that they have an idle state, also known as job 0, where they wait until they receive an integer used as a job number over Modbus from the master PLC. This idle state happens before a large branching path in the program, and depending on what value the job number has, a specified path is chosen and run. Upon completing the sub-program for the given job number, a boolean TRUE is returned to the PLC to indicate that the task is done, and the TM robot then returns to the idle state. The robot is also, as previously mentioned, given the current product ID for certain tasks like assembly and picking up parts, to ensure that the robots picks up the parts in a good and reliable way.

3.9.6.1 Stationary arms

The stationary TM5 arms would have one job only; perform assembly with the printed parts and the acrylic plate at the rotating table they're placed next to. They would await a signal sent over Modbus from the master PLC, which contained a product ID for the product currently being produced by the lab, acting as their job number. Depending on the product, it would then start the appropriate part of the program to perform the assembly phase for the given product. Product 1, the name plate, would require a cooperate effort where one arm lifts up and holds the acrylic plate in place while the other arm picks up and attaches the printed pieces. Product 2, the keychain accessory, has the frame placed on the table while one arm pushes the acrylic piece into the frame. The other arm may be required to hold the frame in place.

To ensure that the arms have the cooperate accuracy required for assembly, especially for the name plate, each arm has a TM landmark attached within easy view range that the other robot can calibrate against, and vice versa.

Upon assembly completion, the product is placed in a spot ready for delivery, a bool is sent back to the master PLC as TRUE to show that the job is done, and the TM arms then go back to idle.

Required items: hand gripper on each TM5 arm, two TM landmarks

3.9.6.2 Mobile arm

The TM5M arm on the LD-130CT on the other hand has a few different tasks, all given by a specified job number as listed below. Due to the variety of tasks, this TM robot needs to be equipped with a bracket that allows it to attach two different hand tools; a standard two-finger gripper, and a vacuum gripper with two suction cups. In order to support carrying different items, the product ID is also transmitted to the robot arm, changing what way it tries to grip the current object. At the end of each task, the arm returns to the idle position, or job 0.

Job 0 The idle state. Place the arm in the idle position that allows the LD robot to drive. This position has the arm leaning forwards, with the hand upside down

and pointing sideways so that the security sensor attached to the vision camera registers against the one located on the front of the LD-130CT robot.

Required items: Security sensor pair

Job 1 Open the lid of the laser cutter and retrieve the freshly cut acrylic plate for the product. To ensure that the robot is able to perform this task regularly and reliably, a vision system will be used to recognise patterns to ensure that it's always correctly oriented, and the robot receives the product ID to determine its gripping method. Then, depending on what is physically possible, lift and either place the freshly cut acrylic piece on a nearby LD-90's conveyor belt, or hold onto it until job 2 is started. Whether this last part is required or not, or how it would be done, could not be determined due to lack of access to the printer and acrylic plates that will be used for the Manulab.

Required items: Vision system to calibrate position, using at least one TM landmark, product ID, two-finger gripper to open the cutter, maybe vacuum gripper to lift the plate, maybe LD-90 with conveyor if required to carry

Job 2 Place the acrylic plate onto the assembly area within view of the assembly arms. If an LD-90 conveyor was required in job 1, then pick the acrylic plate up from that first. Depending on the product ID given to the robot, one of multiple methods are used to pick up and hold the acrylic plate. This job may be unnecessary if the LD-90 is capable of moving the acrylic plate onto the assembly table with just its conveyor, or if the robot arms on the assembly table are able to reach the pieces on the LD-90.

Required items if job is needed: vision system to calibrate position, maybe using TM landmark, product ID, either two-finger or vacuum gripper to lift the acrylic plate, LD-90 with conveyor to carry the acrylic plate if required by job 1

Job 3 Picking up a 3D printer plate from the specified printer in the rack. Like with the LD getting a horizontal number corresponding to the printer's column, the TM arm gets a vertical number to determine whether the specified printer is located on the bottom row or the top row. In order to ensure a good approach, a vision system then is used by the robot to align itself properly with the printer plate by looking for the warning icons at the bottom of the printer plate. The arm then picks the printer plate up by using a vacuum gripper with two suction cups on the bottom left and right corners of the plate and places it on the conveyor of an LD-90 sitting behind the LD-130CT, seeing as it is not able to carry the plate by itself while moving.

Required items: Printer row to adjust height, vision system to calibrate position, vacuum gripper with two suction cups to lift plate, LD-90 with conveyor to carry the printed plate

Job 4 Seeing as the robot is incapable of separating a printed piece from the plate itself, the process instead has to be done by a human sitting at a table near the

kanban storage. Whether this job needs to exist or not depends on the LD-90's capability of putting the printer plate onto the table by moving its conveyor belt, or needs external help. If the latter is the case, the robot simply aligns itself with the printer plate by looking for the warning icons with vision, picks the plate up from the LD with the vacuum gripper, and places it on the work table where a human can separate the pieces and put it into storage.

Required items: Vision system to calibrate position, product ID, vacuum gripper with 2 suction cups to lift plate, LD-90 with conveyor to carry the printed plate

Job 5 In order to gather the right pieces, the product ID is transmitted to the robot arm to change which piece the vision piece will try looking for. Due to having no way of knowing how the kanban storage would look, it's difficult to determine how exactly the pieces would be picked up, but in general, depending on the product ID, use vision to find and pick up either two nameplate legs, or one keychain frame. Due to the small size of the parts, the arm is able to carry these in idle position without issues, but an LD-90 could be utilised.

Required items: Vision system to find items, product ID to determine what items to find and how to grip them, two-finger gripper to carry the item

Job 6 Simply place the printed pieces from job 5 onto the assembly table, in an area where the assembly arms can find them with vision. It may be possible to generalise job 2 so it works for this as well, but this was never tested in time before the COVID-19 outbreak. If an LD-90 is used in job 5, then it may be able to move the part onto the assembly table by itself, or have one of the assembly arms pick it up.

Required items: Vision system, two-finger gripper, potentially LD-90

Job 7 Upon product assembly completion, pick up the finished product. Product ID is used to determine approach and pick-up method, and vision is used to ensure that the product is always picked up in the right way. Due to the small size of the product, the robot can carry the item while in idle position. Alternatively, place it on an LD-90 which then delivers it instead.

Required items: Product ID, vision system, two-finger gripper

Job 8 Place the finished product at the delivery position, with the product ID determining the approach. If an LD-90 is used, it may be able to move the piece onto the table by itself.

Required items: Product ID, vision system, two-finger gripper, potentially LD-90

3.10 GUI

Note: All images used are sourced from their respective manufacturers.

For this project there was a need for a GUI. As the amount of work that are needed on other parts of the project some parts were decided to make it as simple as possible. It was decided to use Movicon.NExT on the recommendation from Omron as it is easy to make Sysmac Studio, which is the program used in the master, to communicate with each other.

The GUI needs to contains multiple pages. They will be categorised under five different categories of pages:

1. Main page
2. Order
3. Status
4. Buffer controls
5. Information pop-up window

It will be needed to hide the status pages and buffer controls in the GUI. They should only be accessible from the lab. A normal user should not be able to access the status of equipment and the buffer controls. Only authorised operators in the lab should have access to them.

The text in use on the buttons and other objects with text fields are strings from a reference table. Movicon.NExT supports a text reference table for strings for multiple languages. By using the strings it will be easy to add other languages and edit grammar mistakes. This function can be used by setting the string in the *text* and *tooltip fields* in the proprieties for the object.

3.10.1 Main page

The first category is the main page. Movicon.NExT has *embedded tab screens* with the ability to insert a smaller screen into other larger screens. With this it is possible to make top banners shared by other screens and make the screens reproducible and let them share buttons and images.

The top button will switch screen to the Order page. Movicon.NExT has some different explorers objects in the screens can use, one of them is the command explorer. The button will given a Open Screen command to switch screen to the order page. The button below is only a placeholder button.

In the top banner there is a logo for the lab in the centre and a button to the top right corner. The button will navigate a operator to the status main page. The button will not be visible for a normal user on the web client. This function can be toggled in properties of the button under visibility.



Figure 3.49: Main page of the GUI in Movicon.NExT as seen in the editor.

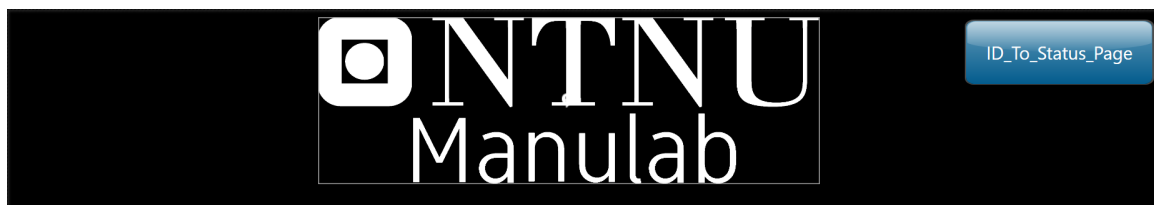


Figure 3.50: The top banner used on the main page and order page as seen in the editor.

3.10.2 Order

The second category of pages is the order pages. This pages uses the same top banner as the main page. As there is only two products they were put directly unto the page instead for making slots as it was done in the status pages.

For each product it was used a *group box* to contain all objects for that product. It will contain at the top a picture or rendering of the complete product and a description of the product. At the bottom is a text field where the user can write the text that will be put on the plates of the products. The bottom button are set to transfer values to tags connected to the master. It will transfer the article number, the text in the text field above and a boolean equals true to start the process of production.



Figure 3.51: Order page as seen in the editor.

3.10.3 Status

The third category of pages is the order pages. Only authorised operators are allowed to check the status of the major equipment in the lab. It will only be accessible from the lab as the button to these pages will be invisible on the web client. These pages will tell the operator the status of all major equipment in the lab. It has a top banner with less height than the one used for the main page to make it so the pages will be able to hold more information. It also has a bottom banner which holds buttons, for any controllers in the GUI, which currently is only the buffer control. The pages use embedded tab screens to make the slots for the equipment. The status pages get a status codes from the master and is translated to the real status in a WWB.NET script running in the background.

The main status page holds error indicators for all workstations. The indicators are directly connected to tags in the tag-list. The tags used for the indicators on the main status page will indicate an error in the workstation if there is an error in one or more of the major equipment's. To achieve this when an error occurs the tag will be turned to true when there is an error.

To access other status pages the operator can navigate using the button on the left side of the screen. The buttons have been placed on their own screen to easier replicate them on all the status pages.

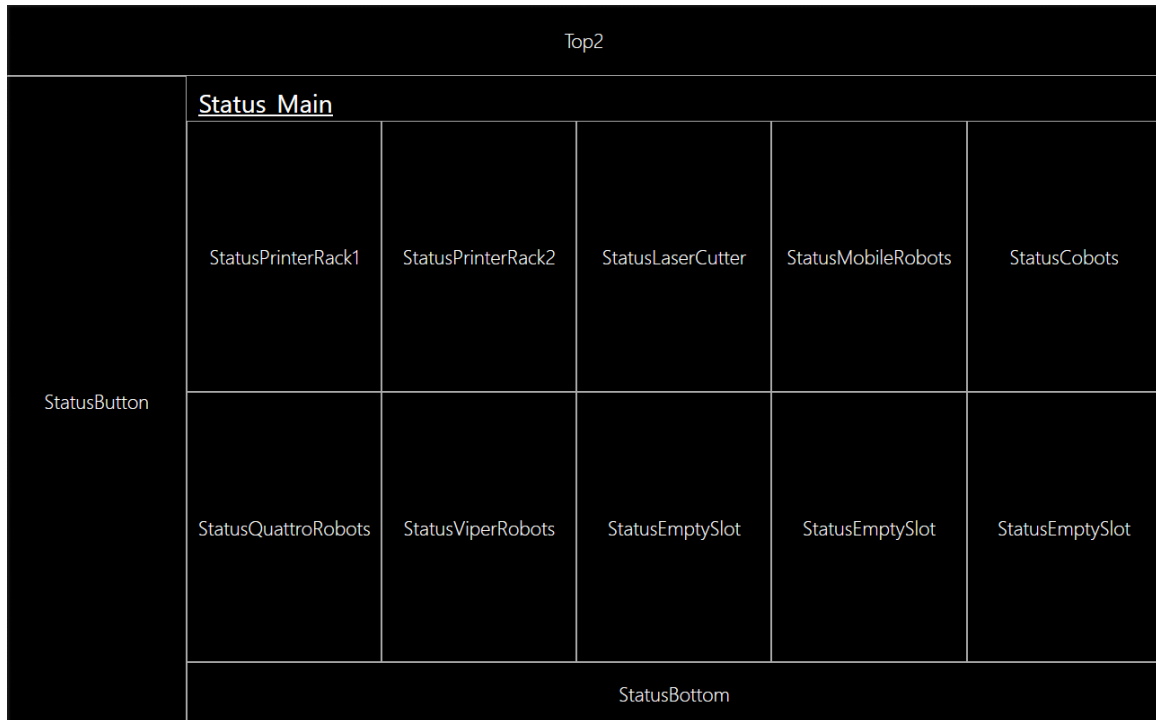


Figure 3.52: A status page, here used as the main status page, as seen in the editor.



Figure 3.53: The second top banner used in the status pages and buffer controls as seen in the editor.



Figure 3.54: The bottom banner used in the status pages and buffer controls as seen in the editor.

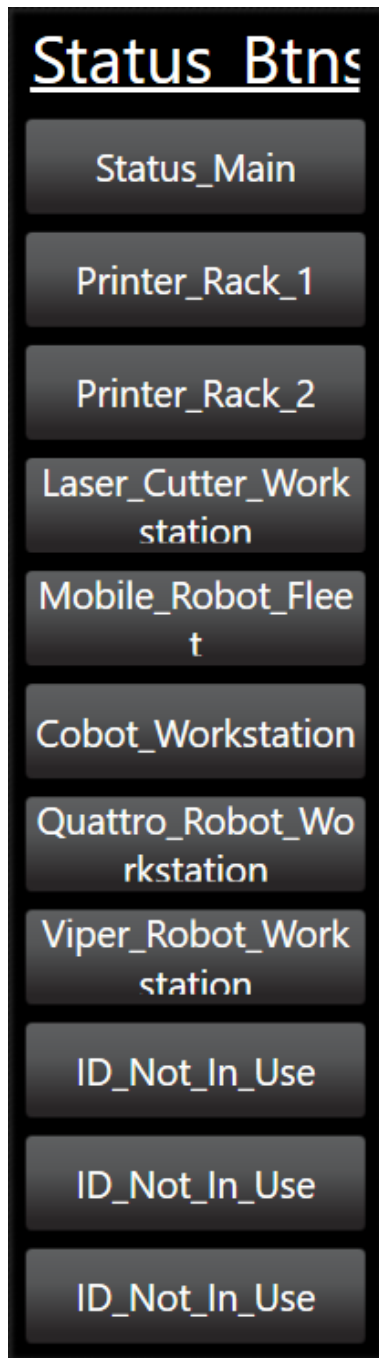


Figure 3.55: The navigation buttons for the status pages and buffer controls in the GUI as seen in the editor.

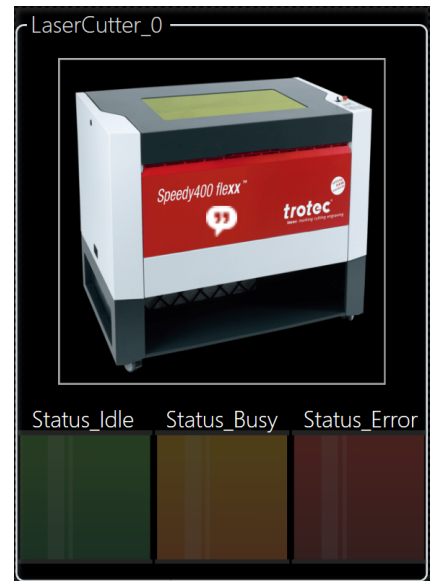


Figure 3.56: The status slot for a Trotec S400 Laser Engraver



Figure 3.57: The status of the workstation of the mobile robot fleet.

3.10.3.1 UpdateStatus

Status shown in the GUI is controlled by a WWB.NET script running in the background. The script gets the status codes sent from the master in the tag list and set them to local variables. Then it will read the the code as a 16 bit integer and set the statuses with a select...case statement, as shown in figure 3.58. The workstation errors are also set with locale variables. The variables will first be set to false before the statements and then set to true, if any of the status codes from the master is an error code or unknown code. Lastly, set the boolean value to the workstation tag in the tag-list to show there is an error or not.

When testing the script offline there were found some issues. After a few tests with the first few iterations of the background script, where the status codes from the master was in an array, it was found that the script function in Movicon.NExT appears to not have access to arrays in the tags list. To solve this issue it was decided to change the array to individual variables instead. Afterwards the script was successfully tested, except when testing it offline the script will try to contact the master to update the tags for each tag in the tags list and slows down the processing of the statuses. It was not tested on the IPC due to complications, which are explained more in section 5.1.

```

84      ' Status for 3D Printers
85      '
86      ' Status for 3D Printer 00
87      Select Case currStatCodePrinter01
88          Case statCodeIdle
89              statusVar_printers_statusPrinter00.IDLE = True
90              statusVar_printers_statusPrinter00.BUSY = False
91              statusVar_printers_statusPrinter00.FAULT = False
92          Case statCodeBusy
93              statusVar_printers_statusPrinter00.IDLE = False
94              statusVar_printers_statusPrinter00.BUSY = True
95              statusVar_printers_statusPrinter00.FAULT = False
96          Case statCodeFault
97              statusVar_printers_statusPrinter00.IDLE = False
98              statusVar_printers_statusPrinter00.BUSY = False
99              statusVar_printers_statusPrinter00.FAULT = True
100             workstationErr00 = True
101          Case Else
102              statusVar_printers_statusPrinter00.IDLE = False
103              statusVar_printers_statusPrinter00.BUSY = False
104              statusVar_printers_statusPrinter00.FAULT = True
105              workstationErr00 = True
106      End Select

```

Figure 3.58: The select...case statement setting the state variables in the tags list.

3.10.4 Buffer Control

The fourth category of pages is the buffer control pages. This page uses the same top banner as the status pages. As there is only two products they were put directly unto the page instead for making slots as it was done in the status pages.

For each product a group box to contain all objects was used. It will contain at the top a picture or rendering of the 3D-printed part and a description of the part.

Below the picture and description there are multiple text fields displayed, which are connected the masters tags in the tags-list. The top display will show the amount of the correlated parts currently in production. Below the in production display there are, to the left, a display that shows the current buffer limit and one for the limit to be set and lastly a button to set it. The upper and lower amount that can be set has a limit decided by the master. To the right the display will show the current amount of parts in storage and the buttons below will correct the amount by either increasing or decreasing the current amount.

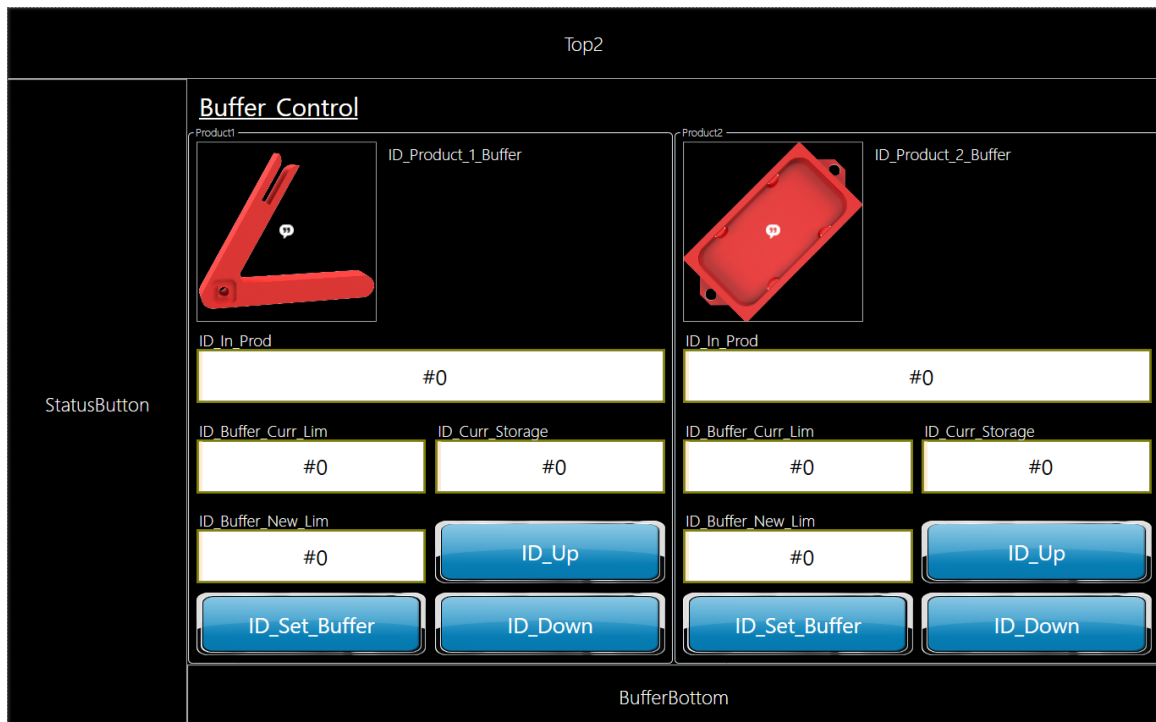


Figure 3.59: Buffer control used in the GUI as seen in the editor.

3.10.5 Information pop-up window

The fifth category of pages are the information pop-up windows. These windows will hold information and instruction of the page they can be accessed on. To access the pop-up window the user can push the button shown in figure 3.60 which is placed on the bottom right of every page. The button has a Open Screen command to open the window. To open the window as pop-up window a setting in the command needs to be set. The Screen Opening Mode needs to be set to Synchro to be a pop-up window.



Figure 3.60: The button used to open the information pop-up window.

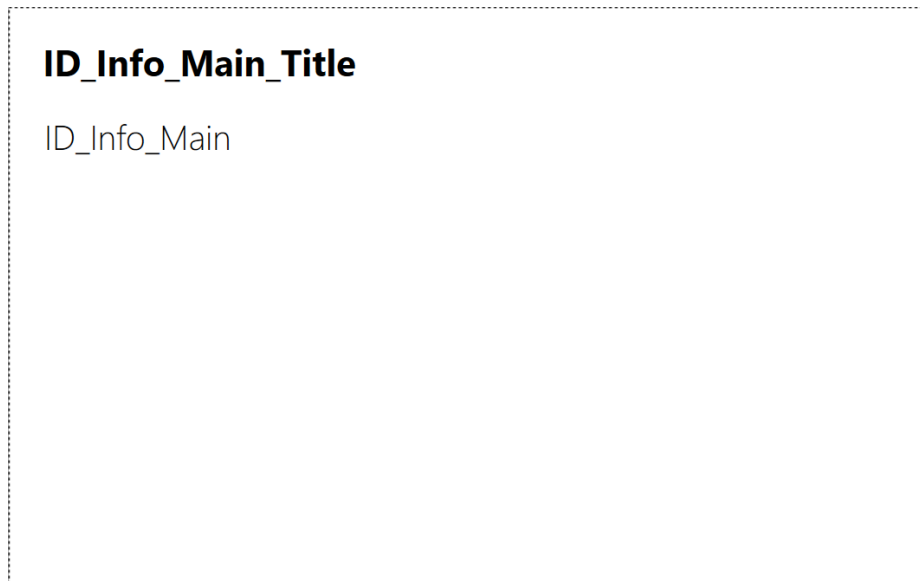


Figure 3.61: Information pop-up for the main page in the GUI as seen in the editor.

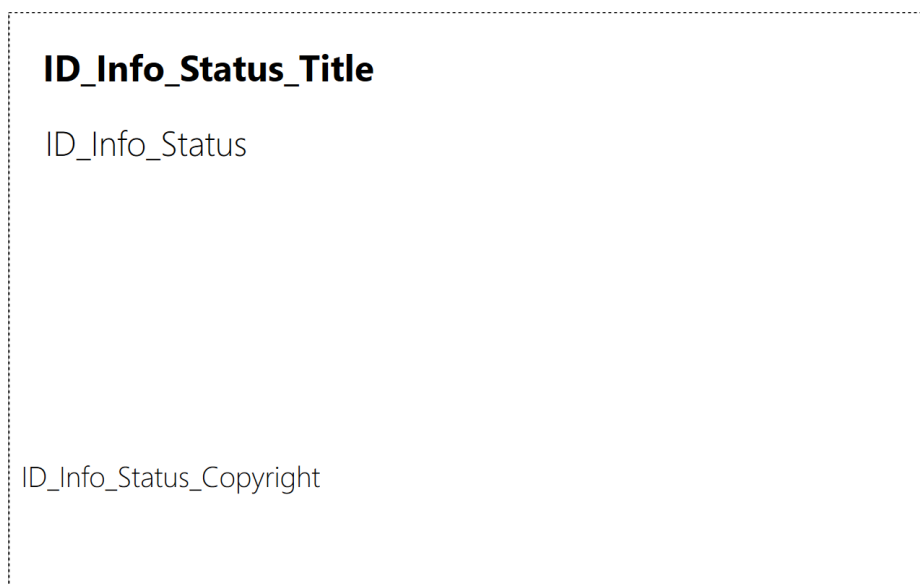


Figure 3.62: Information popup for the status pages in the GUI as seen in the editor.

3.10.6 Web Client

For this thesis there was decided to make a simple website to be able to order products from. Movicon.NExT has the functionality to create a HTML5 web client and a GUI from the same project.

The development of the web client were stopped due to the restructuring after the COVID-19 pandemic outbreak, which are explained more in section 5.1. To continue the development the web client needed to be deployed on the IPC used in the lab.

3.11 Project restructuring: Digital Twin

As mentioned in the introduction, the project had to be restructured in March as no one was allowed to work on the robots on-site due to the COVID-19 pandemic. Initially, the discussion was whether to take the project in a more theoretical direction or going for simulation. The theoretical approach would've been to focus more on Lean and write about logistics for the remainder of the thesis.

Ultimately, the choice was made to finish whatever work was possible to do from home, while experimenting with ways to create a digital twin for the Manulab. In doing so, several new challenges presented themselves:

1. Choosing which software to use for the simulation
2. Collecting and creating a comprehensive library of 3D models
3. Adding kinematics to said models
4. Investigating additional possibilities, like offline programming

3.11.1 RoboDK

Given the limited time available and the amount of work that needed to be done, it was essential to find a program that could be picked up quickly without prior experience. During a supervisor meeting on March 17 it was debated whether to use Dassault's 3DEXperience or RoboDK, with the choice ultimately landing on RoboDK. While not as packed with features, it has a less steep learning curve and a relatively intuitive interface, making it easier for future students to pick up and improve upon.

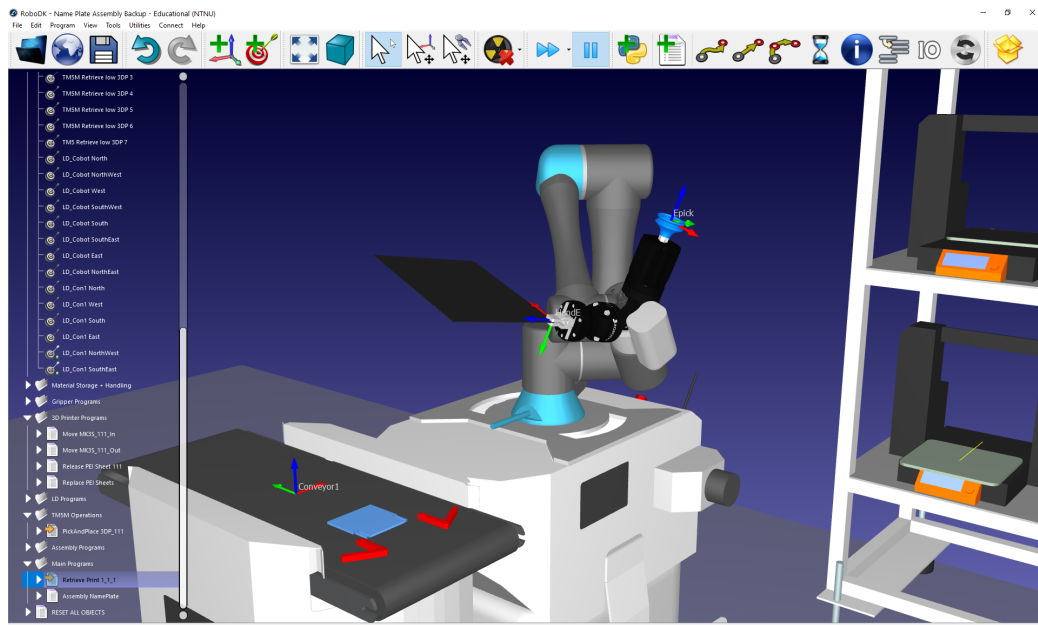


Figure 3.63: Screenshot showing the RoboDK interface

3.11.1.1 Functionality

RoboDK is based around simulation units called *stations*. Within these, coordinate systems known as *reference frames* are placed relative to each other and used to orient objects in three dimensional space. This works similarly to how custom coordinate systems are made for robots that need to do work on custom surfaces or objects. It has a built-in library of more than 500 robot arms, including the TM5-900 used in this project. RoboDK also provides post processors for some robot controllers. This means that certain types of robots can be programmed offline using RoboDK’s 3D environment, then uploaded directly to their real counterparts. Programming is done by placing ”targets” in the environment, then binding them to a robot using the movement methods *joint*, *linear* or *circular*. The targets themselves can be either cartesian or mechanical joint positions. There also exists limited support for adding kinematics to custom mechanisms.

3.11.1.2 Custom mechanisms

To create a custom mechanism in RoboDK, go to ”utilities” then ”Model Mechanism or Robot”. This will open a menu in which you can select what type of robot to build, its base reference frame and the 3D models for its various axes. Any custom mechanism has to be based on the template of a pre-existing robot type, and can interact with objects the same way a robot can. Grippers can be attached to their end-effectors making them able to pick and place objects around with some coding. While suited for developing applications for various stationary machinery, RoboDK lacks any form of innate support for mobile robots, meaning improvisation was needed.

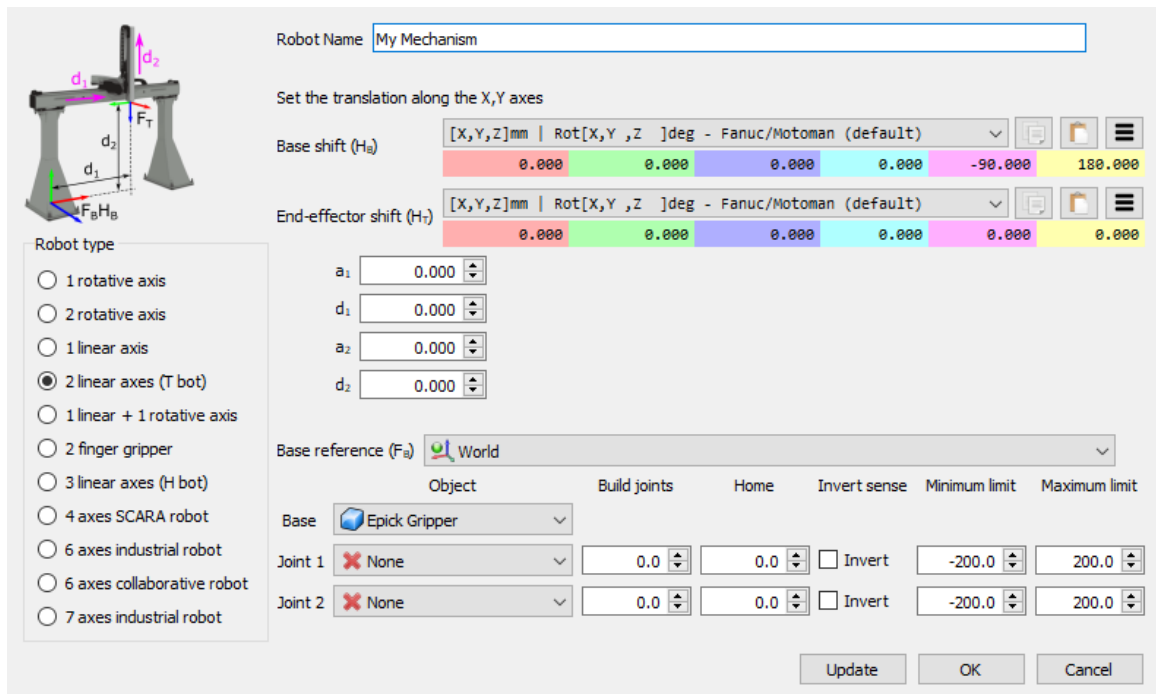


Figure 3.64: RoboDK’s custom mechanism designer

Custom mechanisms were created to add kinematics to a variety of machines and equipment, including the robots, 3D printers, laser cutter and a rotary table for mounting jigs.

The Trotec S400 Laser Engraver has a lid and front door that can be opened. The kinematic model needed two separate joints to simulate them. RoboDK's custom mechanism designer does not have a suitable preset for such a machine out of the box. To be able to make the lid and front door move separately two different mechanisms were made to be able to simulate them being opened.

The Prusa i3 MK3S model was created without an extruder, leaving the heatbed as the sole moving part. It would serve as a linear axis while the 3D printer's chassis was used as the base. The end-effector was set at the center of the heatbed, so that it could be equipped with removable steel sheets.

As for the **rotary table**, the simplified model that was provided got split, with the bottom half acting as the mechanism's base and the upper being the joint of a single rotating axis.

3.11.1.3 Designing Mobile Robot Kinematics

After reworking the CAD models for the Omron LD AMRs, they were imported as objects into their own RoboDK stations. None of the pre-defined robot types supported the translational and rotational movement necessary for a mobile robot. In the end, a solution was found where multiple mechanisms had to be chained together in an abstract fashion to make it work.

The first mechanism is an invisible 2-axis T bot, flipped 90 degrees on its side. These axes are bounded at $\pm 5000\text{mm}$ to simulate the mobile robots' translational movement along the lab floor. It was given no base or joint models, effectively acting as an invisible overhead crane. The end-effector position was set at the same coordinates as the mechanism's base reference frame.

The second mechanism was a single rotational joint, bounded at $\pm 360^\circ$. This mechanism was given no base model, only the LD model itself acting as the "joint". Everything attached to the mobile robots, including the TM5M cobot arm, conveyors and their respective reference frames, were attached to this mechanism.

Attaching the second mechanism to the first one effectively resulted in a crude, but working model of an LD AMR moving across the floor. We reached out to RoboDK to ask whether we could improve upon this design by i.e. adding some sort of path-following, but technical support is not available for educational licenses. A copy of the concept was posted to the RoboDK forums, but received no response.

Table 3.6 shows the kinematic specifications of the LD robots, including max speeds and acceleration for the different models. These values were used as a basis for simulating the mobile robots' motion.

3.11.2 Simulations

As models and kinematics were added, the focus on simulations were based around the name plate product, as this could be used to showcase the equipment that was available for this thesis. By May, most of the lab's equipment had been imported and the following simulations were made:

3.11.2.1 3D Printer Plate Retrieval

The joints moving the mobile robots around were given home positions next to the laser cutter along the western wall. From here, they're able to drive out and spin in the cardinal directions.

Using joint targets, the LD-130CT (mobile cobot) is moved to the printer rack near the lower right printer. A LD90 (conveyor bot) is then moved behind it, and the TM5M extracts the printer plate and places it on the conveyor. Attaching and detaching objects from "grippers" is done by creating "simulation event instructions" in the scripts.

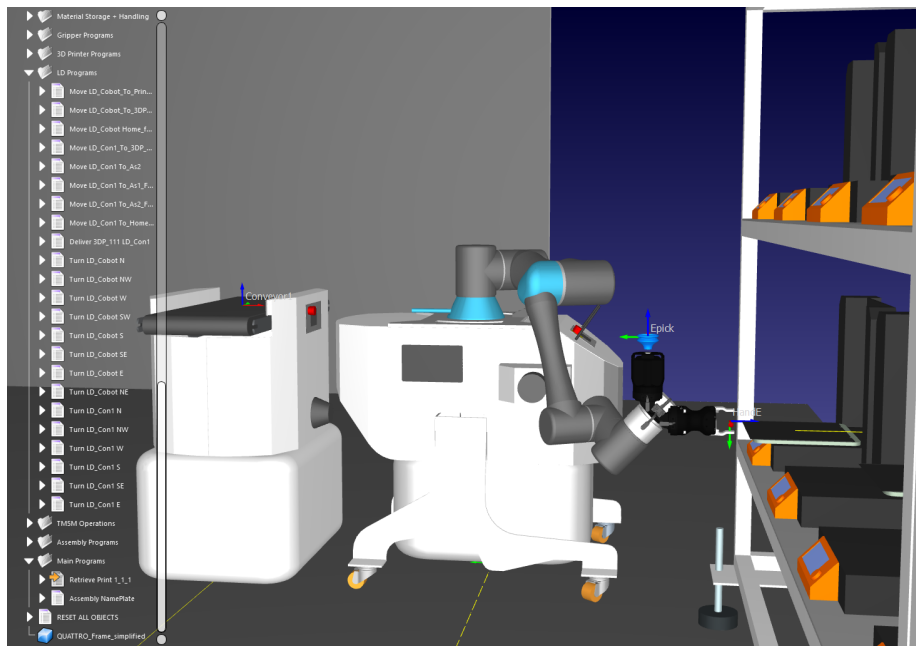


Figure 3.65: Screenshot from RoboDK printer plate retrieval simulation

When the plate has been extracted, the conveyor bot drives to an ad hoc table model for temporary storage, meant to visualise the kanban storage. The printer sheet is then pushed onto the table before the robots return to their home positions.

3.11.2.2 Name Plate Assembly

For the name plate assembly station, two TM5-900 stations are placed around the rotary table. Upon the start of the simulation, the parts are already assembled and

placed on a conveyor bot, which moves over to the rotary table, where the cobot arms pick up the parts using vision. The first cobot holds the name plate upside down, while the second attaches the two brackets. The finished product is then loaded onto the conveyor as it returns to home / storage.

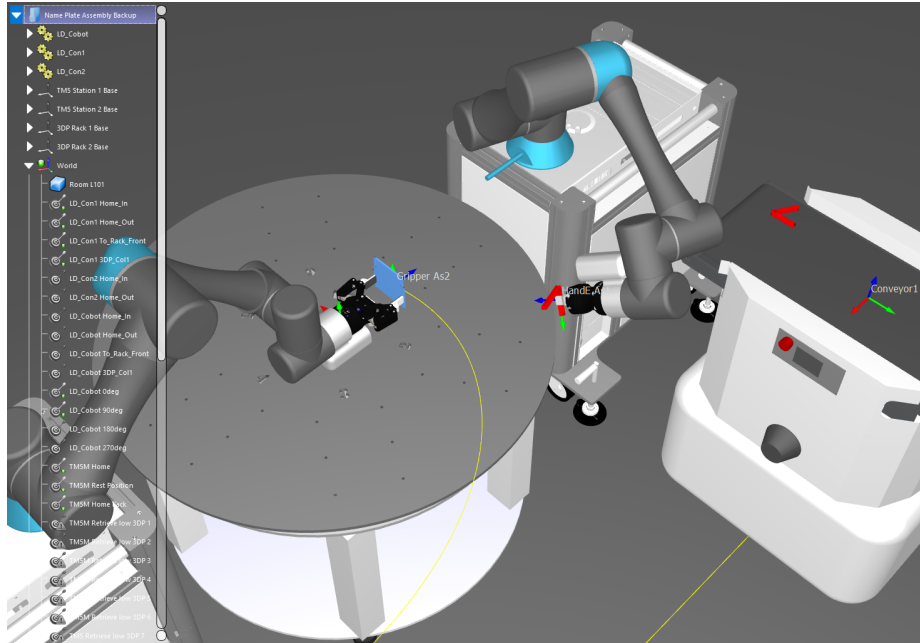


Figure 3.66: Screenshot from RoboDK name plate assembly simulation

3.11.2.3 Opening and Closing of the Laser Cutter

To simplify the full-scale lab simulations, the laser cutter is assumed to have linear actuators installed for opening and closing. This is intended to be done later upon delivery of the machine, but it is not a setup sold by the manufacturer. Thus a separate simulation was made to show how the cobot-equipped LD-130CT could be used to open and close the lid and front wall of the laser cutter. To simulate the robotic arm opening and closing the joints on the laser cutter, the joints were programmed using joint targets, synchronising their movement with that of the robot's.

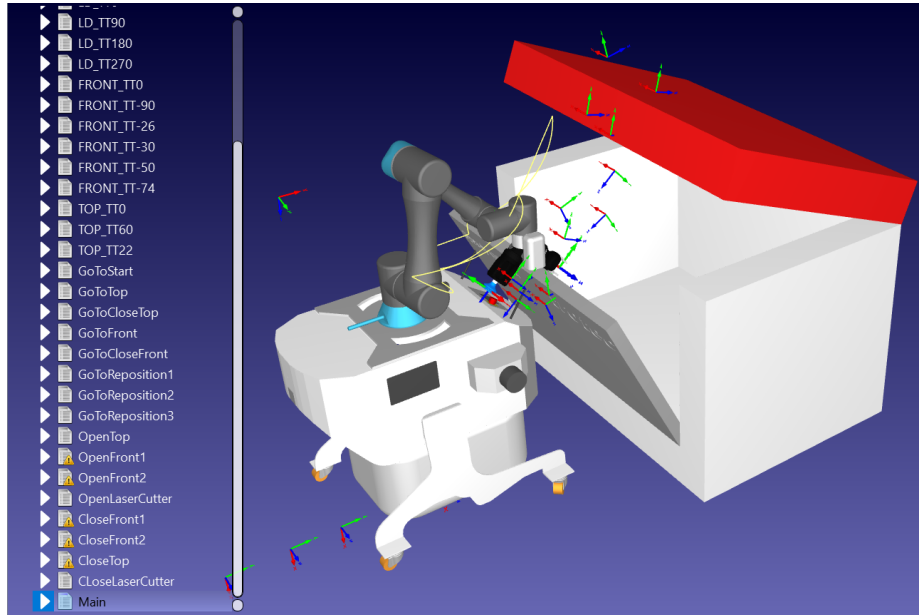


Figure 3.67: Screenshot of RoboDK simulation opening & closing the laser cutter

3.11.3 CAD

As soon as it was decided to work on digital twins, we began cooperating with supervisor Paul Steffen Kleppe on building a library of equipment CAD models. Requests for various CAD files were sent to Omron and Amatec.

3.11.3.1 Attaining the CAD models

Omron already has an extensive online library covering most of its equipment^[18]. Using the product IDs, STEP and Parasolid-files were found for most of the equipment supplied by them.

Amatec sent the STEP-files for the equipment that they were building and modifying via mail. Among the models they shared were the room layout itself, LD mobile robots, Quattro steel frame and more was generously shared by them.

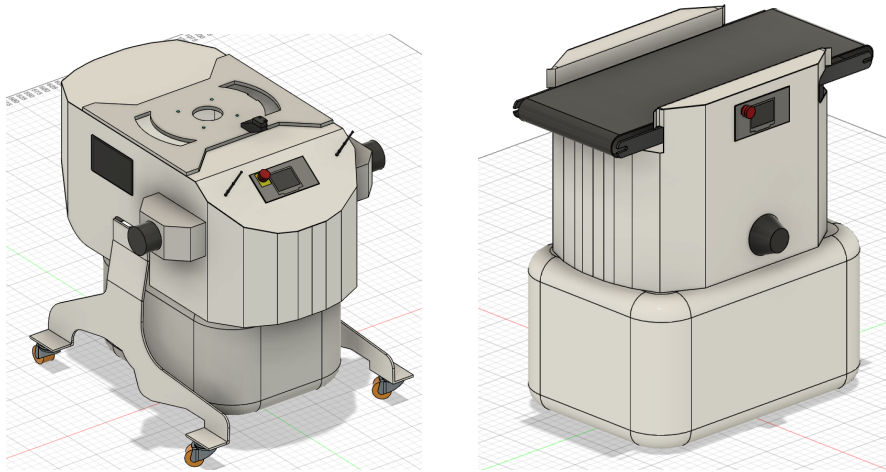
Some third party equipment was modelled from scratch, including the Prusa i3 MK3S 3D printers and the Trotec S400 laser engraver.

3.11.3.2 Model rework for RoboDK

Most - if not all - of the models had a flaw that made them unfit for use in simulations: their file size. Loading them into RoboDK was not feasible, as it took only a few models for the RAM usage to grow out of hand, causing slowdowns and crashes. Most models were also oriented with the Y-axis pointing upwards. While this is pretty common, it was an annoyance due to RoboDK using the Z-axis instead.

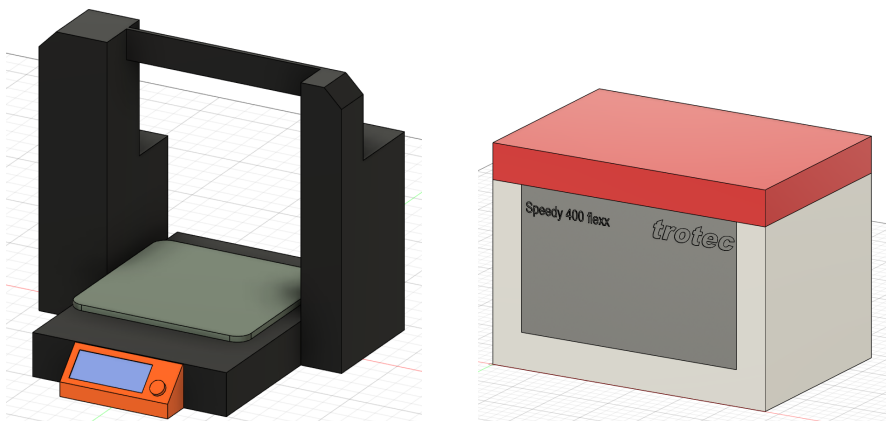
Due to their file sizes, the CAD models had to be either simplified or reworked from scratch. This ended up being done in Fusion360. Most internal parts of the models were completely removed, then the external geometries simplified. This was done by changing from *design* to *simulation*, creating a static stress study and choosing *simplify*. From here, shapes and surfaces can be simplified or replaced by primitives. Depending on their complexity, running these simplifications on a relatively modern desktop computer could take a long time. They also frequently resulted in crashes. After weeks of gathering models and simplifying them with the supervisor, most equipment was able to be imported into the RoboDK digital twin.

To change the models' orientation, the *align tool* was used. It is found under in the *design* menu, under *modify*. After changing the orientation in the settings so that Z points upwards, this can be used to snap a point or a surface to another, including any of the cartesian planes. To view the cartesian planes, *origin* must be set to visible.



(a) Reworked LD-130CT

(b) Reworked LD90 with conveyor belt



(c) Primitive Prusa i3 MK3S model

(d) Primitive Trotec S400 model

Figure 3.68: Some of the reworked and simplified CAD models in Fusion360

Chapter 4

Results

4.1 Products

In order to provide a good proof of concept, two products were designed in such a way that it utilised most of the available robots and machines in the Manulab.

4.1.1 Name Plate

In the end, a fully functional prototype of the name plate was produced, consisting of 3 pieces; a central name plate, and two legs that can work on both sides to cut down on complexity. The printed legs have sloped entrances to make it easier for the robot to thread them onto the acrylic plate's legs. The width of the mounting hole in the printed legs and the width of the acrylic plate's legs are just close enough that a collaborative robot should be capable of pushing it into the slot without much issues, and on the prototype, the legs were still attached well enough to not fall off when lifted or transported.

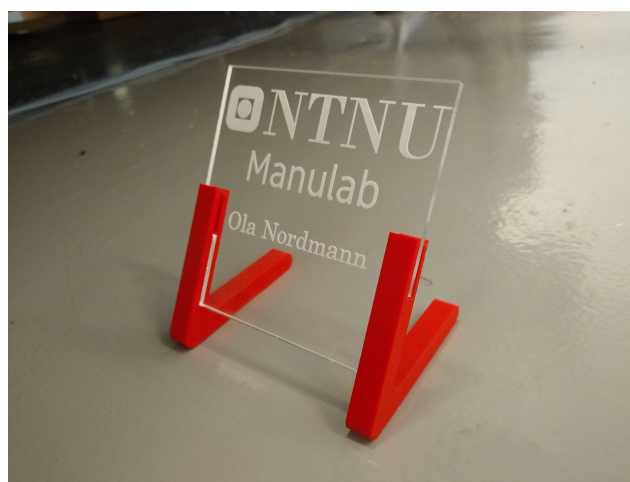


Figure 4.1: The finished nameplate

It takes approximately 1 hour and 45 minutes to print a pair of legs, while the acrylic plate is cut and engraved in a matter of 2 minutes.

4.1.2 Key Chain

As for the keychain, two final versions were eventually created. One was printed in red PLA, as seen in the image below, and one was printed using NinjaFlex TPU, as mentioned in section 3.4.2, both of which having their ups and downs. The former was much more difficult to insert the acrylic piece with the NTNU logo and initials into, but the PLA was much easier to separate from the printing plate, and the acrylic piece was very securely placed once inserted into the frame. The NinjaFlex on the other hand was very easy to handle and insert the acrylic piece into, but it also fell out more quickly due to the flexible nature of the material, and was very difficult to remove from the printing plate. As a result, a material that is a bit more semi-flexible would be preferable to print the frame in, something we were unable to test in time.



Figure 4.2: Finished keychain accessory printed in PLA, without custom name

4.2 Master Program

Due to the contingency measures put in place because of the COVID-19 outbreak preventing access to robots and other lab equipment, as well as the logistical delays prior to that, the finished program was never tested in the completed lab with all its equipment connected. Thus we were partly unable to search for bugs and issues with the code, as well as observe how well it performed and what could be improved.

Testing and verification of functionality has only been possible on separate functions using equipment that was accessible before the outbreak started, and any part of the program that could be simulated from home. Examples of this include reading and writing .CSV files for communicating with printers and the laser cutter, communicating to and from LD- and TM robots, and a few functions such as interpreting gcode file names and assigning jobs to robots.

The state-based production process has been tested to some degree, but fully testing it is impossible due reliance on outside hardware.

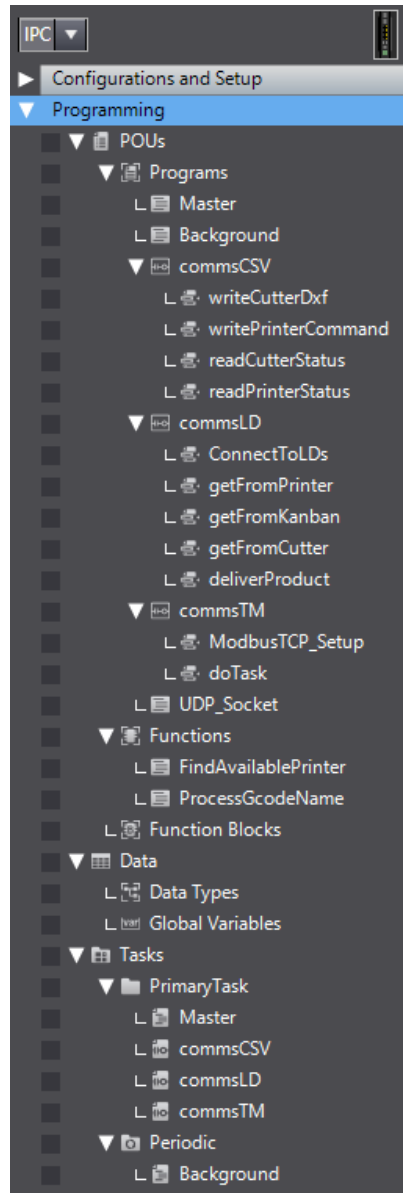


Figure 4.3: Overview of the finished program structure and its files

4.2.1 Main program

With the main part of the program relying heavily on outside signals that are difficult to replicate with the simulation mode of Sysmac, it was difficult to simulate all parts of the program to confirm complete functionality, but several parts of it were still tested separately. The state method has had a basic test run of its early stages, but without external communication present. Testing of Movicon-related functions seemed out of reach after the lockdown went into effect, as the simulation mode and Movicon couldn't communicate with each other. This meant that placing orders and transmitting status was impossible.

If there had been time, and no disruption or restructuring of the project due to the COVID-19 lockdown, one change that would've been done to the program was to decouple the assembly and delivery from the rest of the program, so that new orders can be processed while the original one is being assembled and delivered. This is further discussed in section [5.2.12](#).

4.2.1.1 Queue system

While testing the queue system in the main program itself was nearly impossible without changing a lot of code to become less reliant on outside signals, it was instead tested in a separate program by replicating the code for a simpler method using buttons. Here the queue system was proven to work as intended without any issues.

- The queue successfully added new entries to the end of the array when attempting to add new orders while another was processing, and increased the variable for the queue length appropriately.
- The program immediately started working on the queue if there was a queued order present in array slot 1 after it returned to idle upon completion of the last order.
- Upon completing the first queued order, the array of queued orders all moved down a spot in the array, meaning queued order #2 was now #1 and so on. The previous last position in the queue was also reset, and the queue length updated correctly. The queue was now ready to process the next order as soon as the program is able.
- A full run with continuously adding and working through entries was performed. The program worked flawlessly.

The program to test this can be found and tested in simulation mode in Program2 of PrototypingProgram.smc2 in the .zip folder containing all additional project files.

4.2.2 Background

Testing the background code quickly became problematic once the COVID-19 lockdown came into effect, as almost the entire job of it consists of handling inputs and outputs from communication functions that do not work in simulation mode, and that usually return structures that are difficult to replicate. Behind that, the logic is fairly barebones.

4.2.3 Communication

As previously mentioned, delays with the lab itself and the inability to access the lab after the COVID-19 outbreak prevented any testing of the program in a realistic scenario, however some tests were successfully performed before the COVID-19 lockdown came into effect. Afterwards, testing these became impossible, because Sysmac is incapable of using the required functions in simulation mode.

4.2.3.1 LD robots

Communication to the LD robots using the ARCL library proved to be successful during early testing in week 10. Sysmac successfully connected to the LD robot, status information of the robot was shown in Sysmac, and a few tests were ran where Sysmac told the robot to perform certain pre-made tasks. As such, most of the program relating to controlling LD robots should be functional in the main lab as well, at least when connecting to individual LD robots. With the lack of access to the fleet manager however, creating and testing functions against that was not possible.

4.2.3.2 TM robots

Communication towards TM robots was only briefly tested from an external PLC in week 10, but by using a TM communication program supplied by Omron at the start of the project and integrating it into the master program, variables were successfully transmitted from Sysmac to the TM robot arm and back. Testing the method of transmitting job numbers to change what tasks the robot does was however never tested before the COVID-19 lockdown started.

4.2.3.3 Printer and cutter

While testing in a realistic scenario was never performed due to lack of a completed lab and later the COVID-19 lockdown, the separate functions for reading and writing .CSV info were confirmed to work early in the project. Write functions were all successfully writing to .CSV files located on the IPC's Virtual SD Card folder, while read functions managed to extract the information from the files and turn them into readable information stored in structures in Sysmac. Attempting to make Movicon start the scripts that would then have an easier time transmitting information over UDP was however never tested-

For the printers specifically, the write function proved in initial testing, although some variables were added at a later point after the COVID-19 lockdown came into effect, which resulted in the required initial offset to the start of the first line of data to change. The number used to store this was changed to reflect this, but limitations with simulation mode prevents a full test to confirm if this is indeed the right offset or not.

As for the cutter, seeing as we never had access to the controller that would be used to let the laser cutter accept jobs remotely, we were unable to judge the capabilities of it, and thus making any attempts at creating communication for it would be impossible. We suspect that a communication method similar to printers would be utilised though, with a Python script that more easily handles JSON interpreting, while info is transferred between the script and the PLC as .CSV.

4.2.4 Functions and function blocks

The two functions previously described in section 3.5.6, FindAvailablePrinter and ProcessGcodeName, were both thoroughly tested in a separate, more simple program before they were fully added to the main master program. FindAvailablePrinter was simple and worked off the bat, but ProcessGcodeName took some tweaking. It was eventually proved to work, and was tested repeatedly using various different kinds of gcode names. An example is shown below.

```

8:
9: gcode ▶ P192_NTN... := 'P192_NTNUBadge_10,000pcs.gcode';
10: SubDelimiter(In:=GcodeTest(gcodeName:=gcode ▶ P192_NTN... ), OutStruct:=printInfoTest, Delimiter:=_eDELIMITER#_SEMICOLON);
11: printInfoTest := printInfoTest;
12: printInfoTest.productID ▶ 192 := printInfoTest.productID ▶ 192 ;
13: printInfoTest.number ▶ 10000 := printInfoTest.number ▶ 10000 ;
14:

```

Figure 4.4: Test run of the ProcessGcodeName function

The program to test these can be found and tested in simulation mode in Program1 of PrototypingProgram.smc2 in the .zip folder containing all additional project files.

4.2.5 Simulation mode

While the simulation mode of Sysmac is a thing that could be used to used to counteract some of the problems brought on by the COVID-19 outbreak, a lot of the program relies on outside signals and functions that don't work well in simulation mode of Sysmac, which makes the ability to simulate the program problematic, but it is possible.

Most functions for in-going and outgoing signals that don't work with simulation mode are mostly controlled by boolean, string or integer values that can be changed in simulation mode to replicate a successful transfer of information. As such, it is feasible but very time-consuming, as a lot of manual variable-editing has to be done throughout the run of a program, and it's very easy to trigger error modes due to lack of equipment status information unless preemptive changes are made to some variables. While the arrays and structures storing updated communication information could be replicated, this makes the process even more time-consuming.

It may be possible to make the program more easier to use in simulation mode, but there was not enough time and it was not high enough priority to focus on before the deadline.

4.2.6 Capability to add new products

During development, the program was made with the capability to expand with new products in mind, and was coded to be generic wherever possible. Necessary information on products are stored in arrays where the position in the arrays is equal to the product ID, and as such, any functions in the program that need product information pulls it from those arrays by using the product ID, while also checking for valid

product IDs by comparing it with the size of the arrays. As long as the arrays are expanded and information for them added, the program will accept a new product ID, although Movicon's lacking capability to read from arrays makes a few further changes to be required. See section [5.2.14](#) for a list of changes that are needed to add new products.

4.3 3D Printing

In the end, no more than three 3D printers were available before the March 12 lockdown happened. Regardless, most technical hurdles related to 3D printing were achieved by then. The remaining work on the OctoPrint Communicator script was done from home using an older Prusa i3 MK2 and an original Raspberry Pi 1 model B+.



Figure 4.5: Private Prusa i3 MK2 setup used to complete printer communication. A Raspberry Pi 1 B+ is mounted to the bottom of the table.

4.3.1 OctoPrint Communicator Script

Up until the lockdown, the capabilities of the communications script were limited to updating the printer status from the three printers connected in the test lab at room L108. After spending some time on development from home, it ended up working as intended when deployed via command prompt on a third party Windows machine with Python 3.8 installed. The script was able to start prints, monitor the printer's status and shut down remotely by reading the Printer Commands CSV file. It was never tested in its finished state with more printers connected to an IPC.

4.3.2 Multi-Material Upgrade

The Multi-Material Upgrade Kit provided the functionality it was intended to, but also introduced more complexity and errors. The new filament spool buffer took up more space than we could afford to spare, and starting a single-filament print required the user to manually select which spool to use. Considering how rarely the multi-material function was used, this made it a bad fit for automatisation. The modification to the extruder included a new infrared sensor to detect filament passing through it when shifting spools. The housing for this sensor had to be frequently readjusted using hex keys, as it would periodically slide out of range. This resulted in the extruder gears sometimes damaging the filament without printing anything.

4.3.3 Filament testing

Over the course of the project, several types of plastic filaments were used to print various parts. This ranged from PLA parts for the lab's prototype products to flexible TPU motor belts. Prusa Research's extensive material guide was used as a reference to determine how to achieve the best print quality using various filament types^[19].

PLA (*Polyactide*) is the most commonly used plastic for 3D printing due to its ease of printing relative to its strength. Red and pastel blue PLA ended up being used to manufacture the parts of all of the prototype products in this project.

PETG (*Polyethylene Terephthalate Glycol*) is a relative newcomer to the 3D printing world. It is the same plastic used in water bottles, and offers impressive durability and agreeable thermal characteristics, making it ideal for mechanical parts. The new Prusa 3D printer parts are printed using orange PETG. Its strong adhesion to the print surface proved to be a source of problems.

NinjaFlex is a type of TPU (*Thermoplastic Polyurethane*) used to print flexible parts. Its soft and springy characteristics require high printing temperatures and as little tension in the extruder as possible, making it hard - and perhaps impossible - to use with some types of printers. During testing it was used to print an experimental keychain, a motor belt and quadruped robot rubber shoes in a total of three separate bachelor's theses.

4.3.4 PEI Steel Sheet Adhesion

Proper care and maintenance of the print surface is a necessity to achieve consistent prints. Traditionally, heated glass plates, glue sticks and tape have been used in various combinations to achieve as high adhesion as possible. The magnetic steel plates that serve as the Prusa i3 MK3s print surface are powder-coated with PEI (*Polyetherimide*)^[20], which increase adhesion substantially. Due to this, some materials may end up so attached to the surface that they become troublesome to remove without damaging the sheet. This was very much the case for PETG prints, which would sometimes need prying with screwdrivers and knives on top of bending the sheets themselves.

As a rule of thumb, the print surface should stay clean, without the filament coming into contact with residue particles or fat from e.g. fingerprints. For most materials, it is recommended to wipe the surface clean using more than 90% Isopropyl Alcohol. With materials like PETG or Ninjabflex however, it can be a good idea to attach some glue from a glue stick to act as a separator, making it easier to remove while still maintaining adhesion. In our experience, hand soap and warm water also worked well to clean the steel sheets.

4.3.5 PEI Steel Sheet Extraction

By using custom G-code to drive the heatbed to a fixed position combined with vision on the TM robotic arms, we were able to reliably find and remove the PEI sheets from the heatbed using both the RobotiQ Epick Vacuum gripper and the Hand-E finger gripper. For sheet replacement, custom G-code was intended to keep the heatbed in place while the gripper inserted the new sheets.

4.3.6 Removing prints from the steel sheet

Initially, we assumed that we would be able to remove the printed objects from the PEI sheets by flexing them. This was to be done by simply having a robot press it against a surface, a simple jig or two robots in tandem. After printing a few different items, we quickly realised that this would not work. Some PLA prints would pop off from just flexing the sheet, but different geometries and filament types required considerable amounts of work, beyond what was feasible to do with a robotic arm. As previously mentioned, PETG sometimes proved particularly hard. Some of the larger prints took about half an hour to remove properly when using tools for prying, and even after that the sheet held onto small clusters of PETG particles.

For PLA prints specifically, an automatised removal process seems to be feasible. In this case, it should probably be handed using a jig, where the sheets can be slightly flexed and the parts softly scraped off. We did not get to test this hypothesis.

4.4 Laser Cutter

Laser engraving and cutting is an important part of the Manulab, allowing for high-precision cutting and engraving of acrylic and plywood parts. The lab was supposed to contain a Trotec S400 Laser Engraver, but this machine did not arrive while we were present on campus. The plan was to use it for cutting and engraving the name plates for the prototype products, then handle the transportation of plates and parts using the mobile robots and the LD-130CT's attached robotic arm.

The prototype products created during this thesis were made using an Epilog Fusion M2 Laser Cutter, located on campus. This machine was connected to a dedicated PC, and the software *FlexiDesigner* by AirMark was used to configure and upload CAD files for cutting.

4.4.1 AddTextToDXF

The final program was successful at adding a logo from a PNG image file, as well as the string from the GUI to a DXF CAD file. DXF files were used as templates for the products, then the program added the logos and text on top of them as specified in the CSV file from the master IPC in Sysmac.

The CSV file from Sysmac specifies all required information, including the paths to the relevant files, the text to insert, as well as their coordinates and sizes. Expected errors are handled and logged to a text file named *errLog.txt*.

More about the formatting and inner workings of the program can be found alongside the source code, in appendix I.

4.4.2 Communication

As previously mentioned in methods section 3.7, the Trotec S400 Laser Engraver does not have the ability to communicate in any meaningful capacity with the IPC. To enable that, Amatec was to install an external control board made by GCC before delivering the machine.

By March 12, the machine was not yet available. Due to the restructuring caused by the COVID-19 pandemic outbreak work on the communications were halted indefinitely. More regarding this can be found in section 5.1, [Complications](#).

4.5 TM and LD robots

With the lack of access to the robots in the lab being a problem, creating finished programs for them became impossible. Seeing as creating control programs for the LD robots and TM robots would've been a generally quick process, and would've relied on a lot on both access to the main lab and other parts of the project being finished, the task of making programs for the robots was scheduled for after the lockdown started, and as such could never be done once the project work was moved home for the rest of the semester.

4.5.1 TM Robots

In the case of the TM cobots, while the TMFlow application is able to be installed on any computer, actually creating a program on the robot is simply not feasible. Actually entering the part of the software where programming happens seems impossible without being connected to a cobot, and even if it is accessible otherwise, then creating positional points and vision nodes becomes impossible without access to an actual robot. As such, only ideas for how the program is set up were possible, which were discussed in section 3.9.6.

As such, the only results for the TM arms to speak of would be the early testing by picking up printer plates, simple communication to it over Modbus, and the ideas for what tasks to perform; all of which were already explained in section 3.9.4, section 3.9.5 and section 3.9.6 respectively.

4.5.2 LD Robots

Like with the TM cobots, the ability to perform work on the LD robots was also close to impossible to perform from home. Version 4.7.7 of Mobile Planner, which was used for the LD robots accessible at the time, had absolutely no support for offline programming. The more recent 5.1.6 version does have this capability, however the LD robots we had access to were not compatible with this version, and offline programming would have required already having created a map of the Manulab room using an LD robot. Even if a map was obtained, simulating robot behaviour in the program is only possible with the fleet manager, which there was never access to before the lockdown. Because of this, there were no results past the initial tests, communication attempt and ideas suggested in section 3.8.5, section 3.8.3 and section 3.8.6 respectively.

4.5.3 Communication between modified LD-130CT and TM5M cobot

Amatec and Omron provided instructions from ATC Europe on how to establish wireless communication to the TM5M using a standard Wi-Fi router. Following these instructions yielded no results. The LD could be accessed wirelessly, but port forwarding Modbus TCP instructions from Sysmac to the TM5M still did not work. On recommendation from Omron Support, the setup was attempted using different routers borrowed from the lab faculty.

On March 10, a technician from Omron attempted to fix it along with upgrading the LD AMRs' firmware. This also proved futile, and the firmware on the LD-130CT had to be rolled back afterwards due to it losing its networking functionality. Two days later the school was locked down due to COVID-19.

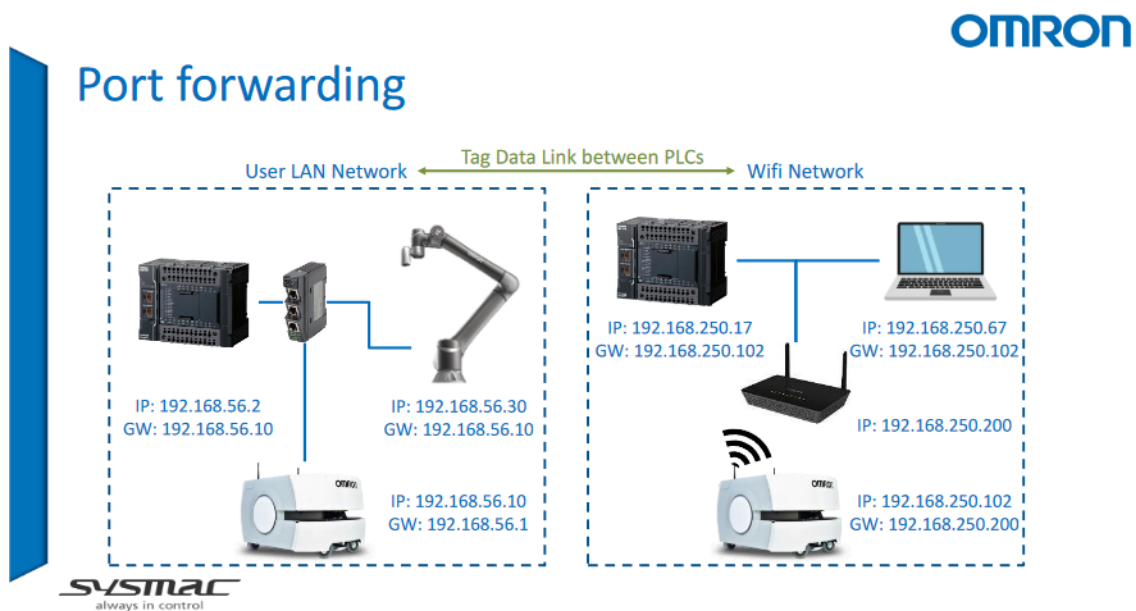


Figure 4.6: Screenshot from ATC Europe's Mobile Robot Port Forwarding guide

4.6 GUI

As described in section 3.10, a GUI was made in Movicon.NExT on Omron's recommendation. While the methods section showcased the internal structure of it, this section shows what the finished product looks like to the end user. It had not yet been tested live on an IPC before the COVID-19 lockdown, but had been simulated on other computers up until that point.

The GUI contains multiple pages which can be divided into five different groups:

1. Main page
2. Order
3. Status
4. Buffer controls
5. Information pop-up window

The status and buffer controls are not accessible for a normal user. Only authorised operators in the lab should have access to them.

4.6.1 Main Page

The main page was made with the thought of having a simple screen to navigate to the other pages in the GUI. At the top centre of the screen, the NTNU Manulab logo is placed. In the top right corner there is a button to navigate to the status pages and buffer controls, which is not visible on the web client.

It also has buttons used to navigate to other pages. The "Order product" button will move the user to the order page, but the "Placeholder button" button is currently not in use and will not navigate to anything. At the bottom there is an information button that opens a popup describing to how to use this page. The last button is an off button to turn the GUI off.



Figure 4.7: Main page of the GUI in Movicon.NExT.

4.6.2 Order

This page is where the end user can order a product from the Manulab. For each product a *group box* is used, containing the text field, description and button used for ordering.

Below the picture and description there are a text field and and a button. The users can write their names in the text field and then push the order button to begin the production of the selected product with their name engraved on it.



Figure 4.8: Order page of the GUI.

4.6.3 Status

These pages will contain the status of all major equipment in the lab. They are not available to normal users and the button to navigate to these pages will be invisible for them.

When the operator enter the status pages he will be met with the main status page. This status page will show if a workstation has an error. Each page contain 10 slots for showing status. The slots on main page shows if the workstations have an error. To navigate to the other status pages buttons on the left side of the screen can be used. Each button in use will navigate to their respective pages.

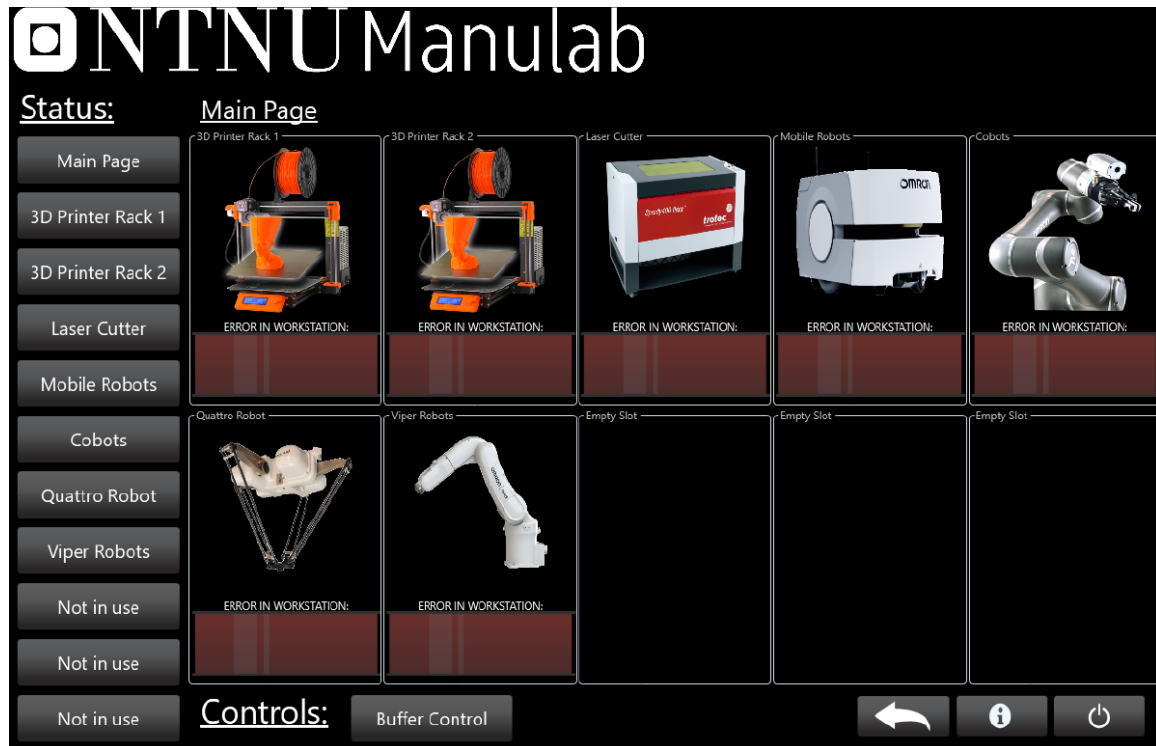


Figure 4.9: Main page for the status pages showing if the workstation has an error or not.

The other status pages show the status of all major equipment in their workstation. Each slot in use will be able to show if the equipment is

- in an idle state (waiting for the master it give it a command)
- in a busy state (currently working on a job)
- in an error state (the equipment is unable to continue working due to errors)

The GUI was only simulated in an offline environment due to complications, which are further explained in section 5.1. After offline testing there was one issue that did not get fixed. When the GUI contacts the Sysmac master program, the status updating process slows down significantly. Deploying the GUI on the IPC with the master may fix this issue.

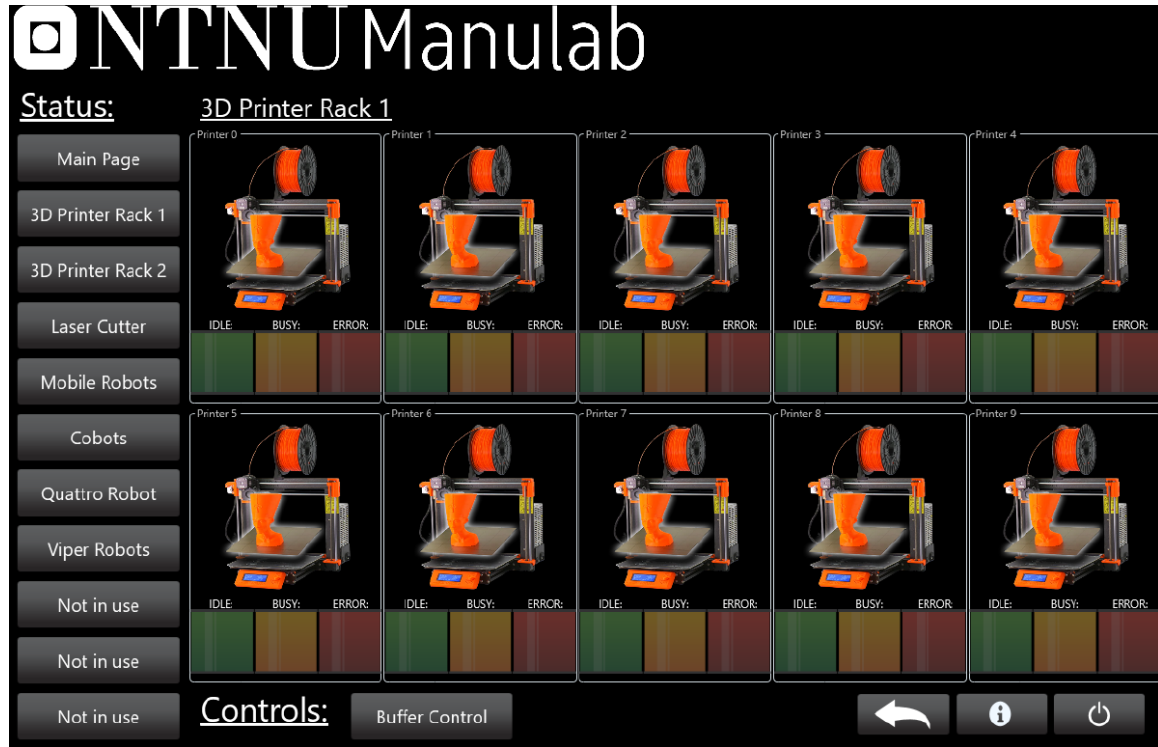


Figure 4.10: Status page for printer rack 1.

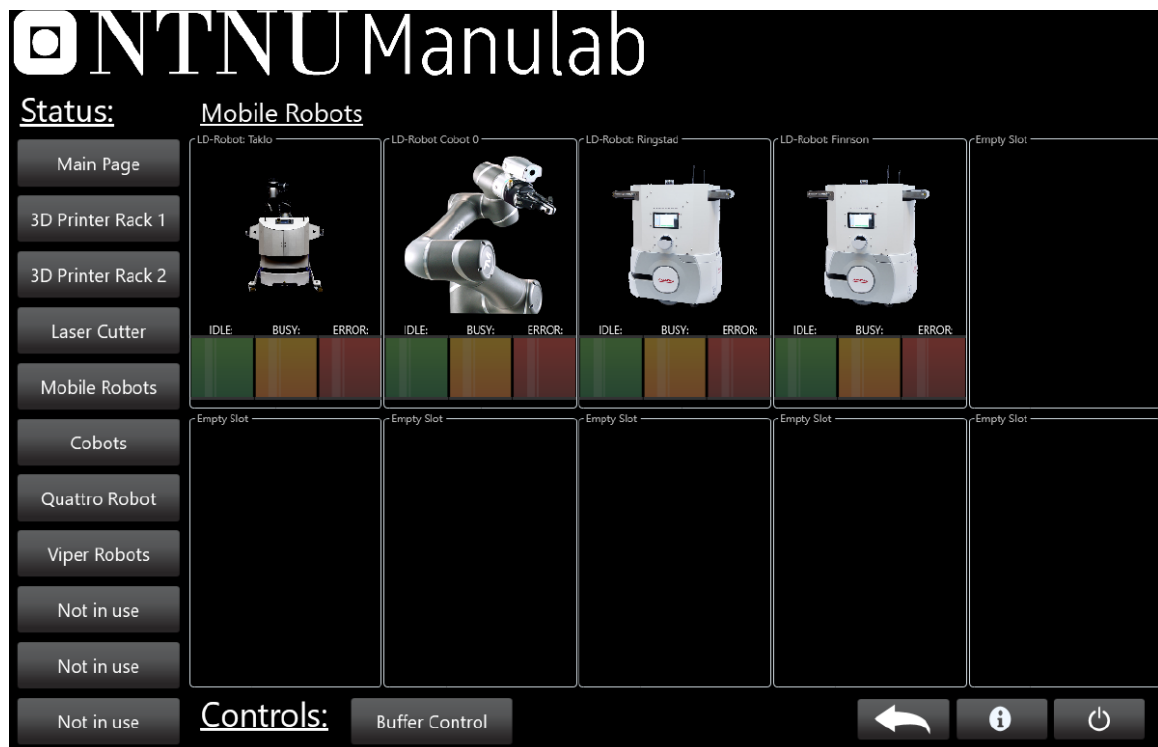


Figure 4.11: Status page for the LD-Robots.

4.6.4 Buffer Page

Printing parts using 3D printers take considerably longer than cutting parts with a laser cutter. Therefore a buffer storage was made and controls for it was added to the GUI.

To access the buffer controls one must go through the status pages. At the bottom of the screens there will be a button to navigate to the controls.

For each part a group box to contain all displays, description and buttons related to the buffer control of the parts was used. It will contain, at the top, a picture or rendering of the 3D-printed part and a description of the part.

Below the picture and description there are multiple displays. The top display will show the amount of the correlated parts currently in production. Below the in production display there are, to the left, a display that shows the current buffer limit and one for the limit to be set and lastly a button to set it. The upper and lower amount to be set are also limited to not overproduce or underproduce parts. To the right the display will show the current amount of parts in the buffer storage and with the buttons below the operator can correct the amount by either increasing or decreasing the current amount.

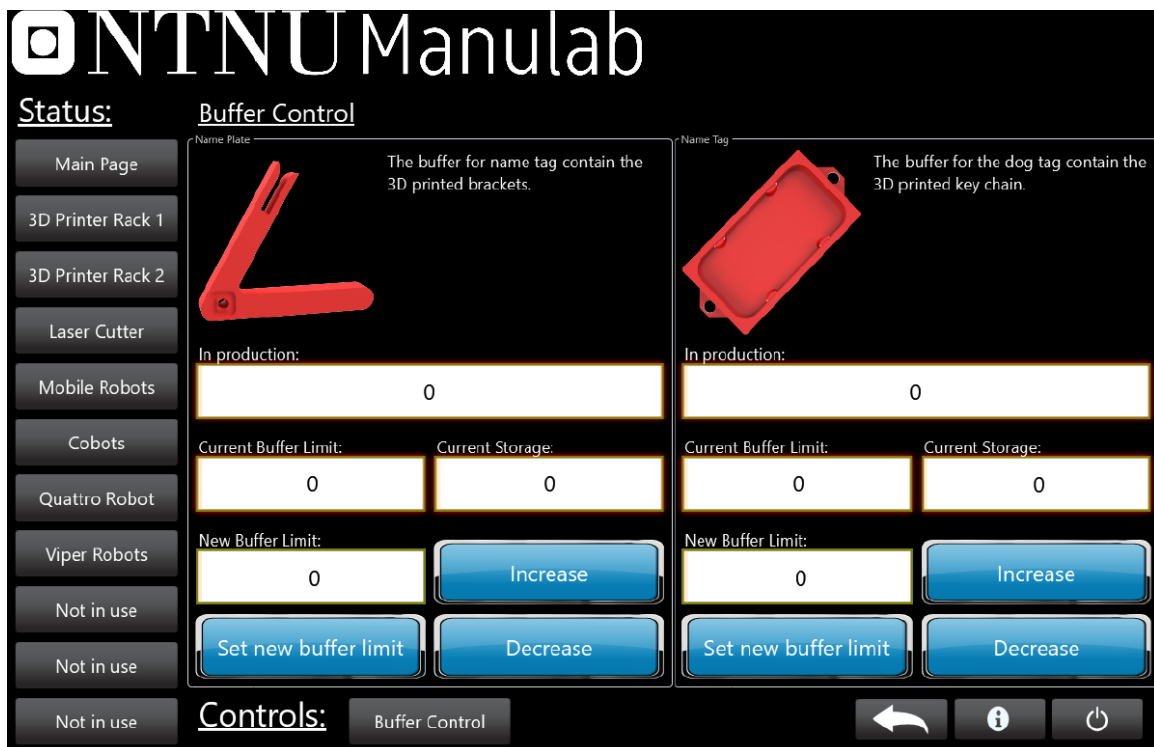


Figure 4.12: Buffer control for the buffer storage.

4.6.5 Information pop-up window

For easier use of the GUI information pop-up windows were added. These windows will hold information and instruction of the page they can be accessed on. To access the pop-up window the user can push the button shown in figure 4.13 which is placed on the bottom right of every page.



Figure 4.13: The button used to open the information pop-up window.

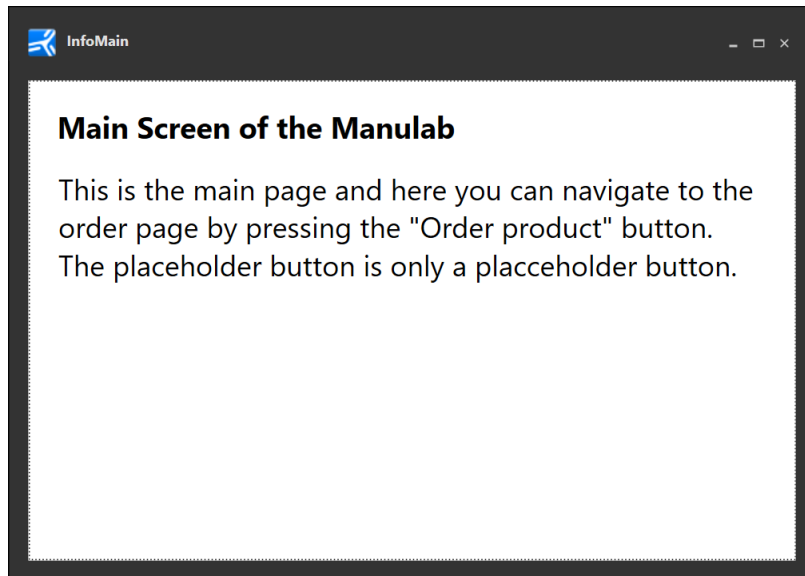


Figure 4.14: Information pop-up window for the main page.

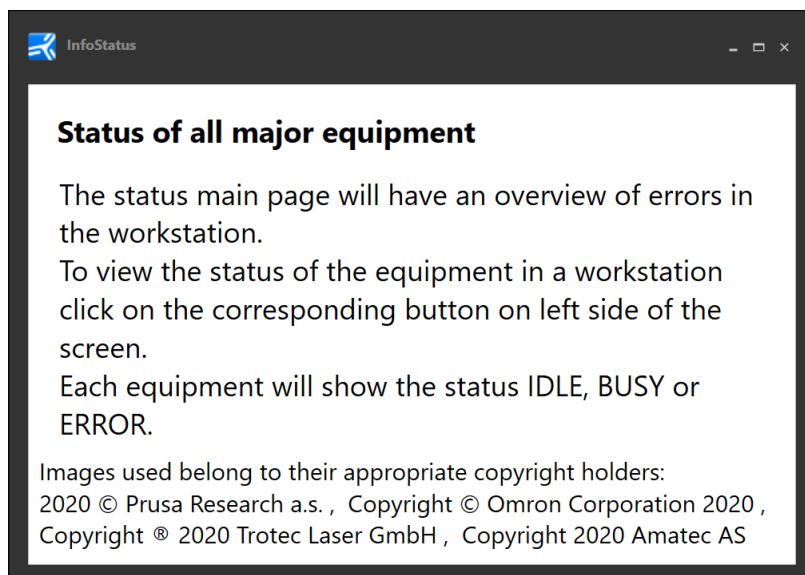


Figure 4.15: Information pop-up window for the status pages.

4.6.6 Web Client

For this thesis there was decided to make a simple website to be able to order products. Movicon.NExT has the functionality to create a web client and GUI from the same project.

The development of the web client were stopped due to the restructuring after the COVID-19 pandemic outbreak, which are explained more in section [5.1](#). To continue the development the web client needed to be deployed on the IPC used in the lab.

4.7 Digital Twin

The RoboDK simulation stations served their purpose as intended, showcasing the imported robots in motion as they retrieved 3D printed parts, assembled the name plate and opening and closing the laser cutter. By early May, most of the Manulab's equipment had been imported into its digital twin.

Given more time, we would have expanded upon it by combining more processes into one seamless simulation. We are still not sure whether all the possible simulation programs should be kept in the same station file, as RoboDK's interface can get cluttered and performance issues eventually arise.

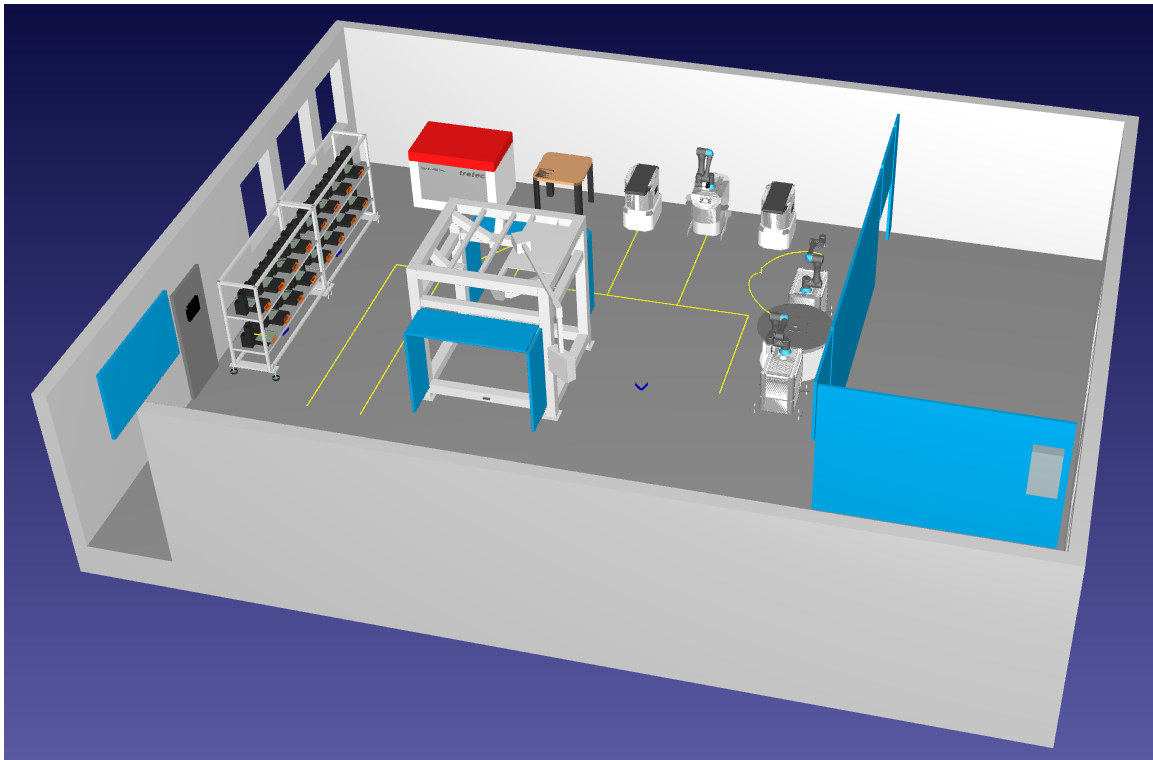


Figure 4.16: Final result of the RoboDK Manulab simulation station

4.7.1 Bugs and their fixes

The two main lab simulations ended up being reprogrammed several times. This was mainly due to the LD-130CT AMR and the TM5M cobot attached to it not always playing nicely with the reference frames and targets they were assigned to.

Presumably due to bugs in the program, the wrong robots would sometimes attempt to perform the motions in the scripts. This happened regardless of which reference frame or tool was set to active, as well as the settings in the cartesian and joint targets. These targets were deemed corrupted, and all of them were deleted and

reprogrammed. After a while it was discovered that some of these issues went away when closing the program.

For the first simulation (section 3.11.2.1, *3D Printer Plate Retrieval*), all robot targets were set in a coordinate system named "World", then the movements were programmed using them as references.

For the second simulation (section 3.11.2.2, *Name Plate Assembly*), the active reference frame was the conveyor belt on the LD-90 AMR. This time, both the mechanism and its intended reference frame were selected simultaneously by pressing left mouse + control in the Tree menu, then movements were added. By doing this, we noticed that the movement-related bugs from earlier went away completely.

As for the third simulation (section 3.11.2.3, *Opening and Closing of the Laser Cutter*) the robot targets were partially located on the lid and front wall. To be able to move the arm without collisions a lot of targets were used. Some issues appeared with the arm's cartesian targets, linked to the laser cutters front wall and lid, not connecting after the LD-130CT moved. To exclude this issue each position of the LD-130CT had to have one program block made for the cobot.

Chapter 5

Discussion

5.1 Complications

5.1.1 Lack of resources and space

When the project started in early January, work on the lab room had not begun, and the university did not have any equipment available save for two TM5-900 robotic arms. During the first month, time was spent on arranging temporary equipment and space for development and testing. The lab chief provided temporary temporary workspace in room L108, while Omron and Amatec spared some of the necessary equipment to get development started. We're deeply grateful for their assistance.

5.1.2 COVID-19 Pandemic

On March 13 2020 several public facilities, including schools and universities, were closed in Norway as a preventive measure against the SARS-CoV-2 pandemic (COVID-19). After being evacuated from the premises, we were not allowed back in, meaning that we could neither work physically on the lab equipment nor do any on-site programming or test what we were working on. At this moment the Manulab room had not yet been completed, new equipment was still being delivered, and many currently ongoing work projects that required access to equipment was prevented from continuing.

5.1.3 Restructuring

After meeting with advisors online, we decided to veer the project in a new direction. After completing what we possibly could from the comfort of our homes, we would work on gathering CAD data of the equipment and start developing a simulation of the lab itself. Ultimately, the goal would be to create a "*digital twin*" for the Manulab project. Together with our advisors, we decided to try to implement this using *RoboDK*.

5.1.4 IPC Firmware Problems

At the start of the project, an Omron NY512-1300 Industrial PC was supplied by Omron Electronics Norway AS to serve this purpose and to begin developing a program on. While this worked well for the most part, the IPC had a firmware version that was too old to be compatible with certain libraries that were required of the project to control the TM robots. While a firmware update would fix this, the firmware version was too old to support the update method normally for the IPC, and getting it updated required intervention from Omron Japan, and was a too lengthy process to wait for. As such, an additional NX102 PLC with a more recent firmware was brought in to control functionality for the TM robot arms until the IPC supplied by Amatec would arrive in the finished lab.

5.1.5 LD Robot Delays

Due to several delays, it took a large amount of time before testing could even begin on the LD mobile robots. The robots were not delivered until February 10 because of Amatec having issues integrating safety features. When the robots were finally delivered, the USB license dongle required to work with them was not included in the delivery, and resulted in yet another week of waiting until the robots could be set up and tested.

In addition to this, communication towards Sysmac on the Master PC was the another problem, as this process was simplified by an ARCL library made locally by Omron Norway and was not accessible elsewhere. This caused another two weeks of delay before testing communication from Sysmac to LD robot could even begin, due to the local Omron representative with access to the library was out of country. Omron Support did eventually help instead.

Lastly, with the main Manulab room being unavailable for a majority of the semester, there was also lack of access to various hardware required by the project, such as the Enterprise Manager that is used for controlling the LD robots as a fleet rather than individual robots. Then when the lab was finally starting to become accessible, the university started shutting down access to campus for students as of March 12 due to the COVID-19 outbreak, it never became available for testing.

5.2 Suggestions for future improvement

From the very beginning, our supervisors wanted us to come up with suggestions for improvements and future theses. The Manulab contains such a vast array of technologies that exploring all of their capabilities and possibilities would go well beyond the scope of just one bachelor's thesis. In this section we have gathered a list of some suggestions for improvements and expansions to the lab:

1. Fix communication between TM5M and LD-130CT
2. Optimisation of laser cutting
3. Efficiently extracting acrylic parts from laser cutter
4. Reducing waste from laser cutting
5. Optimisation of 3D printing
6. Find a way to separate printer part from the plate, and returning plate to printer
7. Adding a 2-axis robot to 3D-printer racks for print retrieval
8. Adding a 3D scanner to analyse printed parts
9. Changing communication protocols from stored CSVs to UDP or TCP
10. Creating a product database
11. Improving the GUI and web shop
12. Improving the digital twin
13. Finding usage for Quattro and Viper robots
14. Allowing for multiple production lines at once
15. Improving the Sysmac master program's simulation capabilities
16. Further generalise Sysmac code to more easily add new products

5.2.1 Fix communication between TM5M and LD-130CT

One big problem that was never fixed due to the COVID-19 lockdown was problematic communication between the TM5M arm on the back of the LD-130CT and the rest of the lab. The LD-130CT itself was able to communicate both with the TM5M and the rest of the network, but despite setting up port-forwarding as instructed by Omron, communication between the TM5M and the rest of the network was never successfully established. Omron was in the process of helping to solve this problem, but the lockdown prevented this from ever happening.

Considering the LD-130CT with the TM5M is by far one of the most vital parts of the Manulab, fixing this is a top priority for any future projects.

5.2.2 Optimisation of laser cutting

One part of this project that needs further development is the laser cutter. As was planned before the restructuring was to implement it into the greater lab. It was also planned to let it cut out one part of the plates and then discard the rest of it. This is inefficient and produces a lot of waste. Another part is to find the most efficient method to extract the finished cut and engraved plates.

One other part possible to add is actuators. Opening the laser cutter can be cumbersome for a robotic arm. Therefore to simplify opening the lid and front wall actuators can be installed.

5.2.2.1 Efficiently extracting acrylic parts from laser cutter

Due to lack of access to the actual laser cutter for the entirety of the project, as well as inability to use a robot near one, we were unable to even test how a robot might go about removing acrylic pieces from the laser cutter. Vacuum gripper may be possible, but it depends on how well the plate may still be attached to the acrylic plate. An alternative that could be to have an attachment point located near the cut acrylic piece that the robot has an easier time grabbing. Another option would be removing the entire cutting plate from the machine and have a separate station that handle extracting the laser-cut part.

5.2.2.2 Reducing waste from laser cutting

An unavoidable problem with the laser cutter is that a lot of waste is left over after cutting out the parts for the products, which has to be thrown away. A future project would be to find a way to use the acrylic plate as efficiently as possible in the laser cutter, possibly by using a vision system what's left of the plate then use genetic algorithms to find the best area to perform the next cut.

5.2.3 Optimisation of 3D printing

One problem with the OctoPrint Communicator script is that we never got to test it on a real printer rack in the finished Manulab, meaning unforeseen bugs could be an issue. On top of that, the code could benefit from decoupling the printer command parsing from the main part of the script, as it is currently hard-coded. Furthermore, implementing multi-threading could prove a worthwhile pursuit if the amount of printers lead to performance issues.

The kanban storage for the printer sheets holding parts should be more than just a table. A rack-system similar to what the printers themselves are placed in could be a good solution, taking advantage of the lab's vertical space. There should be some manner of separate storage for clean and used sheets.

And speaking of sheets, they need to be cleaned. Administering isopropyl alcohol and rubbing them dry should be enough for maintaining proper adhesion when printing

PLA. Hand soap and warm water should be enough for the rest. These are simple tasks for a human, but automating them may be more hassle than it is worth, as there seems to be no universal solution for all filament types. Automating it likely require a jig, a specialised tool for the robot, vision recognition of filament types, and more. One alternative would be to simply stack new and used plates in their own, kanban-style storage cells. From there, a human could quickly clean the plates and place them in the new stack when needed.

5.2.4 Find a way to separate printer part from the plate, and returning plate to printer

Two tasks we were never able to do before the COVID-19 lockdown was how to handle printers plates after they have been picked up from the printers. Separating the printed piece from the plate proved fairly difficult depending on the filament type, and returning the printer plate to the printer was simply never investigated since it wasn't a priority yet before the lockdown happened.

While the Prusa MK3S printer plates are designed to be easy to separate parts from by bending the plate, this isn't always reliable, and some cleanup tend to be needed for the parts. Making this automated may quickly become quite complicated. As for the latter problem, after cleaning has been performed, it would most likely require picking up the plate in such a way that it doesn't block the vision camera, then use vision recognition on the markings on the printer surface to place the plate in the correct place. Using custom G-code to either push the plate in place or idle the stepper motors may prove useful.

5.2.5 Adding a 2-axis robot to 3D-printer racks for print retrieval

This is the supervisors' idea, and something that Amatec is looking into. The idea is to add a mechanism to the printer racks themselves, able to extract and insert steel sheets for the printers, then delivering them directly to the conveyor robots. This would leave the mobile cobot free to do other tasks, and decrease takt time. It would be worth pursuing if the production throughput is high enough to save considerable amounts of time. Given the usual speed of FDM 3D printing, this may not be very likely.

5.2.6 Adding a 3D scanner to analyse printed parts

When 3D printing certain geometries, a lot of structure materials may be needed. Currently they have to be manually removed, but to make the lab fully automatic, a 3D-scanner could be implemented. The idea is for it to compare the printed part to its original model, then using the robots to remove supports. Doing this would require the parts themselves to be designed in a way that made removal of supports possible.

A scanner could also possibly be used to see whether the part has been printed correctly. When the parts get printed they sometimes are printed incorrectly depending on factors like the printer sheets' cleanliness and ambient temperatures. Therefore having a 3D scanner that will tell if any parts are incorrectly printed are optimal for an fully automatic manufacturing lab.

5.2.7 Changing communication protocols from stored CSVs to UDP or TCP

As mentioned previously in the report, using a mix of .CSV and Python scripts is not the best method of doing communication in a project such as this. It was chosen because it was simple and Python could handle JSON interpreting more easily than Sysmac could, allowing for more time to focus on other parts of the project. In the future, a dedicated UDP socket program with JSON interpreting should be developed for the program, allowing for more direct communication without a middle man, also resulting in less potential points of failure. As briefly mentioned in section 3.5.5, a UDP socket program was included, but was never used due to the aforementioned difficulty with JSONs.

5.2.8 Creating a product database

When expanding the lab more products will be needed to explore the capabilities of all robots and fill the web shop with a bigger variety of products. They would also need to be designed in way for the robots to easily be assembled.

This project never saw the need for a proper database, as a mere two products were enough to showcase the capabilities of most of the available equipment. In this case the product details were stored in a simple array on the IPC.

5.2.9 Improving the digital twin

After the restructuring of the project it was decided to make a simulation, a simple digital twin. This makes it easier to test robot movements and deciding where equipment should be placed. Offline programming of fleet-controlled mobile robots seems to be off the table in RoboDK, but it would be feasible to do so for some of the other robots.

5.2.9.1 Offline Programming

RoboDK allows for using post processors for offline programming of robots. These post processors are essentially Python-scripts that translate the movement instructions from RoboDK into text in a language that the robot can understand.

Most of the stationary robots in the Manulab could feasibly be offline programmed using RoboDK, but we do not think the same goes for the mobile Omron LD AMRs. All mechanical movement in RoboDK is based on stationary joints, which is not a

good fit for mobile robot kinematics. Programming the LDs in the Mobile Planner software itself is done using a map-based GUI, and while offline programming is an option here, it requires already having scanned in a map of the Manulab using an LD robot, which we were never able to do before the lockdown due to the lab being incomplete. On top of that, offline programming in Mobile Planner is only supported in version 5 and newer, but the robots we had access to were only compatible with version 4.7.7. Even if a map was obtained and there were no compatibility issues, simulating the LD robots in the program requires the Fleet Manager, which we never had access to.

5.2.10 Improving the GUI and web shop

Another part of this project was building a GUI and web shop. For this project a simple ad hoc GUI and web shop were designed in *Movicon.NExT*. This web shop was only designed to handle the two prototype products described in this thesis. Future theses can expand it and add functionality to handle a expandable amount of products.

A possible project for future theses could be to add support for custom products. For a more versatile lab, it would be desirable to have the possibility to upload custom parts, then have them manufactured and sorted with as little human interaction as possible. These parts would still need to be designed in such a way that the robots would be able to process them.

5.2.11 Find usage for the Quattro and Viper robots

This project saw no usage of the Quattro and Viper robots, as they had not yet been delivered. For most of the prototype product cases in this project they were deliberately not needed. In the future, these robots can be implemented to allow for manufacturing of more complex products.

5.2.12 Allowing for multiple production lines at once

One minor flaw with the state-based process that the main PLC program uses to keep track of production is that only one production process can be run at a time, even though it would be feasible to start a new one already from the start of assembly. This could be achieved by decoupling the assembly and delivery process from the rest of the state-based tasks and instead return to state 1000 after 4000, running it off a separate state variable with its own product variables to keep track of. Actually doing this change and testing it to ensure no issues arise was however deprioritised after the COVID-19 lockdown came into effect, both due to a lack of time and from shifting the project focus to simulation.

5.2.13 Improving the Sysmac master program's simulation capabilities

While simulating the main IPC program is possible, it is very time-consuming and problematic as mentioned in section 4.2.5, and was not prioritised before the deadline. A potential future improvement would be to make the simulation process more streamlined and easier to use in order to test improvements in the lab without having to have all equipment connected and functional.

5.2.14 Further generalise Sysmac code to more easily add new products

As mentioned in section 4.2.6, the program currently has a good capability to add new products, but it requires a few minor changes to the program. A future project would be to completely generalise the code in order to make adding new products not require any changes to the program itself. One method to fix this would be to have an external database that the program is able to get information from by using just the product ID. Fixing or bypassing Movicon's lacking capability to read arrays from Sysmac will however be more problematic.

In the current state of the program, the procedure that is required to add new a new product is shown below. Generalising would require fixing these points so they are no longer necessary.

- The cutterProductInfo, printerProductInfo and productStatus arrays in global variables are expanded from [1..2] (meaning array from 1 to 2, as those are the current product IDs) to [1..3], and have their respective information filled in the same format as for product ID 1 and 2. This is required because Sysmac seems incapable of creating arrays that can change their size.
- Due to Movicon's inability to read arrays from Sysmac; ID3NameNumber, ID3NamePrinting and ID3NameBuffer integer values need to be added to the moviconSendStatus structure.
- Continuing from the above point: In the Background program, add a new status update for Movicon similar to the ones on lines 135-143. Changes will have to be made in the GUI to support these status updates though.

The rest of the program should then accept any new product IDs, as the rest of the program is completely generic and only check against the size of the product-Info/Status arrays to see if the ID is valid, and then work off the ID.

Chapter 6

Conclusions

The purpose of this thesis was to assist in the development of the NTNU Manulab project in Ålesund, as well as create a suitable pilot project for it. Due to logistical problems surrounding the construction of the lab facilities and delivery of equipment, several of our initial goals had to be postponed indefinitely. Nevertheless, we succeeded in developing methods to make the available equipment communicate and perform according to specifications.

Due to the outbreak of the COVID-19 pandemic and the subsequent lockdown caused by it, we were unable to continue work on the thesis from March onwards. As such, the thesis was restructured towards creating software solutions, including a digital twin of the lab using RoboDK. In the end, we were able to finish most of our software and create a functioning simulation of most of the lab's available equipment. Given the circumstances, we're pleased with the results of our labour.

All in all, the project has been a valuable learning experience for all of us, teaching us a wide variety of new technical skills, as well as the importance of proper planning and logistics. We believe that in the future, the Manulab will open up new and exciting possibilities within education and research for NTNU.

References

- [1] Omron Corporation, “LD robot specifications.” <https://www.ia.omron.com/products/family/3664/specification.html>. [Online; 30-April-2020].
- [2] Omron Corporation, “TM robot specifications.” <http://www.ia.omron.com/products/family/3739/specification.html>. [Online; 1-May-2020].
- [3] NTNU, “Manulab.” <https://www.ntnu.edu/ivb/manulab>. [Online; 12-May-2020].
- [4] RoboDK, “Simulator for industrial robots and offline programmin - robodk.” <https://robodk.com/>. [Online; 25-March-2020].
- [5] Omron Corporation, “Sysmac studio overview.” <https://industrial.omron.no/no/products/sysmac-studio>, 2020. [Online; accessed 31-January-2020].
- [6] Autodesk, “AutoCAD.” <https://www.autodesk.co.uk/products/autocad/overview>. [Online; accessed 31-January-2020].
- [7] Progea, “The innovative, revolutionary and future-proof software technology for every supervision, remote control and scada/hmi application..” <https://www.progea.com/en/movicon-next/>. [Online; 13-March-2020].
- [8] RobotiQ, “Hand-E Instruction Manual.” https://assets.robotiq.com/website-assets/support_documents/document/Hand-E_TM-OMRON_InstructionManual_PDF_20191219.pdf. [Online; 06-May-2020].
- [9] NinjaTek, “NinjaFlex material.” <https://ninjatek.com/ninjaflex/>. [Online; 06-May-2020].
- [10] Prusa Research, “Original Prusa i3 MK3S to MMU2S.” https://help.prusa3d.com/en/category/original-prusa-i3-mk3s-to-mmu2s_288. [Online; 05-May-2020].
- [11] Prusa Research, “Calibration.” https://help.prusa3d.com/en/category/calibration_199. [Online; 05-May-2020].
- [12] G. H. OutsourcedGuru, “Known printer profiles for octoprint.” <https://community.octoprint.org/t/known-printer-profiles-for-octoprint/3032>. [Online; 30-April-2020].

REFERENCES

- [13] G. Häußge, “Octoprint REST API.” <https://docs.octoprint.org/en/master/api/#>. [Online; 30-April-2020].
- [14] F. S. Taklo, “OctoPrint Communicator Git Repository.” <https://github.com/Manulab2020/OctoPrintCommunicator>. [Online; 30-April-2020].
- [15] IETF, “The JavaScript Object Notation (JSON) Data Interchange Format.” <https://tools.ietf.org/html/std90>. [Online; 04-May-2020].
- [16] Omron Corporation, “Enterprise manager 2100 user guide.” <https://assets.omron.com/m/69c1134242668d6d/original/Enterprise-Manager-2100-User-Manual.pdf>. [Online; 30-April-2020].
- [17] Omron Corporation, “TM flow manual.” <https://assets.omron.com/m/66721cde51512bbc/original/TM-Software-Manual-TM-Flow-SW-Ver-1-72.pdf>. [Online; 30-April-2020].
- [18] Omron Corporation, “Cad library.” <https://industrial.omron.eu/en/services-support/support/cad-library>. [Online; 08-May-2020].
- [19] Prusa Research, “Material Guide.” https://help.prusa3d.com/en/category/material-guide_220. [Online; 05-May-2020].
- [20] Prusa Research, “PEI Print Surface Preparation.” https://help.prusa3d.com/en/article/pei-print-surface-preparation_2203. [Online; 05-May-2020].

Appendices

Appendix A

Manulab 2020 Git

<https://github.com/organizations/Manulab2020/>

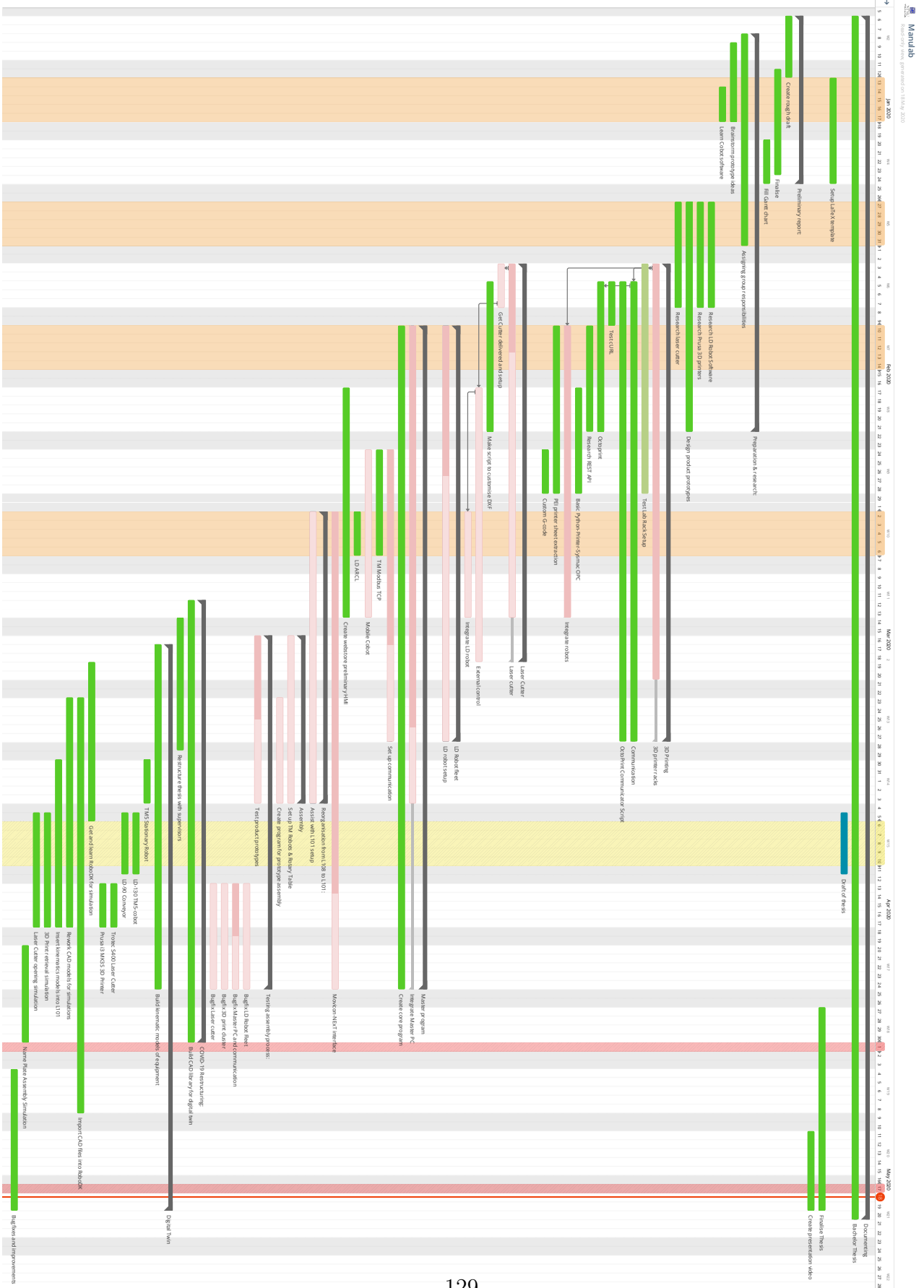
The Manulab 2020 project has its own Git, on which the source code is shared for the following:

- The IPC Master Program
- The AddTextToDXF program for laser cutting
- The OctoPrint Communicator script for 3D printing
- The Movicon-NExT HMI code

Appendix B

Gantt Chart

APPENDIX B. GANTT CHART



Appendix C

Working hours and descriptions

APPENDIX C. WORKING HOURS AND DESCRIPTIONS

Work hours & descriptions: Fredrik Sten Taklo

Week	Day	Date	Work time	Hours	Comment	Work description
Week 2	Monday	06/01/2020	08:00 - 15:00	7		Preliminary report
	Tuesday	07/01/2020	08:00 - 15:00	7		Preliminary report
	Wednesday	08/01/2020	08:00 - 12:00	4		Preliminary report
	Thursday	09/01/2020	08:00 - 16:00	8		First meetings, planning
	Friday	10/01/2020	08:00 - 14:00	6		Omron meeting, preliminary report complete
Week 3	Saturday	11/01/2020				
	Sunday	12/01/2020				
	Monday	13/01/2020	08:00 - 16:00	8		Industry 4.0 (Lean)
	Tuesday	14/01/2020	08:00 - 16:00	8		Omron meeting, set up L108, test of TM cobot
	Wednesday	15/01/2020				Industry 4.0 (Lean)
	Thursday	16/01/2020				Industry 4.0 (Lean)
	Friday	17/01/2020	08:00 - 16:00	8		Visited Amatec to look at equipment
	Saturday	18/01/2020				
	Sunday	19/01/2020				
	Monday	20/01/2020	08:00 - 16:00	8		Industry 4.0 (Lean)
Week 4	Tuesday	21/01/2020	08:00 - 16:00	8		Lean report & flowcharts finished (Industry 4.0)
	Wednesday	22/01/2020	08:00 - 16:00	8		Supervisor meeting, preliminary lab work
	Thursday	23/01/2020	08:00 - 16:00	8		Preliminary report done, made first prototype(s), got test printer
	Friday	24/01/2020	09:00 - 17:00	8		More prototyping, TM robot -> controller communication
	Saturday	25/01/2020				Set up & calibrated Prusa MK3S, started work on Octopri on a Raspberry Pi 3B
Week 5	Sunday	26/01/2020				
	Monday	27/01/2020	09:00 - 09:30	0.5		Industry 4.0 (AM)
	Tuesday	28/01/2020	16:00 - 17:00	1		Industry 4.0 (AM)
	Wednesday	29/01/2020	16:00 - 17:00	1		Prusa MK3S Prototype printing
	Thursday	30/01/2020	08:00 - 17:00	9		Octoprint setup using DHCP (WiFi)
	Friday	31/01/2020	09:00 - 18:00	9		Helped with laser cutting, Raspberry Pi network config, 3D printer troubleshooting
	Saturday	01/02/2020				
	Sunday	02/02/2020				
	Monday	03/02/2020	08:00 - 16:30	8.5		Got substitute laser cutter for testing, custom G-code for 3D printer after job, made TM landmarks
	Tuesday	04/02/2020	08:00 - 17:00	9		Writing, Research on printer UDP communication (total failure), made printer table fastener prototype
Week 6	Wednesday	05/02/2020	08:00 - 16:30	8.5		Printer/Robot vision experiments, made 3D printer fasteners, arranged Amatec delivery for Monday
	Thursday	06/02/2020	08:00 - 17:00	9		Received 2 new Prusa MK3S printers, started assembling Multi Material Extrusion Unit
	Friday	07/02/2020	09:00 - 17:30	8.5		Completed Multi Material Extrusion Unit. Apparently clogged printer extruder...
	Saturday	08/02/2020	13:00 - 16:30	3.5		Calibrated printer #2, cleaned extruder, tested different filaments (PLA + PETG)
	Sunday	09/02/2020				
Week 7	Monday	10/02/2020	08:00 - 17:00	9		Worked on printer communication (REST API, Curl), Received LD Robots
	Tuesday	11/02/2020	08:00 - 16:30	8.5		Made cables, mechanical work, set up Raspberry Pi #2, status meeting, started scripting printer comms in Python 3
	Wednesday	12/02/2020				Industry 4.0 (Digital Factory)
	Thursday	13/02/2020				Industry 4.0 (Digital Factory)
	Friday	14/02/2020				Industry 4.0 (Digital Factory)
	Saturday	15/02/2020				Industry 4.0 (Digital Factory)
	Sunday	16/02/2020				
	Monday	17/02/2020	08:00 - 17:00	9		Set up Raspberry Pi 4 Octoprint server with Power over Ethernet. Work on Python printer comms
	Tuesday	18/02/2020	08:00 - 16:30	8.5		Worked on report, Python printer comms, laser cutting, designed & printed second product prototype (dog tag)
	Wednesday	19/02/2020	08:00 - 16:30	8.5		Omron visit for help with IPC, set up UPS, Python printer comms, tried TM robot vision / z-axis compensation (no dice)
Week 8	Thursday	20/02/2020	08:00 - 16:30	8.5		More scripting, started integrating Git version control in Python/Systemc, researched flexible 3D printer filament
	Friday	21/02/2020	09:00 - 17:00	8		Tried printing using flexible TPU filament, worked on Octoprint Python Comms
	Saturday	22/02/2020				
	Sunday	23/02/2020				
	Monday	24/02/2020	08:00 - 17:30	9.5		TM cobot gripper installation, figured out python printer communication (status retrieval, file selection, job commands now working)
Week 9	Tuesday	25/02/2020	08:00 - 17:00	8		Mechanical work, cabling of LD robot. Set up another Octoprint Prusa printer. LD->TM port forwarding shenanigans

APPENDIX C. WORKING HOURS AND DESCRIPTIONS

Wednesday	26/02/2020	08:00 - 16:00	8		Work on robot communication. Made renders and new PLA table fastener for prusa printers
Thursday	27/02/2020	08:00 - 17:30	9.5		TM modbus TCP testing, network programming and exception handling in Printer OPC program
Friday	28/02/2020	09:00 - 16:30	7.5		TM modbus TCP testing, worked on getting necessary libraries for robot communication from Omron Norway
Saturday	29/02/2020				
Sunday	01/03/2020				
Monday	02/03/2020	08:00 - 17:00	9		Fixed Modbus TCP communication between NX controller and TM robots using new libraries. Refactored Python script, researched TCP comms
Tuesday	03/03/2020	08:00 - 17:00	9		Tried setting up NAT / port forwarding for Modbus TCP between main subnet & mobile TM via LD (to no avail). 3D printed some parts.
Wednesday	04/03/2020				
Thursday	05/03/2020				
Friday	06/03/2020				
Saturday	07/03/2020				
Sunday	08/03/2020				
Monday	09/03/2020	08:00 - 16:30	8.5		Fixed clogged MMU2S printer, tested TM landmarks for automatic vertical alignment, wrote and formatted/cleaned up thesis.
Tuesday	10/03/2020	08:00 - 17:30	9.5		Advisor/supplier meeting, worked with Omron on LD-TM-communication (still no success), fetched new NX1 PLC from Omron offices.
Wednesday	11/03/2020	08:00 - 17:30	9.5		Worked to get LD/TM-robot up and running again. Reset firmware. Programmed CSV-to-print function in Python OPC.
Thursday	12/03/2020	07:30 - 11:00	3.5		Cabled & set up S8BA UPS. Sent home due to Covid-19 pandemic.
Friday	13/03/2020		7		Team meeting. Set up LAN for testing HMI & Octoprint-enabled Prusa MK2 from home.
Saturday	14/03/2020				
Sunday	15/03/2020				
Monday	16/03/2020		7		NOTE: The following hours are estimates, as working from home resulted in fractured work days
Tuesday	17/03/2020		9		Team Skype meeting. Wrote thesis + Industry 4.0 report. Flight check & refactoring of code to make printer run.
Wednesday	18/03/2020		8		Advisor Skype meeting. Achieved MVP for Python script. Still has fatal bugs that must be handled.
Thursday	19/03/2020		8		Bugfixing and refactoring. Added verbosity and print-finished-commands between IPC & Python comms.
Friday	20/03/2020		8		More bugfixing. Started trying to deploy program from different machines.
Saturday	21/03/2020				Meetings, planning and gathering of CAD files for database. Finally managed to run OPC from shell commands on other PCs.
Sunday	22/03/2020				
Monday	23/03/2020		8		[Re]wrote the thesis (Introduction fleshed out, documentation regarding 3D printing begun)
Tuesday	24/03/2020		7		Got RoboDK licenses, subsequently started learning how to use it. Wrote more in thesis.
Wednesday	25/03/2020		8		RoboDK / thesis work
Thursday	26/03/2020		8		RoboDK / Thesis work. Meeting regarding digital factory / NX CAD collection and RoboDK.
Friday	27/03/2020		8		Got ahold of Amatec's conveyor robot CAD files, learned more about RoboDK
Saturday	28/03/2020				
Sunday	29/03/2020				
Monday	30/03/2020		9		Tried trimming CAD models in Fusion (not a big success), made TM's mobile station in RoboDK, started adding new fields to printer status
Tuesday	31/03/2020		8		Meeting, more CAD work, refactored Printer Python script and added new status features
Wednesday	01/04/2020		8		Created primitive CAD model of Prusa MK3S with heated, used it to create a RoboDK linear axis robot, set up LD Cobot kinematics
Thursday	02/04/2020		7		CAD work, Industry 4.0 Writing
Friday	03/04/2020		8		CAD simplification of conveyor robots, tried importing various systems into L108 stp-file
Saturday	04/04/2020				
Sunday	05/04/2020				
Monday	06/04/2020		8		Covid-19 lockdown
Tuesday	07/04/2020		8		Team meeting. Modelled and programmed double gripper setup for LD robot.
Wednesday	08/04/2020		8		Meeting. Made more detailed LD models and programmed kinematics for floor translation + rotation movement. Cobot rest position set.
Thursday	09/04/2020		8		Reworked 3D printer kinematics to work with steel sheets. Created 3D printer rack.
Friday	10/04/2020		8		Started work on RoboDK printer rack retrieval
Saturday	11/04/2020				Continued work on RoboDK printer rack retrieval
Sunday	12/04/2020				
Monday	13/04/2020		9		Covid-19 lockdown
Tuesday	14/04/2020		8		Reworked LD conveyor CAD, made table model for storage, continued working on RoboDK 3D print retrieval simulation
Wednesday	15/04/2020		8		Supervisor meeting, got new models and expanded RoboDK station
Thursday	16/04/2020		8		Created rotary table with kinematics, more work on simulating mobile robots
Friday	17/04/2020		7		Mostly more work on RoboDK. Tried finding a better solution for mobile robot targeting, but didn't succeed
Saturday	18/04/2020				Minor corrections in the thesis. Added some gripper functions to the RoboDK station.
Sunday	19/04/2020				

APPENDIX C. WORKING HOURS AND DESCRIPTIONS

Week 17	Monday	20/04/2020	8	Covid-19 lockdown	Continued working on gripper functionality. Began looking at RoboDK's Python API.
	Tuesday	21/04/2020	9	Covid-19 lockdown	Supervisor meeting. Set up L101 in RoboDK, reworked kinematics for LIDs, programmed some paths & robot movement. Thesis corrections.
	Wednesday	22/04/2020	8	Covid-19 lockdown	Bugfixing and tweaking of mechanics / kinematics. An absolute nightmare of reference frames.
	Thursday	23/04/2020	8	Covid-19 lockdown	More work on the lab in RoboDK. Printer part almost done. Worked on the thesis.
	Friday	24/04/2020	8	Covid-19 lockdown	Finished the printer retrieval simulation, put some printer-related stuff into the thesis.
	Saturday	25/04/2020			
	Sunday	26/04/2020			
Week 18	Monday	27/04/2020	9	Covid-19 lockdown	Started work on assembly simulation, CAD / kinematics and station expansions. Prettied up printer retrieval, wrote in the thesis.
	Tuesday	28/04/2020	8	Covid-19 lockdown	Continued working on assembly simulation & CAD rework. Added more about 3D printing in thesis.
	Wednesday	29/04/2020	8	Covid-19 lockdown	Finished a rather wonky assembly simulation.
	Thursday	30/04/2020	9	Covid-19 lockdown	Created charts and graphics for thesis, cleaned up simulation a bit, wrote and corrected.
	Friday	01/05/2020	8	Covid-19 lockdown	Team meeting. Gathered media for documentation, created graphics, wrote some more - albeit less than I'd hoped for!
	Saturday	02/05/2020			
	Sunday	03/05/2020			
Week 19	Monday	04/05/2020	12	Covid-19 lockdown	(Almost) completed everything related to 3D printing in thesis. Did a last batch of bugfixes and cleaned up the OPC script.
	Tuesday	05/05/2020	12	Covid-19 lockdown	(Hopefully) Completed 3D printer-related stuff in methods and results.
	Wednesday	06/05/2020	12	Covid-19 lockdown	Thesis: corrections, network setup overview, simulation sections begun, report layout changes
	Thursday	07/05/2020	12	Covid-19 lockdown	Thesis: simulation, cleanup operations
	Friday	08/05/2020	12	Covid-19 lockdown	Started adding the last models and fixes into RoboDK. Thesis: simulation and 3D printing results & discussion
	Saturday	09/05/2020	12		Writing & Gantt
	Sunday	10/05/2020	12		Thesis: mostly complete, aside from proofreading and fixes. RoboDK: Last models added, colored. AMR paths and acceleration improved.
Week 20	Monday	11/05/2020	13	Covid-19 lockdown	Began work on presentation video: finished all manuscripts, most recordings. Gathered footage.
	Tuesday	12/05/2020	12	Covid-19 lockdown	Supervisor meeting, recordings, video work. Thesis: Rewrote introduction and objective sections based on supervisor feedback.
	Wednesday	13/05/2020	12	Covid-19 lockdown	Video: Finished introduction part of presentation. Created footage for other parts.
	Thursday	14/05/2020	12	Covid-19 lockdown	Video: Finished 3D printing part. Continued on simulation. Thesis: proofreading, corrections, SKP's validation test
	Friday	15/05/2020	9	Covid-19 lockdown	Video: Finished presentation video and delivered through Teams. Thesis: more corrections, cleaned up formatting of some sections
	Saturday	16/05/2020	4		Cleaned up and exported Gantt chart. Proofread and formatted meeting reports for appendices.
	Sunday	17/05/2020			
Week 21	Monday	18/05/2020		Covid-19 lockdown	Final clean-ups. Added appendices and delivered thesis and project folder.
	Tuesday	19/05/2020		Covid-19 lockdown	
	Wednesday	20/05/2020		Deadline & Presentation	

APPENDIX C. WORKING HOURS AND DESCRIPTIONS

Work hours & descriptions: Hans-Christian Ringstad

Week	Day	Date	Work time	Hours	Comment	Work description
Week 2	Monday	2020-01-06	08:00 - 15:00	7		Preliminary report
	Tuesday	2020-01-07	08:00 - 15:00	7		Preliminary report
	Wednesday	2020-01-08			Ingen kollektiv som gikk frå Heim til NTNU	Storm
	Thursday	2020-01-09	08:00 - 16:00	8		Meeting/Preliminary report
	Friday	2020-01-10	08:00 - 14:00	6		Meeting/Preliminary report
Week 3	Saturday	2020-01-11				
	Sunday	2020-01-12				
	Monday	2020-01-13			Lean-module	Industri 4.0
	Tuesday	2020-01-14	8:00 - 16:00	8	Had a meeting with Omron in Moa	Meeting/Testing the TM-robots
	Wednesday	2020-01-15	13:30 - 15:00	1.50	Lean-module: Visited Ekornes's factory in Ikorneset/ Had a meeting with Amatec in Sykkylven	Industri 4.0/Meeting
Week 4	Thursday	2020-01-16			Lean-module	Industri 4.0
	Friday	2020-01-17	8:00 - 16:00	8	Tested IPC and Sysmac studio/Began writing in Lean-report and made process map	Testing IPC/Writing Lean-report
	Saturday	2020-01-18				
	Sunday	2020-01-19				
	Monday	2020-01-20	08:00 - 16:00	8	Finished writing Lean-report	Writing Lean-report
Week 5	Tuesday	2020-01-21	08:00 - 16:00	8	Meeting: with Paul about the 3D-Printers/ with Omron about IPC	Continue preliminary report/Meeting
	Wednesday	2020-01-22	08:00 - 15:00	7		Finished preliminary report/ Began design on product
	Thursday	2020-01-23	08:00 - 16:00	8	Omron helped with communication	Test TM-robot/Design product
	Friday	2020-01-24	08:00 - 16:00	8		Design product/Write thesis
	Saturday	2020-01-25				
Week 6	Sunday	2020-01-26				
	Monday	2020-01-27			Additiv tilvirkning-module	Industri 4.0
	Tuesday	2020-01-28	09:00 - 09:30	0.5	Additiv tilvirkning-module/Status meeting	Industri 4.0/Meeting
	Wednesday	2020-01-29			Additiv tilvirkning-module	Industri 4.0
	Thursday	2020-01-30	08:00 - 16:00	8	Meeting with omron about moviconect	Meeting/Design product
Week 7	Friday	2020-01-31	08:00 - 16:00	8		Design product/Write thesis
	Saturday	2020-02-01				
	Sunday	2020-02-02				
	Monday	2020-02-03	08:00 - 15:00	7	Researching communication	Researching Laser cutter
	Tuesday	2020-02-04	08:00 - 16:00	8	Researching format for CAD	Researching Laser cutter
Week 8	Wednesday	2020-02-05	08:00 - 16:00	8	Making script in python for dxf for names	Laser cutter
	Thursday	2020-02-06	08:00 - 16:00	8	Making script in python for dxf logo image	Laser cutter
	Friday	2020-02-07	08:00 - 16:00	8	Making script in python for dxf reading csv	Laser cutter
	Saturday	2020-02-08				
	Sunday	2020-02-09				
Week 9	Monday	2020-02-10	08:00-15:00	7	Adding function for writing to csv	Laser cutter
	Tuesday	2020-02-11	08:00-16:00	8	Adding function for writing to an error log/Status meeting	Laser cutter/Meeting
	Wednesday	2020-02-12			Digital Factory-module	Industri 4.0

APPENDIX C. WORKING HOURS AND DESCRIPTIONS

	Thursday	2020-02-13				Digital Factory-module	Industri 4.0
	Friday	2020-02-14	14:00 - 16:00	2		Digital Factory-module/Finalize	Industri 4.0/Laser cutter
	Saturday	2020-02-15	10:00 - 21:00	11		Finalized script, cleaned up internal, comments and README and made it readable, made it into an executable, unexpectedly hard to setup	Laser cutter
	Sunday	2020-02-16	10:00 - 13:00	3		Written about the script	Write thesis
Week 8	Monday	2020-02-17	08:00 - 16:00	8		Searched for a way to communicate	Laser cutter/Meeting
	Tuesday	2020-02-18	08:00 - 16:00	8		Talked to Alvøen and found out Epilog laser cutter has no external control or status checking option	Laser cutter/Design prototype 2
	Wednesday	2020-02-19	08:00 - 16:00	8		Designed product prototype 2	Design product
	Thursday	2020-02-20	08:00 - 16:00	8		Designed product prototype 2/Begin creating movicon,next	Design product/HMI/Web Client
	Friday	2020-02-21	08:00 - 17:00	9		Testing movicon	HMI/Web Client
	Saturday	2020-02-22					
	Sunday	2020-02-23					
Week 9	Monday	2020-02-24	08:00 - 16:00	8		Created multiple pages and made buttons to move between them	HMI/Web Client
	Tuesday	2020-02-25	08:00 - 16:00	8		Created more pages, made global strings and added variable to communicate to synmac studio	HMI/Web Client
	Wednesday	2020-02-26	08:00 - 16:00	8		Created more pages for status	HMI/Web Client
	Thursday	2020-02-27	08:00 - 16:00	8		Continued creating more pages for status	HMI/Web Client
	Friday	2020-02-28	08:00 - 16:00	8		Continued work on status screens	HMI/Web Client
	Saturday	2020-02-29					
	Sunday	2020-03-01					
Week 10	Monday	2020-03-02	08:00 - 16:00	8		Finished status screens, began work on order page	HMI/Web Client
	Tuesday	2020-03-03	08:00 - 16:00	8		Added script to update status	HMI/Web Client
	Wednesday	2020-03-04				Robotic-module	Industri 4.0
	Thursday	2020-03-05				Robotic-module	Industri 4.0
	Friday	2020-03-06				Robotic-module	Industri 4.0
	Saturday	2020-03-07					
	Sunday	2020-03-08					
Week 11	Monday	2020-03-09	08:00 - 15:00	7		Tried to find a way to access arrays in the tag list	HMI/Web Client
	Tuesday	2020-03-10	08:00 - 16:00	8		Status meeting, meeting with master student from Sperre / Dropped arrays, Finalized order page	Meeting / HMI/Web Client
	Wednesday	2020-03-11	08:00 - 16:00	8		Finished buffer controls, fixed misc. bugs	HMI/Web Client
	Thursday	2020-03-12	08:00 - 11:00	3		Edited info pages, began mandatory NTNU Covid-19 contingency measures	HMI/Web Client
	Friday	2020-03-13	08:00 - 16:00	8		Covid-19 contingency measures	Written in thesis
	Saturday	2020-03-14					
	Sunday	2020-03-15					
Week 12	Monday	2020-03-16	08:00-17:00	9		Covid-19 contingency measures / Made storage correction controls	HMI/Web Client
	Tuesday	2020-03-17	08:00 - 16:00	8		Covid-19 contingency measures / Updated UpdateStatus to increase performance / Extra status meeting	HMI/Web Client / Meeting

APPENDIX C. WORKING HOURS AND DESCRIPTIONS

	Wednesday	2020-03-18	08:00 - 16:30	8.5	Covid-19 contingency measures / Tried to find a solution for the slow speed while reading from tag-list	HMI/Web Client
	Thursday	2020-03-19	08:00 - 16:00	8	Covid-19 contingency measures / Find a solution using sysmac to hold all logic for the HMI	HMI/Web Client
	Friday	2020-03-20	08:00 - 16:00	8	Covid-19 contingency measures / Find a solution using sysmac to hold all logic for the HMI	HMI/Web Client
	Saturday	2020-03-21				
	Sunday	2020-03-22				
Week 13	Monday	2020-03-23	08:00 - 16:00	8	Covid-19 contingency measures / finalized storage correction controls, various bugfixes	HMI/Web Client
	Tuesday	2020-03-24	08:00 - 16:00	8	Covid-19 contingency measures / Got licenses for RoboDK and began testing it / status meeting	Written in thesis / RoboDK / Meeting
	Wednesday	2020-03-25	08:00 - 16:00	8	Covid-19 contingency measures	Written in thesis
	Thursday	2020-03-26	08:00 - 16:00	8	Covid-19 contingency measures	Written in thesis
	Friday	2020-03-27	08:00 - 16:00	8	Covid-19 contingency measures	Written in thesis
	Saturday	2020-03-28				
	Sunday	2020-03-29				
Week 14	Monday	2020-03-30	08:00 - 16:00	8	Covid-19 contingency measures / Learning to use RoboDK	Digital Twin
	Tuesday	2020-03-31	08:00 - 15:00	7	Covid-19 contingency measures / Learning to use RoboDK / status meeting	Digital Twin / Meeting
	Wednesday	2020-04-01	08:00 - 16:00	8	Covid-19 contingency measures / Learning to use RoboDK	Digital Twin
	Thursday	2020-04-02	08:00 - 17:00	9	Covid-19 contingency measures / Learning to use RoboDK, began building station for laser cutter	Digital Twin
	Friday	2020-04-03	08:00 - 16:00	8	Covid-19 contingency measures	Written in thesis
	Saturday	2020-04-04				
	Sunday	2020-04-05				
Week 15	Monday	2020-04-06	08:00 - 16:00	8	Covid-19 contingency measures / Began working on a digital twin of the trotec laser	Written in thesis/Digital twin
	Tuesday	2020-04-07	08:00 - 16:00	8	Covid-19 contingency measures / Made a digital twin of the trotec laser / status meeting	Digital twin / Meeting
	Wednesday	2020-04-08	08:00 - 16:00	8	Covid-19 contingency measures / Tried adding targets at correct spots	Digital twin
	Thursday	2020-04-09	08:00 - 16:00	8	Covid-19 contingency measures / Added temporary LD Cobot to the station	Digital twin
	Friday	2020-04-10	08:00 - 16:00	8	Covid-19 contingency measures / Added LD Cobot with double gripper to the station	Digital twin
	Saturday	2020-04-11				

APPENDIX C. WORKING HOURS AND DESCRIPTIONS

	Sunday	2020-04-12				
Week 16	Monday	2020-04-13	08:00 - 16:00	8	Covid-19 contingency measures / Tried making program for opening the laser cutter	Digital twin
	Tuesday	2020-04-14	08:00 - 16:00	8	Covid-19 contingency measures / Moved LD from left to right side due to laser cutter placement in the lab / status meeting	Digital twin / Meeting
	Wednesday	2020-04-15	08:00 - 17:00	9	Covid-19 contingency measures / Modified LD to be able to move on the floor and rotate	Digital twin
	Thursday	2020-04-16	08:00 - 16:00	8	Covid-19 contingency measures / Added targets to LD and laser cutter	Digital twin
	Friday	2020-04-17	08:00 - 16:00	8	Covid-19 contingency measures / Began programming a simulation of the laser cutter	Digital twin
	Saturday	2020-04-18				
	Sunday	2020-04-19				
Week 17	Monday	2020-04-20	08:00 - 16:00	8	Covid-19 contingency measures / Finished simulation for opening the laser cutter, began working the on closing the laser cutter	Digital twin
	Tuesday	2020-04-21	08:00 - 16:00	8	Covid-19 contingency measures / Finished simulation for closing the laser cutter / status meeting	Digital Twin / Meeting
	Wednesday	2020-04-22	08:00 - 16:00	8	Covid-19 contingency measures	Written in thesis
	Thursday	2020-04-23	08:00 - 16:00	8	Covid-19 contingency measures	Written in thesis
	Friday	2020-04-24	08:00 - 16:00	8	Covid-19 contingency measures	Written in thesis
	Saturday	2020-04-25				
	Sunday	2020-04-26				
Week 18	Monday	2020-04-27	08:00 - 16:00	8	Covid-19 contingency measures	Written in thesis
	Tuesday	2020-04-28	12:00 - 20:00	8	Covid-19 contingency measures / Meeting	Written in thesis / Meeting
	Wednesday	2020-04-29	12:00 - 20:00	8	Covid-19 contingency measures	Written in thesis
	Thursday	2020-04-30	08:00 - 20:00	12	Covid-19 contingency measures	Written in thesis
	Friday	2020-05-01	08:00 - 20:00	12	Covid-19 contingency measures	Written in thesis
	Saturday	2020-05-02	08:00 - 20:00	12		Written in thesis
	Sunday	2020-05-03	08:00 - 20:00	12		Written in thesis
Week 19	Monday	2020-05-04	08:00 - 20:00	12	Covid-19 contingency measures	Written in thesis
	Tuesday	2020-05-05	08:00 - 20:00	12	Covid-19 contingency measures / Meeting	Written in thesis / Meeting
	Wednesday	2020-05-06	08:00 - 20:00	12	Covid-19 contingency measures	Written in thesis
	Thursday	2020-05-07	08:00 - 20:00	12	Covid-19 contingency measures	Written in thesis
	Friday	2020-05-08	08:00 - 20:00	12	Covid-19 contingency measures	Written in thesis
	Saturday	2020-05-09	08:00 - 20:00	12		Written in thesis
	Sunday	2020-05-10	08:00 - 20:00	12		Written in thesis
Week 20	Monday	2020-05-11	08:00 - 16:00	8	Covid-19 contingency measures / Planned what was needed in my part	Written in thesis / Presentation video

APPENDIX C. WORKING HOURS AND DESCRIPTIONS

	Tuesday	2020-05-12	08:00 - 20:00	12	Covid-19 contingency measures / Meeting / Made manuscript and began audio recording	Written in thesis / Meeting / Presentation video
	Wednesday	2020-05-13	08:00 - 20:00	12	Covid-19 contingency measures / Completed audio recording and began making slides	Written in thesis / Presentation video
	Thursday	2020-05-14	08:00 - 20:00	12	Covid-19 contingency measures / Finished slides / Added copyrights	Written in thesis / Presentation video / HMI
	Friday	2020-05-15	08:00 - 16:00	8	Covid-19 contingency measures	Written in thesis
	Saturday	2020-05-16	08:00 - 16:00	8		Written in thesis
	Sunday	2020-05-17				
Week 21	Monday	2020-05-18			Covid-19 contingency measures / Gather appendices and deliver project	Written in thesis
	Tuesday	2020-05-19			Covid-19 contingency measures	
	Wednesday	2020-05-20			Covid-19 contingency measures / Present project on Zoom	Presentation day! / Deadline 12:00

APPENDIX C. WORKING HOURS AND DESCRIPTIONS

Work hours & descriptions: Hans Christian Haugan Frimson

Week	Day	Date	Work time	Hours	Comment	Work description
Week 2	Monday	06/01/2020	08:15 - 14:30	6h15m		Started preliminary report
	Tuesday	07/01/2020	08:10 - 15:00	6h50m		Continuing with preliminary report
	Wednesday	08/01/2020	08:15 - 12:00	3h45m		Finished as much of the preliminary report as we can
	Thursday	09/01/2020	08:10 - 16:10	8h0m		Meeting with advisors, project planning
	Friday	10/01/2020	08:05 - 14:05	6h0m		Meeting with Omron, finishing draft of preliminary report
	Saturday	11/01/2020				
	Sunday	12/01/2020				
Week 3	Monday	13/01/2020			Industri 4.0	
	Tuesday	14/01/2020	08:10 - 15:30	7h20m		Working on Gantt chart, meeting with Omron, setting up temporary workplace, teach group how to use TM cobot
	Wednesday	15/01/2020			Industri 4.0	
	Thursday	16/01/2020			Industri 4.0	
	Friday	17/01/2020	08:15 - 16:15	8h0m		Set up IPC in lab, installed software, started writing on Lean report, created product prototype in Fusion 360
	Saturday	18/01/2020				
	Sunday	19/01/2020				
Week 4	Monday	20/01/2020	08:10 - 16:00	7h50m		Finished Lean report
	Tuesday	21/01/2020	08:30 - 16:00	7h30m		Continued on preliminary report, worked on gantt chart, meeting with Paul and teamviewer session with Torbjørn
	Wednesday	22/01/2020	08:10 - 15:10	7h0m		Finished preliminary report, started printing and cutting physical prototype for product
	Thursday	23/01/2020	08:15 - 16:00	7h45m		Tweaked prototype and printed a new version, Omron helped setting up some communication
	Friday	24/01/2020	08:15 - 16:00	7h45m		Further tweaked prototype, researched controlling LD robot and TM robot from Sysmac, wrote on the report
	Saturday	25/01/2020				
	Sunday	26/01/2020				
Week 5	Monday	27/01/2020			Industri 4.0	
	Tuesday	28/01/2020			Industri 4.0	
	Wednesday	29/01/2020			Industri 4.0	
	Thursday	30/01/2020	08:15 - 16:00	7h45m		Testing new prototype brackets, help with Sysmac and Moxicon with Omron, some more setup in the lab
	Friday	31/01/2020	08:20 - 16:00	7h45m		Tested new bracket versions, helped with further continuing design, wrote on main report, group meeting
	Saturday	01/02/2020				
	Sunday	02/02/2020				
Week 6	Monday	03/02/2020	08:10 - 16:00	7h50m		Group meeting, helping with modifications to the brackets, worked on Industri 4.0 tasks
	Tuesday	04/02/2020	10:00 - 17:00	7h0m		Woke up late and stayed at home due to sickness, wrote on report and made renderings of name plate, worked on Industri 4.0 tasks
	Wednesday	05/02/2020	08:15 - 16:00	7h45m		Added images to report, made programs on TM robot to test vision, further writing on report
	Thursday	06/02/2020				
	Friday	07/02/2020	08:10 - 16:00	7h50m		Wrote on the report, helped unpack the new 3D printers, observed work for other group members, waiting for LD robot
	Saturday	08/02/2020	08:20 - 16:00	7h40m		Worked on report, researched exporting Sysmac variables to csv, group meeting, waiting impatiently for LD robot
	Sunday	09/02/2020				
Week 7	Monday	10/02/2020	08:10 - 16:00	7h50m		Continued working on exporting Sysmac variables to csv, LD robots arrived, attempted basic setup but couldn't continue without license key
	Tuesday	11/02/2020	08:10 - 16:00	7h50m		Tested Sysmac Variable export, confirmed it worked on NX102 but not on NY512. Updated TMFlow on robot arm, but struggling with update on NY IPC.
	Wednesday	12/02/2020	15:15 - 16:15	1h0m	Industri 4.0	Tried to fix yesterday's Sysmac issue by enabling Virtual SD Card. Unsuccessful, asking Omron for help.
	Thursday	13/02/2020			Industri 4.0	
	Friday	14/02/2020			Industri 4.0	
	Saturday	15/02/2020				

APPENDIX C. WORKING HOURS AND DESCRIPTIONS

	Sunday	16/02/2020				Connected to LD robot and set it up. Created maps, tested movements and tweaked settings for it, set up goals and tested them.
Week 8	Monday	17/02/2020	08:10 - 16:10	8h0m		Emailed Omron about ARCL library
	Tuesday	18/02/2020	08:10 - 16:00	8h0m		Continued setup of LD robot, made a new map after changing room layout, set up meeting with Omron, wrote on the report.
	Wednesday	19/02/2020	08:10 - 16:10	8h0m		Visited Omron to update IPC firmware (unsuccessful) and fix Virtual SP Card problems (successful), tested new vacuum gripper on TM robot, finished csv export program
	Thursday	20/02/2020	08:05 - 16:10	8h5m		Started working on master program. Finished csv write and csv read functions. Added TM robot communication and UDP socket made by Omron. Added Git functionality.
	Friday	21/02/2020	09:00 - 17:00	8h0m		Continued working on master program, fixed bug with csv write- and read functions, unsuccessfully tried to find cause of EC_Indataivaldler error
	Saturday	22/02/2020				
	Sunday	23/02/2020				
	Sunday	24/02/2020				
Week 9	Monday	25/02/2020	08:10 - 16:10	8h0m		Installed hand gripper on LD-130CT's TM robot arm, made program using it to pick up printer plate (previously only done with vacuum), wrote on report, emailed amatec
	Tuesday	26/02/2020	08:05 - 16:10	8h5m		Improved TM program from yesterday, status meeting, master program uploaded to Github, fixed wifi connectivity to robot arm on LD robot, set up communication between LD-130CT and the TM robotarm on its back, tried making it accessible on the rest of the network
	Wednesday	27/02/2020	08:10 - 16:10	8h0m		Researched setting up Modbus on LD's TM robot, continued working on master program, added queue and worked on importing/exporting variables
	Thursday	28/02/2020	08:10 - 17:10	9h0m		Tried adding ability to read multiple lines of csv in program, spent most of the day troubleshooting, sent email to Omron asking for help
	Friday	29/02/2020	08:10 - 16:10	8h0m		Got help from Omron with yesterday's problems, a lot of bugfixing and testing afterwards, finally finished multi-line csv reading, updated Github with new and improved program. End-of-week meeting.
	Saturday	01/03/2020				
	Sunday	02/03/2020				
Week 10	Monday	03/03/2020	08:15 - 16:15	8h0m		Helped set up communication with TM robot, added functionality to write commands to printer, merged master program with Fredrik's TM Support program to fix incompatibility issues and add communication with the TM robot.
	Tuesday	04/03/2020	08:10 - 16:10	8h0m		Set up variables to export to Movicon and created structs + conversion methods for them. Also started working on setting up communication to LD robot, but ran into difficulties setting up ARCL server. Solved it in the end by resetting the robot.
	Wednesday	05/03/2020				Industri 4.0
	Thursday	06/03/2020				Industri 4.0
	Friday	07/03/2020				
	Saturday	08/03/2020				
	Sunday	09/03/2020				
Week 11	Monday	10/03/2020	08:15 - 16:15	8h0m		Made prototype program on mobile TM robot to pick from different heights and to see how generic it could be made. Wrote about .csv reading scripts in the master program on the report.
	Tuesday	11/03/2020	08:10 - 16:55	8h45m		Meeting with advisors, meeting with Omron, watched hardware get loaded into the Manulab room, got help from Omron regarding the LD robots, worked on master program, wrote on report, tried troubleshooting new LD robot wifi connectivity problem
	Wednesday	12/03/2020	08:10 - 16:10	8h0m		Expanded master program, reworked states, wrote .txt explaining states, added more communication variables to Movicon
	Thursday	13/03/2020	08:10 - 11:00	2h50m		Continued working on master program, added buffer capability to program. University closed due to corona outbreak
	Friday	14/03/2020		--7h		Worked on master program from home, added retrieval of finished prints, started restructuring the program by putting states and periodic background tasks in different ST programs. Wrote on thesis
	Saturday	15/03/2020				
	Sunday	15/03/2020				

APPENDIX C. WORKING HOURS AND DESCRIPTIONS

Week 12	Monday	16/03/2020		~8h	Covid-19 contingency measures	Group meeting, finished Industry 4.0 worked, continued generalizing and restructuring program, adding functions, etc
	Tuesday	17/03/2020		~7h	Covid-19 contingency measures	Meeting with advisors, continued expanding and reworking master program, variables, functions, etc
	Wednesday	18/03/2020		~8h	Covid-19 contingency measures	Further improved master program. More variable rework, moved csv reading/refresh to background tasks, improved startup phase with error-checking, future-proofed program to allow for new product IDs without breaking, updated printer>Status format, updated program status tk.
	Thursday	20/03/2020		~7h	Covid-19 contingency measures	Expanded program that connects to LDs, including error checks. Reworked automated buffer restock, investigating ways to generalize it for any product ID. Made some changes to how active prints are saved, and started looking into updating it based on the name of the code that is currently being printed.
	Friday			~9h	Covid-19 contingency measures	Several meetings with group advisors, more decisions for what to do and how to proceed. Gathered CAD models for simulation. Made function that can parse product ID and number from specific print code names, and implemented it into the program.
	Saturday	21/03/2020				
	Sunday	22/03/2020				
Week 13	Monday	23/03/2020		~8h	Covid-19 contingency measures	Did some minor tweaks throughout the master program, added more documentation, updated documentation/commenting, tried communicating between Sysmac simulation and Movicon, wrote on report
	Tuesday	24/03/2020		~8h	Covid-19 contingency measures	Meeting with advisors, installed RoboDK, made some minor changes and added more comments to master program code, wrote on report
	Wednesday	25/03/2020		~8h	Covid-19 contingency measures	Some improvements to master program, wrote on report, started moving certain parts I've written from Methods to Results and filling in blanks.
	Thursday	26/03/2020		~8h	Covid-19 contingency measures	Progress update on 3D models and simulations, minor improvements to program, wrote on report, made flowchart
	Friday	27/03/2020		~8h	Covid-19 contingency measures	Wrote on report, some programming on the side, started looking at RoboDK
	Saturday	28/03/2020				
	Sunday	29/03/2020				
Week 14	Monday	30/03/2020		~7h	Covid-19 contingency measures	Worked on integrating LDs into master program, but without being able to test it, functionality remains limited. Minor improvements here and there in the program.
	Tuesday	31/03/2020		~8h	Covid-19 contingency measures	Moved large parts of text I wrote in Results to Methods to fit with advisor's layout suggestion, misc programming in Sysmac, continued working on LD tasks, started testing RoboDK
	Wednesday	01/04/2020		~8h	Covid-19 contingency measures	Finished as much as I'm able to do for LD commands right now without being able to test them. Minor master program improvements. Continued testing with RoboDK. Wrote on report
	Thursday	02/04/2020		~8h	Covid-19 contingency measures	Finished final report for Industry 4.0
	Friday	03/04/2020		~8h	Covid-19 contingency measures	
	Saturday	04/04/2020				
	Sunday	05/04/2020				
Week 15	Monday	06/04/2020		~8h	Covid-19 contingency measures	Group meeting, more testing with RoboDK, misc programming, more additions to LD commands, started working on error handling in master, wrote a bit on report
	Tuesday	07/04/2020		~8h	Covid-19 contingency measures	More work on error handling, expanded LD functionality, some progress towards TM functionality, more state instructions.
	Wednesday	08/04/2020		~4h	Covid-19 contingency measures	Reduced working time because easter preparations. Bugfixed master program, fixed some for-loops, better consistency and state-processes in master.
	Thursday	09/04/2020		~8h	Covid-19 contingency measures	Misc program improvements, better startup, better accounting for time it takes to read statuses, waiting for initial read to complete, etc. Wrote on report
	Friday	10/04/2020		~8h	Covid-19 contingency measures	Reworked LD functions from scratch due to problems with repeatability. Added basic TM functionality, although not much more can be done due to being unable to make anything in TMFlow without hardware access. Group meeting
	Saturday	11/04/2020				
	Sunday	12/04/2020				
Week 16	Monday	13/04/2020		~9h	Covid-19 contingency measures	Improved LD functions to be fully repeatable, simulated to ensure there were no further problems. Reduced number of variables used by 50%. Some more testing with RoboDK.

APPENDIX C. WORKING HOURS AND DESCRIPTIONS

	Tuesday	14/04/2020	~9h	Covid-19 contingency measures	Meeting with advisors. Finalised LD functions (to the extent that's currently possible). Improved function for processing gcode name into ID and number. More status info sent to movicon, like master state and actual LD status. Various bugfixes and improvements to Background in master.
	Wednesday	15/04/2020	~8h	Covid-19 contingency measures	Wrote on report. Improved function for processing gcode name to be more robust. Added ability to send TM status to Movicon.
	Thursday	16/04/2020	~8h	Covid-19 contingency measures	Various other improvements to program.
	Friday	17/04/2020	~8h	Covid-19 contingency measures	Minor changes in sysmac program, mainly wrote on report regarding master program, LDs and csv.
	Saturday	18/04/2020			Primarily wrote on report
	Sunday	19/04/2020			
	Monday	20/04/2020	~8h	Covid-19 contingency measures	Expanded and completed TM functionality in Sysmac. Updated device settings to allow TM functions to be on the same device as the rest of the program. Completed as much as I can of the Sysmac program.
Week 17	Tuesday	21/04/2020	~8h	Covid-19 contingency measures	Status meeting with supervisors, wrote on report
	Wednesday	22/04/2020	~8h	Covid-19 contingency measures	Wrote on report
	Thursday	23/04/2020	~6h	Covid-19 contingency measures	Wrote on report, discussed report structure and presentation with group
	Friday	24/04/2020	~8h	Covid-19 contingency measures	Wrote on report, group meeting
	Saturday	25/04/2020			
	Sunday	26/04/2020			
Week 18	Monday	27/04/2020	~12h	Covid-19 contingency measures	Reworked CSV communication text on report, added flowcharts
	Tuesday	28/04/2020			
	Tuesday	29/04/2020	~8h	Covid-19 contingency measures	Advisor meeting, made a few fixes in sysmac program, made some visual explanations of program, wrote on report, prepared Industry 4.0 exam PDF
	Wednesday	30/04/2020	~8h	Covid-19 contingency measures	Wrote on report, reworked flowcharts, added more text for TM robots
	Thursday	30/04/2020	~9h	Covid-19 contingency measures	Wrote on report, made tables of specifications, greatly expanded text on LD and TM robots
	Friday	01/05/2020	~8h	Covid-19 contingency measures	Group meeting, wrote on report, added table for TM specifications, wrote more about TM robot, started planning presentation
	Saturday	02/05/2020			
	Sunday	03/05/2020			
Week 19	Monday	04/05/2020	~9h	Covid-19 contingency measures	Wrote on report, added more to results, minor fixes on sysmac program
	Tuesday	05/05/2020	~9h	Covid-19 contingency measures	Continued to write on report, adding text here and there
	Wednesday	06/05/2020	~8h	Covid-19 contingency measures	Wrote on report, finished up work on sysmac program, slight distraction due to sudden job searching opportunity
	Thursday	07/05/2020	~9h	Covid-19 contingency measures	Wrote on report, started on Lean section, expanded various previously written sections
	Friday	08/05/2020	~10h	Covid-19 contingency measures	Wrote on report, expanded Lean section
	Saturday	09/05/2020	~5h		Some writing, but largely distracted throughout the day, will be more focused tomorrow
	Sunday	10/05/2020	~12h		Wrote on report, finished Lean section, wrote about simulation abilities in Sysmac, few more additions here and there, added more comments to PLC program variables, finalised(?) PLC program
Week 20	Monday	11/05/2020	~10h	Covid-19 contingency measures	A few improvements to text written in project report, started working on presentation
	Tuesday	12/05/2020	~10h		
	Wednesday	13/05/2020	~10h	Covid-19 contingency measures	Generalised sysmac program to support all possible product IDs, added text about it in report, more writing on presentation
	Thursday	14/05/2020	~10h	Covid-19 contingency measures	Recorded presentation audio, worked on presentation video
	Friday	15/05/2020	~8h	Covid-19 contingency measures	Proofread report, worked on images for the last two slides of the robot part
	Saturday	16/05/2020			Completed presentation, shorter day due to apartment viewing
	Sunday	17/05/2020			
Week 21	Monday	18/05/2020		Covid-19 contingency measures	
	Tuesday	19/05/2020		Covid-19 contingency measures	
	Wednesday	20/05/2020		Covid-19 contingency measures	Presentation day

Appendix D

Progress Meeting Reports

Progress meeting reports

Week 3 progress meeting

Friday:

DONE THIS WEEK:

- LaTeX template created
- IPC, LAN and power set up in room L108
- Preliminary report “Beta”
- Product case 1 (Acrylic glass + PLA name sign) designed
- Meeting with suppliers: Omron & Amatec

TODO NEXT WEEK:

- Write report for Lean module (Industry 4.0 subject)
 - Write strategically so that we can reuse assets for main thesis
 - Flow charts / Process mapping
- Complete preliminary project report
- See if a laser cutter can be moved to L108 for testing
- Delivery of LD AMRs? Possibly, but not likely.
- **Start work on the main thesis. LaTeX template is ready for writing.**

Week 4 progress meeting

Monday:

TODO THIS WEEK:

- Finish the lean report
- Finish the preliminary report
- Begin writing bachelor thesis
- Hopefully move laser cutter to L108
 - If there is time: look into laser cutter software
- Negotiate dates for supervisor meetings
- Learn about IPC-controller communication from Omron

Friday:

DONE THIS WEEK:

- Lean report completed and submitted
- Preliminary report is assumed to be finished
- Work on thesis begun
- Supervisor meetings are to be held biweekly, starting next Tuesday
 - They collide with Industry 4.0 lessons!
- With help from Omron, managed to connect TM robots to NX1 PLC
 - IPC has outdated firmware. Omron will have to return to fix it.
- "Name tag/sign" product prototype completed
 - May require further revisions to simplify assembly process
- **LASER CUTTER CANNOT BE MOVED TO L108.** Not feasible, according to lab chief.
 - Alternative solutions are needed if we're going to test communications in time.

TODO NEXT WEEK:

- Find out if laser cutter in the Electro Workshop (basement) can be used temporarily instead
 - If so, test communication and interfacing using AutoCAD, Fusion etc.
- LD AIV robots are supposed to arrive. Set them up in L108's workspace.
- More prototyping
- Make OctoPrint work properly with the temporary Prusa Mk3

Week 5 progress meeting

Monday:

TODO THIS WEEK:

- Monday meeting skipped due to colliding with Industry 4.0 lectures

Friday:

DONE THIS WEEK:

- Industry 4.0: Additive manufacturing
- Written in thesis
- Continued design on product 1 (Desktop Name Sign)
- **LD-ROBOTS SHOULD HAVE BEEN DELIVERED THIS WEEK, WHICH IS NOT THE CASE**
- Raspberry Pi got static IP address
 - Basic OctoPrint up and running on test printer
 - Very inconsistent print quality on borrowed Prusa i3 MK3S
- Meeting with Omron to get started with Movicon.NExT and IPC-PLC communication

TODO NEXT WEEK:

- **GET LD-ROBOTS DELIVERED**
- Begin testing LD-Robots
- Get two Prusa 3D-Printers ordered by supervisor
- Assemble/set up and begin testing 3D-Printers
- Research Laser Cutter

Week 6 progress meeting

Monday:

TODO THIS WEEK:

- **LD-ROBOTS SHOULD BE DELIVERED THIS WEEK**
 - **The robots are overdue**
 - **Begin testing LD-Robots**
- Research Laser cutter
- Get two Prusa 3D-Printers ordered by supervisor
- Begin testing 3D-Printers
 - Test Raspberry Pi 4 with OctoPrint

Friday:

DONE THIS WEEK:

- **LD-ROBOTS STILL NOT DELIVERED**
 - **NEW DELIVERY DATE MONDAY (10.02.20)**
- Tested vision with the TM-robots on the 3D-printer steel sheet
- Two Prusa i3 MK3S 3D printers delivered
 - One Multi-material extrusion add-on included
 - Finished assembled
- Written in thesis
 - Made rendering of product prototypes
- Researched laser cutter
 - Communication
 - Made script to insert text and image to DXF
- Communication between programs are to be carried out by reading and writing to csv files

TODO NEXT WEEK:

- **LD-ROBOTS SHOULD BE DELIVERED NEXT WEEK**
 - **SETUP LD-ROBOTS IN L108**
 - Very limited space. There may not be room for more than one. Prioritise.
- Experiment with OctoPrint REST API for 3D printer network communication
- Continue work on scripts to insert text and image to DXF and add communication
- Work on thesis and prototypes

Week 7 progress meeting

Monday:

TODO THIS / NEXT WEEK:

- Industry 4.0: Digital factories will take most of this weeks' time
- **LD-ROBOTS SHOULD BE DELIVERED TODAY (Monday)**
 - Two were delivered, but without license keys. Can't be programmed?
 - Acquire MobilePlanner 5 to be able to create basic programs without license
- Experiment with OctoPrint REST API for communication
- Continue work on scripts to insert text and image to DXF and add communication

Friday:

DONE THIS WEEK:

- Internal team meeting skipped due to Industry 4.0 lectures
- Acquired two LD AMRs: one equipped with conveyor, the other one with a TM5M cobot arm
- Programming of scripts

Week 8 progress meeting

Monday:

TODO THIS WEEK:

- Test of LD-robots
 - See if security settings / safety switch position for the TM5M arm needs to be changed
- Investigate and decide which laser cutter to use for development
 - Maybe choose older, less compatible machine if intended cutter does not get delivered
- Make script to read status of 3D printers
 - Try setting up Power-over-Ethernet switch with the Raspberry Pi 4B.
- Design product prototype 2
- Incorporate script AddTextToDXF with IPC

Friday:

DONE THIS WEEK:

- Designed product 2: “dog tag”, a key chain accessory with customizable name
 - Obtained flexible TPU filament (*NinjaFlex*)
- Talked to Alvøen and found out Epilog laser cutter has no external control or status checking options
 - Come up with ways to use robotic arms to start and stop laser cutting process
- Set up LD-robots and maps
- Started work on Sysmac master program
 - Created program to read and write CSV files in Sysmac studio
 - Tried adding Git to master put run into firmware problems
- Started work on HMI and Web Client in Movicon.NEXT
- OctoPrint Python “OPC” script
 - Read data from 3D printers
- Visited Omron’s offices at Moa
 - Obtained Uninterruptible Power Supply
 - Omron unable to update firmware due to its age. This could get problematic.
 - Fixed SD memory card issues
- Incorporate script AddTextToDXF in IPC
- Made plan for static IP address reservations for equipment, created a table

TODO NEXT WEEK:

- Continue work on HMI, Web Client, Sysmac master program, OctoPrint Python script
- Test if LD-Robot can lift the 3D printer plates of the 3D printers
 - Find ways to use the arm on the robot with different grippers
 - Test if a LD-Robot with conveyor can be used to help

Week 9 progress meeting

Monday:

TODO THIS WEEK:

- Continue work on HMI and Web Client
- Continue work on Sysmac master program
- Test if LD-Robot can lift the 3D printer plates of the 3D printers
 - Find a way to use the arm on the robot with different grippers
 - Test if a LD-Robot with conveyor can be used to help
- OctoPrint Python OPC
 - Start prints

Friday:

DONE THIS WEEK:

- HMI and Web Client
 - Status screen almost finished and has begun work on an order screen
 - Learned how to make HMI a simple in Movicon.NExT
 - Troubleshooting
- OctoPrint Python OPC
 - All basic communication working
 - Read status fields using HTTP + JSON
 - Write commands using Python script (JSON formatted HTTP requests)
 - Created simple code for printing G-code stored on Raspberry Pi
 - Troubleshooting
- Sysmac Master program
 - Added queue function
 - Working on communication
 - TM-Robot
 - 3D Printer
 - Added capability to read multiple csv files
 - Troubleshooting
- LD-Robot
 - Installed gripper
 - Made possible to communicate with robot arm
 - Tested if arm could lift plate from printer with a gripper
- Version control implemented for all programs

TODO NEXT WEEK:

- Industry 4.0
- Sysmac Master program
 - Add more functionality to the program
- OctoPrint Python OPC
 - Add functionality to communicate with Sysmac/Movicon
- HMI and Web Client
 - Finish Status screen
 - Continue work on Order screen

Week 10 progress meeting

Monday:

TODO THIS / NEXT WEEK:

- Industry 4.0 lectures will take up most of the week
- Sysmac Master program
 - Add more functionality to the program
- OctoPrint Python OPC
 - Add functionality for CSV-based communication with Sysmac/Movicon
- HMI and Web Client
 - Finish Status screen
 - Continue work on Order screen

Friday:

DONE THIS WEEK:

- Meeting skipped due to Industry 4.0 lectures
- Got Modbus TCP up and running on stationary TM5 Robots on local subnet

Week 11 progress meeting

Monday:

TODO THIS WEEK:

- Write in thesis
- 3D-printers
 - Add TM-landmark
- LD-Robot
 - Add prototype program for TM robot arm
- Add communication between programs
- HMI/Web Client
 - Fix arrays
 - Create a web client
 - Finalize order page
- Work on main Sysmac program
- Meeting with Omron to update LD-Robots

Friday:

DONE THIS WEEK:

- Began mandatory NTNU Covid-19 contingency measures on Thursday. All students sent home.
- 3D Printing
 - Script updated, and new test station set up at home using private Prusa i3 MK2
- Meeting with Omron and Amatec on Tuesday
 - Omron technician upgraded LD-Robots' firmware
 - Update bricked the TM5M-equipped robot
 - We reinstalled old firmware (for now)
- HMI/Web Client
 - Dropped arrays and uses struct instead
 - Finished order page
 - Finished buffer controls
 - Tested Web Client on Windows 7
- Master program
 - Cleaned up program
 - Made .TXT for documentation
 - Changed communication to Movicon to only structs and not arrays
 - Added buffer controller for Movicon
 - Added function to fill buffer automatically
 - Separated master program to states and background programs
- Written in thesis

TODO NEXT WEEK:

- Write in thesis and document work done thus far
- Find out what can be done from home other than writing in thesis
 - Discuss with supervisors!
- Program what we can from home
 - Refactor and clean up code
 - Test and simulate where applicable

Week 12 progress meeting

Monday:

TODO THIS WEEK:

- Write in thesis
- Find out what can be done from home other than writing in thesis
 - Discuss with supervisors
- Program what we can from home
 - Refactor and clean up code
 - Test and simulate where applicable
- Make OctoPrint Communicator Python script deployable on any PC

Friday:

DONE THIS WEEK:

- Adjustments due to Covid-19 outbreak.
 - Many original objectives will have to be scrapped completely. Restructuring needed.
- Currently researching ways to simulate the different lab processes.
- Worked on programs/script from home
 - Done testing with Movicon on local machine, needs further testing with the IPC
- Had two extra status meetings to decide further developments in the project
 - We may possibly try creating a digital twin of the lab using RoboDK
- Deployed OctoPrint Communicator on a 3rd party machine through command line

TODO NEXT WEEK:

- Write more in thesis
- Start looking at simulation software. Integrate CAD-models from suppliers.
- Continue working on programming where applicable.

Week 13 progress meeting

Monday:

TODO THIS WEEK:

- Write more in thesis
- Start looking at simulation software RoboDK.
 - Get licenses from lab engineer and start integrating CAD-models from suppliers.
- Continue working on programming where applicable.

Friday:

DONE THIS WEEK:

- Written in thesis
 - Documented the Movicon GUI with images
- Tried using RoboDK
 - Installed and began testing
 - Got CAD-models for some robots
- Misc. programming
 - Attempted communicating between Movicon and simulated Sysmac on private PC; failed

TODO NEXT WEEK:

- Write more in thesis
 - Ask about preferred structure going forward
- Test and learn more about RoboDK
 - Continue cooperating with supervisor to gather a library of CAD files. Import into RoboDK.
- Misc. programming

Week 14 progress meeting

Monday:

TODO THIS WEEK:

- Write more in thesis
 - Ask about preferred structure
- Test more RoboDK
 - Cooperate with supervisor to get the CAD-files into RoboDK
- Misc. programming

Friday:

DONE THIS WEEK:

- Written in thesis
- Created some basic RoboDK programs with kinematics for testing purposes
- Added 3D models to RoboDK
 - Simplified models for better performance when running the program
 - Includes LD90 with Conveyor
 - LD130 with TM5
 - Some of the grippers
 - Prusa i3 MK3S + steel sheet
- Added LD functions to master program

TODO NEXT WEEK:

- Write more in thesis
- Continue work with RoboDK
 - Implement double gripper on LD-Cobot. Simulate 3D printer sheet retrieval.
- Misc. programming
- Adjust the gantt chart

Week 15 progress meeting

Monday:

TODO THIS WEEK:

- Write more in thesis
- Continue work with RoboDK
 - Implement double gripper on LD-Cobot. Simulate printer retrieval.
- Misc. programming
- Adjust the gantt chart

Friday:

DONE THIS WEEK:

- Written in thesis
- Worked with RoboDK
 - Worked on kinematics
 - Finished simple laser cutter
 - Added to 3D printers
 - Added to LD robots
 - Reworked / simplified 3D models in Fusion360
 - CAD models for LD robots
 - TM robots
 - 3D printers (created from scratch)
- Misc. Programming
 - Sysmac
 - Expanded LD functionality
 - Added basic TM functionality
 - Misc. improvements to program states and background processes
 - Added and improved error handling
 - 3D printers
 - Added more settings, external controls for 3D printers
- Adjusted the gantt chart

TODO NEXT WEEK:

- Write more in thesis
- Discuss possibilities for returning to campus with supervisors
- Continue work with RoboDK
- Continue work on improving code

Week 16 progress meeting

Monday:

TODO THIS WEEK:

- Write more in thesis
- Discuss with supervisor whether we can or should return to campus
- Continue work with RoboDK
- Continue work on improving code

Friday:

DONE THIS WEEK:

- Written in thesis
- Continued work with RoboDK
 - Worked on simulating the simple mobile robot opening the laser cutter
 - Made temporary material storage
 - Imported rotary table and added kinematics
 - Continued work on printer retrieval station
 - Sent an email to RoboDK to ask for help to improve our mobile robot kinematics
- Continued work with improving code
 - Various improvements and bugfixes throughout the program
 - Made G-code name interpreter more robust
 - Finished up almost everything that can be done without access to lab. Only some functions for TM cobots remain.

TODO NEXT WEEK:

- Write more in thesis
- Continue work with RoboDK
 - Finish simulation of the simple mobile robot opening the laser cutter
 - Finish up 3D printer sheet retrieval simulation
- Continue work on improving code
 - Finish functions for TM cobot arms

Week 17 progress meeting

Monday:

TODO THIS WEEK:

- Write more in thesis
- Continue work with RoboDK
 - Finish simulation of the simple mobile robot opening the laser cutter
 - Finish 3D printer retrieval functions
- Continue work on improving code
 - Finish functions for TM cobot arms

Friday:

DONE THIS WEEK:

- Written more in thesis
- RoboDK
 - Finished simulation of opening and closing laser cutter
 - Finished simulation of 3D printer plate retrieval
- Done more improvements in code
 - Completed functionality as far as possible in the current environment
 - Expanded TM robot function

TODO NEXT WEEK:

- Write more in thesis
- RoboDK
 - Finish simple assembly simulation
- Begin planning the presentation video
- Send copy of thesis to supervisors

Week 18 progress meeting

Monday:

TODO THIS WEEK:

- Write more in thesis
- RoboDK
 - Finished assembly simulation for Desktop Name Plate prototype product
- Begin planning the presentation video
 - Write manuscript together using Overleaf
 - Everyone records a section detailing their own work
 - Finsson & Taklo puts the video together using Sony Vegas
- Send copy of thesis to supervisors

Friday:

DONE THIS WEEK:

- Written in thesis
- Almost finished with simulation in RoboDK
 - Missing the last 3D models.
 - Joint and linear speeds/acceleration for robots need tweaking.
 - AMR paths will likely need to be reworked after last models are put in place.

TODO NEXT WEEK:

- Write more in thesis, hopefully get it done
- Begin work on making the presentation video
- Add last 3D models in the simulation and perform final required tweaks

Week 19 progress meeting

Monday:

TODO THIS WEEK:

- Finalize thesis
- Work on making the presentation video
- Add last 3D models in the simulation, tweak the robots' speed and acceleration in accordance with their data sheets

Friday:

DONE THIS WEEK:

- Almost finalized thesis
 - Proofreading and formatting complete
 - Only appendixes remain
- Finished presentation video
- Made final adjustments to Sysmac program
- Made final adjustments to the project in Movicon
- RoboDK:
 - L101 Manulab simulation station is assumed complete (for this project at least)

TODO THIS WEEKEND / NEXT WEEK:

- Finalize thesis

Week 20 progress meeting

Monday:

TODO THIS WEEK:

- Finalize thesis
- Work on making the presentation video

Friday:

DONE THIS WEEK:

- Almost finalized thesis
- Finished presentation video
- Made final adjustments to sysmac program
- Made final adjustments to the project in movicon

TODO NEXT WEEK:

- Present the project
- Finalize thesis
 - Add Gantt
 - Format & add "work hours & descriptions"
 - Add progress reports
- Deliver thesis
 - Validate thesis with Skipnes before delivery
 - Deadline on May 20 12:00
- Present the project on May 20 09:15
 - Validate thesis before delivery
 - Deadline 20. May 12:00

Week 21 progress meeting

Monday:

TODO THIS WEEK:

- Present the project
- Finalize thesis
 - Add Gantt
 - Format & add "work hours & descriptions"
 - Add progress reports
- Deliver thesis
 - Validate thesis before delivery
 - Deadline 20. May 12:00

Appendix E

Manulab Preliminary Report

PRELIMINARY REPORT
FOR BACHELOR THESIS



TITLE: Manulab

CANDIDATE NUMBER(S): Fredrik Siem Taklo – 10060 Hans-Christian Ringstad – 10058 Hans Christian Haugan Finnson – 10067			
DATE: 22/01/2020	SUBJECT CODE: IE303612	SUBJECT: Bacheloroppgave	DOCUMENT ACCESS: - Open
FIELD OF STUDY: AUTOMATION		PAGES/ATTACHMENTS: 16 / 2	BIB. NR: - Not in use -

CLIENT(S) / ADVISOR(S): Clients: NTNU CPS Lab, Manulab Advisors: Ottar L. Osen, Paul Steffen Kleppe

<p>THESIS/SUMMARY:</p> <p>This is the preliminary report for the bachelor thesis <i>Manulab</i>, written at the Norwegian University of Science and Technology (NTNU) in Ålesund.</p> <p>The purpose of this thesis is to assist in the development of an Industry 4.0-compliant manufacturing laboratory, in which autonomous systems and robots are used to test and prototype production lines. The end result will be a proof of concept, as well as a working basis for further projects.</p> <p>The clients of this thesis are the <i>NTNU CPS Lab</i> and <i>Manulab</i>.</p>
--

This thesis is a work by students at NTNU in Ålesund.

Postadresse
Høgskolen i Ålesund
N-6025 Ålesund
Norway

Besøksadresse
Larsgårdsvegen 2
Internett
www.hials.no

Telefon
70 16 12 00
Epostadresse
postmottak@hials.no

Telefax
70 16 13 00

Bankkonto
7694 05 00636
Foretaksregisteret
NO 971 572 140

TABLE OF CONTENTS

TABLE OF CONTENTS	2
1 INTRODUCTION	3
2 TERMINOLOGY	3
3 PROJECT ORGANISATION	4
3.1 PROJECT GROUP	4
3.1.1 <i>Organization of project group</i>	4
3.1.2 <i>Assignments for all members</i>	4
3.1.3 <i>Project leader assignments</i>	4
3.1.4 <i>Project secretary assignments</i>	4
3.1.5 <i>Project engineer assignments</i>	5
3.2 CONTROL GROUP (ADVISORS AND CLIENT’S PERSONAL CONTACTS).....	5
4 AGREEMENTS	5
4.1 WORKSPACE AND RESOURCES	5
4.2 GROUP NORMS – RULES FOR COOPERATION – ATTITUDE	6
5 PROJECT DESCRIPTION	6
5.1 PROBLEM TO BE ADDRESSED - OBJECTIVE – PURPOSE	6
5.2 PROJECT SPECIFICATION	7
5.3 DEVELOPMENT METHODOLOGY	8
5.4 COLLECTING INFORMATION	8
5.5 RISK ANALYSIS	9
5.6 MAIN WORK ACTIVITIES	11
5.7 PROJECT SCHEDULE	12
5.7.1 <i>Main plan</i>	12
5.7.2 <i>Project control tools</i>	12
5.7.3 <i>Development tools</i>	12
5.7.4 <i>Internal control – evaluation</i>	13
5.8 DECISION MAKING PROCESS.....	13
6 DOCUMENTATION	13
6.1 REPORTS AND TECHNICAL DOCUMENTS	13
7 PLANNED MEETINGS AND REPORTS	13
7.1 MEETINGS.....	13
7.1.1 <i>Meeting with control group</i>	13
7.1.2 <i>Internal meetings</i>	13
7.1.3 <i>Progress reports</i>	14
8 DEVIATION MANAGEMENT	14
9 REQUIRED EQUIPMENT FOR EXECUTING PROJECT	14
10 REFERENCES	14
APPENDIXES	14

1 INTRODUCTION

Being able to accurately simulate and prototype real industrial processes is a boon when designing any production line. That is the goal of the Manulab project, in which we will assist in the construction of a laboratory for testing various methods of production. Conveyor systems, laser cutters, 3D printers and robots all play a part in the overall process.

Taking inspiration from Industry 4.0, the lab consists of autonomous AIVs moving between various production lines and storage facilities. Everything is connected to a LAN, and a master computer processes orders and assigns tasks to maximise efficiency.

After our first meeting with the control group, we learned that several robot cells will not be available for most of the semester due to renovation. Due to this, we've decided to primarily focus on producing parts using laser cutters and 3D printers, as well as the logistics between them using AIVs.

The Manulab project was chosen after a meeting with the student advisor. The majority of the equipment which will be used for the project is manufactured by Omron, whose products the group has some prior experience with. Some members of the group also have work experience with production lines from earlier. With these factors in mind we believe this thesis will be a good fit for the team.

2 TERMINOLOGY

Manulab	Manufacturing Laboratory
LAN	Local Area Network
Istagentt	Tool for creating project timelines and gantt charts
LD Robot	«AIV Warehouse robot» that moves along the floor and can autonomously navigate the workspace.
AIV	Autonomous Intelligent Robot
TM Robot	Techman Robot, a 6DOF cobot arm delivered by Omron
Cobot	Collaborative robot, sensitive enough to work in tandem with humans

3 PROJECT ORGANISATION

3.1 *Project group*

Student number(s)
484137 – Fredrik Siem Taklo 484217 – Hans-Christian Ringstad 488565 – Hans Christian Haugan Finnson

3.1.1 Organization of project group

Name	Assigned role
Fredrik Siem Taklo	Project leader
Hans-Christian Ringstad	Project secretary
Hans Christian Haugan Finnson	Project engineer

The productivity tools Instagantt and Asana will be used to plan and monitor the workload as time goes by. If these turn out to be unsatisfactory, a shared excel sheet will be used instead.

Each task is assigned to a member who is responsible of seeing it to completion. An assistant is also assigned for every task in case support is needed.

3.1.2 Assignments for all members

- Complete assigned tasks
- Comply with agreed upon work hours and norms

3.1.3 Project leader assignments

- Main contact person
- Responsible for keeping the group on schedule

3.1.4 Project secretary assignments

- Secondary contact person
- Responsible for arranging and recording meetings

3.1.5 Project engineer assignments

- Primarily focused on the work itself
- Responsible for planning work tasks for the group

3.2 Control group (*advisors and client's personal contacts*)

- Ottar Laurits Osen and Paul Steffen Kleppe
- Oversees the progress of the thesis

4 AGREEMENTS

4.1 Workspace and resources

- Access to workspace:
 - Access to room L101 or the room in use for all members has been requested.
 - Workspace inaccessible, moved to room L108.
 - Access to workshops for performing mechanical work.
- Access to Resources:
 - Essential components and robots required for the project are missing. Estimated delivery times stretching late into January, possibly February.
 - Some robot cells will probably *not* be available throughout the semester due to renovation of room L101. We will have to work with what we have in room L108.
- Access to key persons:
 - Advisors will be available for contact.
- Agreement regarding reporting:
 - Everyone writes timetables and reports when their work is done for the day. Manuals and data sheets are to be collected.

4.2 ***Group norms – rules for cooperation – attitude***

The group will strive to keep conflicts to a minimum by maintaining a positive attitude. In the event of a project-related disagreement, a vote will be held among the members. If at a standstill, the group will ask the advisors for the best possible solution.

Core working hours are between 9-15 Mon-Fri
Ideal working hours are between 8-16 Mon-Fri

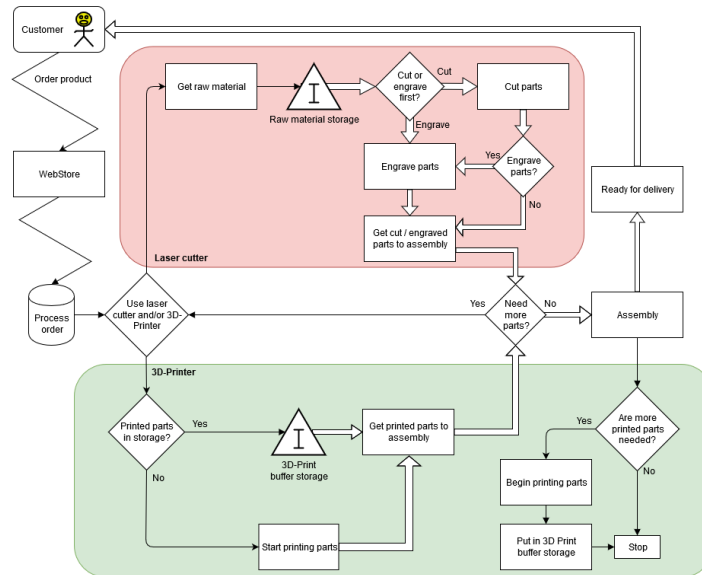
- Work hours are ideally logged and submitted every workday.
- Lost hours should be worked in later.
- If a member cannot show up, the group must be notified.
- On days of Industry 4.0 classes, work is not mandatory. The same applies for holidays.
- Progress meetings are held every Monday morning. A brief summary meeting is also held at the end of each week.
- Every member keeps focus on their own task, providing assistance for other members when necessary. This is done to maximise efficiency.
- If a task comes to a halt, the entire group is notified.

5 PROJECT DESCRIPTION

5.1 ***Problem to be addressed - objective – purpose***

Manulab is a national project for state-of-the-art manufacturing research, with several laboratories spread across the nation. The purpose of this thesis is to assist in the development of such a laboratory in Ålesund.

Ultimately, the goal is to design and develop a working production line that is entirely autonomous. It will serve as a proof of concept and be a basis for future projects – including bachelor theses – to expand upon. The general idea is that a customer can order a specific product from a webstore, and a master computer will then process said order. Product components will then be constructed and assembled (if possible), and stored upon completion, ready to be delivered to the customer as shown in the flowchart below.



Flowchart: Rough idea of how the Manulab will work

5.2 Project specification

The project must...	The project should...	The project could...
Work as a production line, with the possibility of producing a product	Work as intended, demonstrating an example of a production process.	Have a digital / simulated counterpart
Utilise the different machines to demonstrate their capabilities and usefulness	Be flexible enough to allow for changes to the production line	A GUI that shows progress on the current task
Be secure for humans in the open workspace	Allow for creating different products simultaneously	Improve upon clients' specifications
Communicate between the different stations via LAN	Include descriptions for how the production line could be improved, and act as a good basis for that	All levels of the production line, including maintenance, should be entirely autonomous
	Have the production itself be autonomous	

5.3 *Development methodology*

The productivity tools Instagantt and Asana will be used to plan and keep track of individual tasks and progress. We will be working closely with clients and suppliers to ensure that the final outcome meets their expectations for the project.

Taking inspiration from Industry 4.0, we will attempt to implement Lean Product Development (LPD) into our development process. LPD is an approach to product development that emphasises cutting down on costs and waste, developed by Toyota. It encompasses various concepts like:

- Build up re-usable knowledge, to mitigate repetition and cut down on time spent.
- Concurrent engineering, meaning parallelization of tasks.
- Clear individual responsibilities.
- «Cadence and pull». Engineers plan their own work and work autonomously.
- Emphasis on visualization. (E.g. Kanban & gantt charts)
- Entrepreneurial system design, making one person responsible for the project as a whole.

5.4 *Collecting information*

Information collected thus far:

- Manuals, instructional videos and Powerpoint presentations for Omron hardware, in addition to some practical experience for using some of the robots, courtesy of H.C. Finnson
- Limited idea of hardware and robots we will have access to once all the required parts of the Manulab have been delivered and set up.
- L101 will *not* be available for the majority of this thesis. Parts of L108 will be available instead.
- Omron provides a SharePoint and will lend equipment if need be.

Information to be collected:

- Product specifications and expectations from client
- Further details from supplier (Amatec) and advisors about when to expect final deliveries, layout and usage of equipment.
- How to use various software.
- How communication between equipment is to be set up.

5.5 Risk analysis

Frequency of incident	Severity of consequences				
	Very Low Severity	Low Severity	Medium Severity	High Severity	Very High Severity
Very High Frequency					9.
High Frequency					1.
Medium Frequency			3. 5.	2. 10	
Low Frequency				4. 11. 12.	
Very Low Frequency		6.	14.		

	Physical risks	Description	Risk mitigation
1.	Crush hazard	Body parts might be crushed by belts or robots.	Don't allow operation without ensuring no personnel can be harmed with a security fence and/or light curtain.
2.	Electrical fire	Electrical fire might occur in the workspace.	The group will make sure to pay attention to fire exits and countermeasures before work starts.
3.	Injuries when working with power tools	Power tools can cause injury when used irresponsibly	The group will take necessary precautions and use safety gear.

4.	Electrical shock	Equipment running on 230V AC and can cause 1 st to 3 rd degree burns and ventricular fibrillation.	All conductive parts will be isolated. When working with wiring, electricity will be disconnected. In the case of electrical shock, the person in question will be sent to the closest medical centre for inspection.
5.	Short circuit	Short circuits may cause damage to equipment.	Electrical circuits will be equipped with fuses to minimize potential damage.
6.	Sensor loss or control loss	Robots and conveyors may lose sensors or control while active.	Both hardware and software should have failsafes implemented.

	Risk to project	Description	Risk mitigation
7.	Redesign of work area	Originally planned layout of the lab may have to be revised.	It is important to have alternative, viable layout plans ready.
8.	Failed plans	There is a possibility that the plans made only work in theory, but not in practice.	The group will work with advisors to find an alternative solution.
9.	Lack of resources / testing equipment	If the necessary equipment isn't delivered in time, there may be too little time left to test and prepare the project before the deadline.	Make sure everything is ready before testing.
10.	Unpredicted issues	There is a chance that some unpredicted issues might be time consuming and delaying other tasks.	If some issues come up, the group will work together with the advisor(s) to solve the issues.
11.	Overwork	If too much work / stress is placed upon one person, it may cause burnout.	Work hours are agreed upon at the beginning of the project. Members should take pauses.
12.	Sickness	Sickness may put a member out of commission, halting progress.	Maintaining proper hygiene is <i>mandatory</i> .
13.	Long term injury	Carelessness during mechanical work and bad posture may lead to long term injuries.	Necessary precautions will be followed, like following the lab manual and EHS rules.
14.	Unauthorized access	Unauthorized access to the workspace by others may have undesired consequences.	The project is being built in a room where a authorized access card is needed to enter, and security fences will also further hinder unauthorized access to the workspace.

NTNU I ÅLESUND
PRELIMINARY REPORT – BACHELOR THESIS

SIDE 11

5.6 *Main work activities*

FST – Fredrik Siem Taklo
HCR – Hans-Christian Ringstad
HCF – Hans Christian Finnson

Nr	Main activity	Responsibility , Assistant	Time / Scope
A1	Documenting		5 months
A11	Bachelor Thesis	All	5 months
A2	Preliminary report		1 month
A21	Assign roles	All	1 day
A22	Preliminary report draft	All	1 week
A23	Preliminary report completed	All	1 month
A3	Preparation & research		
A31	Research Omron software	All	1 month
A32	Research laser cutter	HCR , HCF	1 month
A33	Research 3D printers	FST , HCR	1 month
A34	Research LD robots	HCF , FST	1 month
A35	Create and test product prototypes	HCF , FST	3 weeks
A4	Building lab / Temporary space		2 months
A41	Get equipment from suppliers	FST , HCR	2 months
A42	Set up LD AIV fleet	HCF , FST	2 weeks
A43	Set up TM robots	HCF , FST	2 weeks
A44	Set up laser cutter	HCR , HCF	2 weeks
A45	Set up 3D printer rack	FST , HCR	2 weeks
A5	Programming		2 month
A51	Create LD-robot programs	HCF , FST	2 months
A52	Create laser cutter programs	HCR , HCF	2 months
A53	Create 3D printer programs	FST , HCR	2 months
A54	Create 3D model of products	HCF , FST	2 days
A55	Create Webstore	HCR , HCF	2 days
A56	Integrate master computer	All	1 month
A6	Testing and fine-tuning		1 months
A61	Testing and bugfixing	All	1 months

5.7 Project schedule

5.7.1 Main plan

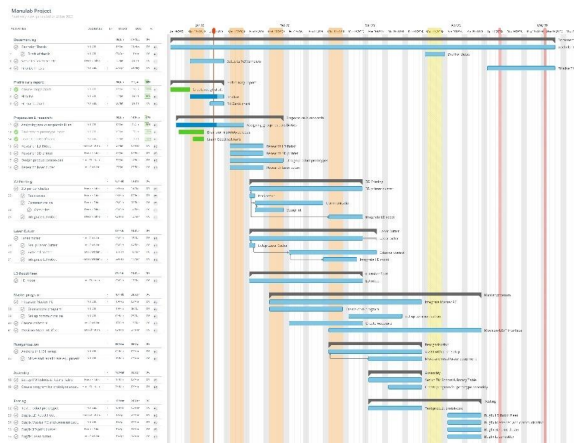


Figure: tentative gantt chart of planned tasks. See appendix A for a higher resolution image.

5.7.2 Project control tools

To keep track of workload and progress we use the productivity tool Asana, which combined with the online service Instagantt is used to create gantt charts.

A gantt chart is a chronologically ordered visualisation of tasks that need to be done. Using Instagantt we can assign tasks to individual group members, as well as order tasks by priority.

5.7.3 Development tools

This project encompasses a wide variety of equipment, tools and robots, all of which require the right tools and software to operate.

- Sysmac Studio: for programming Omron devices
- MobilePlanner: for programming LD floor robots
- Fusion 360: for 3D modelling parts
- Cura: for preparing 3D models for printing
- Github: for maintaining code repositories

5.7.4 Internal control – evaluation

Internal controls will be an important part of the project throughout the whole thesis. As previously mentioned, the group will have meetings every Monday morning and Friday to discuss progress and plan ahead.

In particular, these meetings will be spent on figuring how each member's task affects the others', and what to prioritise in the future to maximise efficiency.

There will also be biweekly meetings with advisors on Tuesdays to discuss the progress on our project and any problems we may have run into.

5.8 *Decision making process*

Every major decision is to be decided upon by everyone during meetings. This includes decisions like which tasks get delegated to whom, or whether to cancel a task altogether.

6 DOCUMENTATION

6.1 *Reports and technical documents*

Every member is responsible for documenting their own work. Gantt charts and taking pictures of ongoing work is encouraged.

7 PLANNED MEETINGS AND REPORTS

7.1 *Meetings*

7.1.1 Meeting with control group

There will be a biweekly meeting with the control group, held on Tuesdays of every odd-numbered week.

7.1.2 Internal meetings

One meeting on Mondays to discuss anything we've done over the weekend, and what we're planning to do during the week.

A small meeting at the end of Friday to discuss what we did during the week, and how well it met the original plans from Monday.

7.1.3 Progress reports

Every day members will write down what they have done including working hours. A small progress report will be made

8 DEVIATION MANAGEMENT

If deviation occurs, the group will meet to decide on how to proceed, whether by reworking the part of the project causing problems, replacing it with another function, or cutting it from the project all together. If the deviation is caused by- or would cause major changes, then we will discuss with the project advisor on how to proceed.

9 REQUIRED EQUIPMENT FOR EXECUTING PROJECT

To build the final lab (room L101), we'll need

- Omron TM collaborative robot arms
- Master PC with monitor, keyboard & mouse
- Omron LD mobile robots with MobilePlanner software
- Omron LD mobile robots, Mobile Planner software and license key
- Omron LD mobile robots with Conveyor belt
- Omron LD mobile robots with arm
- Safety fences
- 3D-printers & racks

To build temporarily testing area (room L108) we need

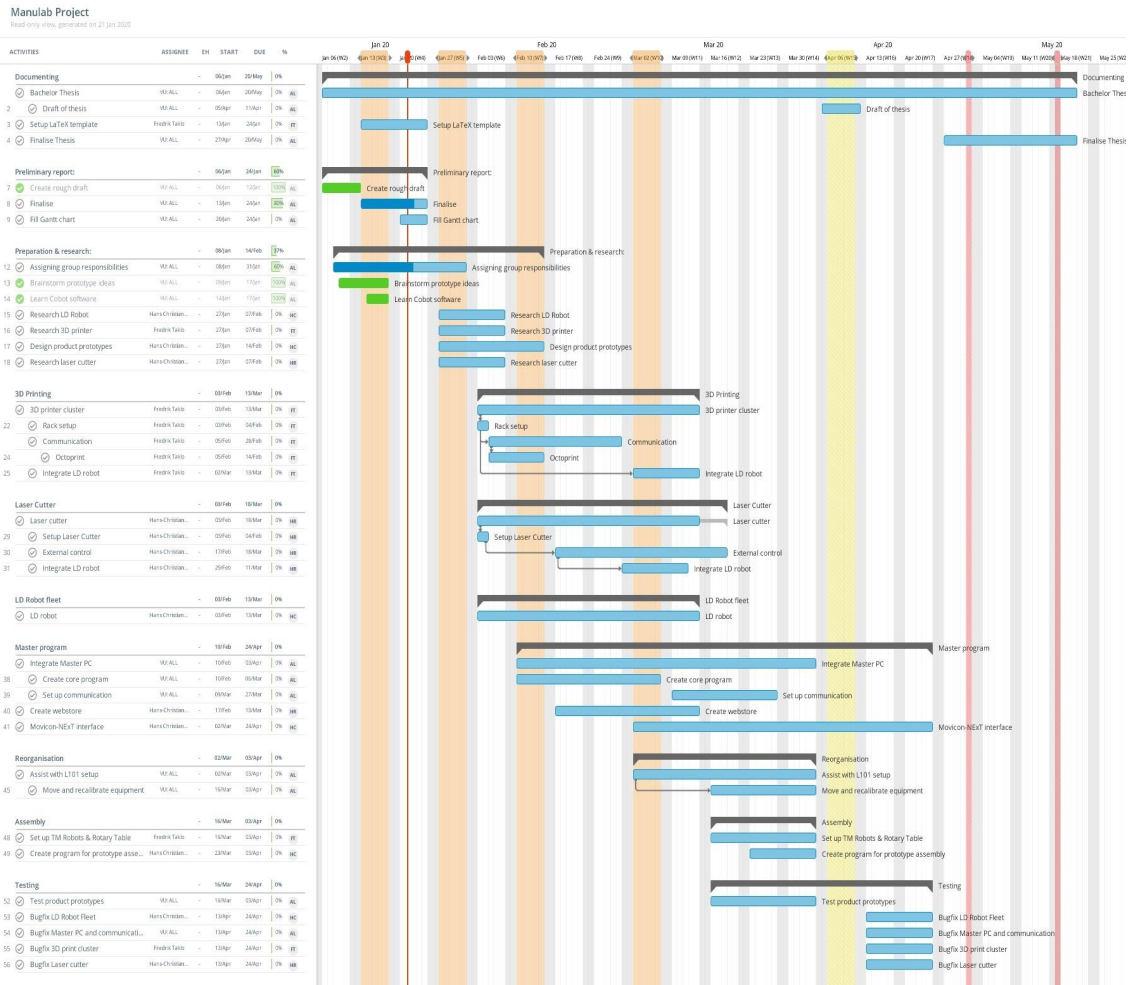
- Two 3D-printers in rack
- One laser cutter
- Master PC with monitor, keyboard & mouse
- Omron LD robot with arm, different grippers

10 REFERENCES

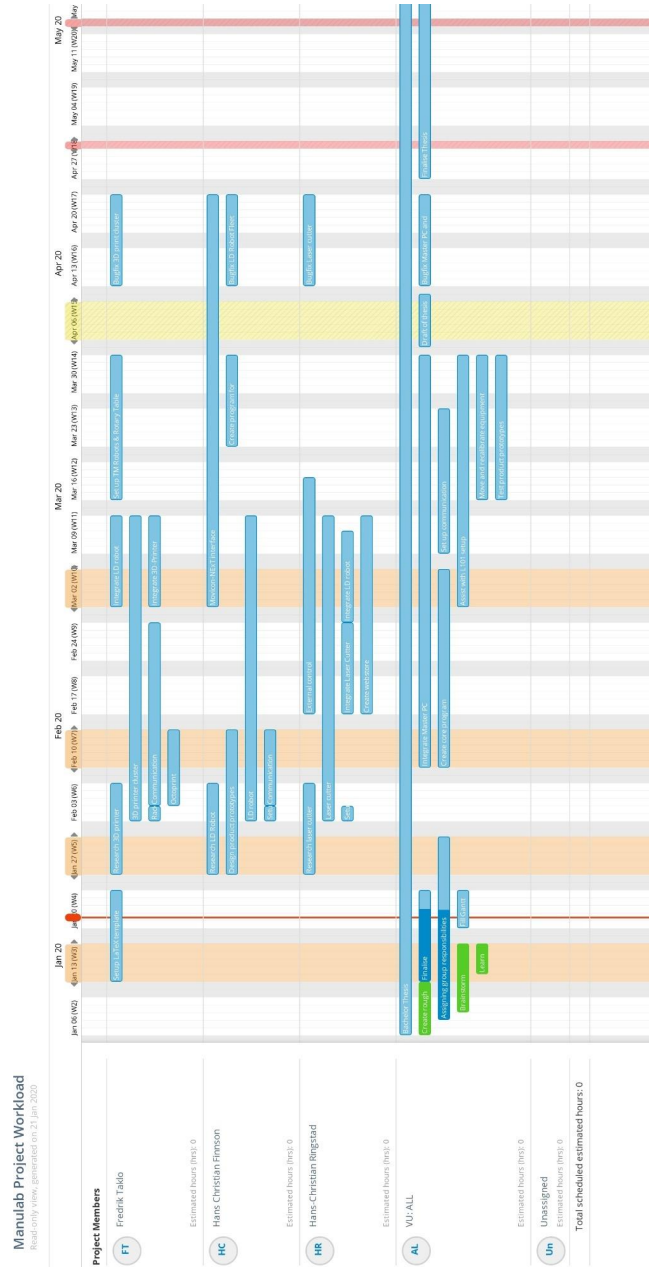
APPENDIXES

Appendix A	Gantt diagram
Appendix B	Workload diagram

10.1 Appendix A: Gantt diagram



10.2 Appendix B: Workload Diagram



Appendix F

Sysmac Master Program Source Code

Master Program_5

Master Program_5

Author: HC Finnson, HC Ringstad, Fredrik Taklo
Created: 20/02/2020 15:08:02
Last Modified: 12/05/2020 18:44:54
Comment:

Sysmac Studio Module Version: 1.29.1.0

Master Program_5

Table of Contents

Master Program_5	3
1.IPC	3
1-3.Controller Setup	3
1-3-2.Built-in EtherNet/IP Port Settings	3
1-5.Task Settings	5
1-6.POU's	6
1-6-1.Programs	6
1-6-1-1.Master	6
1-6-1-1-1.Variables	6
1-6-1-1-2.Program Body	8
1-6-1-2.Background	16
1-6-1-2-1.Variables	16
1-6-1-2-2.Program Body	17
1-6-1-3.commsCSV	22
1-6-1-3-1.Variables	22
1-6-1-3-2.writeCutterDxf	23
1-6-1-3-3.writePrinterCommand	24
1-6-1-3-4.readCutterStatus	25
1-6-1-3-5.readPrinterStatus	26
1-6-1-4.commsLD	29
1-6-1-4-1.Variables	29
1-6-1-4-2.ConnectToLDs	31
1-6-1-4-3.getFromPrinter	33
1-6-1-4-4.getFromKanban	37
1-6-1-4-5.getFromCutter	39
1-6-1-4-6.deliverProduct	41
1-6-1-5.commsTM	43
1-6-1-5-1.Variables	43
1-6-1-5-2.ModbusTCP_Setup	45
1-6-1-5-3.doTask	52
1-6-1-6.UDP_Socket	53
1-6-1-6-1.Variables	53
1-6-1-6-2.Program Body	54
1-6-2.Functions	56
1-6-2-1.FindAvailablePrinter	56
1-6-2-1-1.Variables	56
1-6-2-1-2.Program Body	57
1-6-2-2.ProcessGcodeName	58
1-6-2-2-1.Variables	58
1-6-2-2-2.Program Body	59
1-7.Data	61
1-7-1.Data Types	61
1-7-2.Global Variables	65

1.IPC

1-3.Controller Setup

1-3-2.Built-in EtherNet/IP Port Settings

Built-in EtherNet/IP Port Settings

TCP/IP Settings	
EtherNet/IP Port - IP Address Settings	Fixed setting
EtherNet/IP Port - IP address	192.168.250.1
EtherNet/IP Port - Subnet mask	255.255.255.0
Internal Port1 - IP address	192.168.254.1
Internal Port1 - Subnet mask	255.255.255.0
Default gateway	192.168.250.240
DNS	Do not use
Priority DNS server	
Secondary DNS server	
Domain name	
Host Name - IP Address	No settings
Keep Alive	Use
Keep Alive monitoring time	300 sec
Linger option	Do not specify
IP Router Table	No settings
IP Forward	Use
NAT	Use
Packet filter	Use
Pass Frame	
No.1 Specification Method	Interface network
No.1 IP Address	
No.1 Mask	
No.1 Protocol	any
No.1 Range specification	False
No.1 Port A	
No.1 Port B	
LINK Settings	
EtherNet/IP Port - LINK settings	Auto
FTP Settings	
FTP server	Do not use
Port No.	21
Login name	
Password	
SNMP Settings	
SNMP service	Do not use
Port No.	161
Address	
Location	
Send a recognition trap.	FALSE
Recognition method	IP address
IP address	0.0.0.0
Host name	
Community name	public
Recognition 2	Do not use
Recognition method	IP address

Master Program_5

IPC Built-in EtherNet/IP Port Settings

IP address	0.0.0.0
Host name	
Community name	public
SNMP Trap Settings	
SNMP trap	Do not use
Port No.	162
Specifying method	IP address
IP address	
Host name	
Community name	public
Version	SNMPv2C
Trap 2	Do not use
Specifying method	IP address
IP address	
Host name	
Community name	public
Version	SNMPv2C
CIP Settings	
CIP routing	Use

Master Program_5

IPC Task Settings

1-5.Task Settings

Task Name	Period/Execution Conditions	Detailed Execution Conditions	Task Period Exceeded Detection	Task Timeout Detection Time	Execution Priority	Variable Access Time [%]
Primary periodic task						
PrimaryTask	2ms		Detect	10ms	4	3
Periodic task						
Periodic	100ms		Detect	500ms	16	3

Unit	Task Name
EtherCAT Network Configuration	

Task Name	Assigned programs	Initial status
PrimaryTask		
1	Master	Run
2	commsCSV	Run
3	commsLD	Run
4	commsTM	Run
Periodic		
1	Background	Run

Variable to be refreshed	Accessing Task
PrimaryTask	
Periodic	

APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC Variables

Master

1-6.POUs

1-6-1.Programs

1-6-1-1.Master

1-6-1-1-1.Variables

Name	Data Type	Initial Value	AT	Retain	Constant	Comment
VAR						
orderChosenPrinterNr	INT			False	False	PrinterNr of the first available printer that was found
orderCurrent	ST_OrderDetails			False	False	Struct containing information on the order that is currently being processed
customerQueue	ARRAY[1..100] OF ST_OrderDetails			False	False	Array of orders that have been received while another order was being processed
customerQueueLength	INT			False	False	Length of the customer queue, due to Sysmac being unable to have flexible length arrays.
startedFromQueue	BOOL			False	False	Boolean used to check if the currently processing order was started from the queue or not, to determine if queue updating is necessary
index	INT			False	False	Basic index used for various iteration tasks
connectedPrinters	INT			False	False	Number of connected printers
printerPartsInStore	BOOL			False	False	If the part needed for the order is in stock, then this is TRUE
orderCutterDone	BOOL			False	False	Bool used for storing that the printed part has been collected
orderPrinterDone	BOOL			False	False	Bool used for storing that the lasercut part has been collected
refreshTimer	TON			False	False	General purpose timer, used for waiting before checking connection status rather than spam continuously
VAR_EXTERNAL						
masterState	INT				False	
customerOrder	ST_OrderDetails				False	
csvBools	ST_CsvTools				False	
cutterDxfInfo	ST_CutterDxfExport				False	
printerStatus	ARRAY[0..19] OF ST_3DPrinterStatus				False	
printerCommand	ST_3DPrinterCommand				False	
moviconSendStatus	ST_MoviconStatus				False	
LDCommands	ST_LDTools				False	
moviconSendCommand	ST_MoviconCommands				False	
TMCommands	ST_TMTTools				False	
printerActivePrints	ARRAY[0..19] OF ST_3DPrinterCurrentPrintInfo				False	
cutterStatus	ST_CutterStatus				False	
firstBackgroundRunDone	BOOL				False	
cutterProductInfo	ARRAY[1..2] OF ST_CutterDxfExport				True	
printerProductInfo	ARRAY[1..2] OF STRING[256]				True	

APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC Variables

Master

backgroundWritingCommand	BOOL				False	
productStatus	ARRAY[1..2] OF SI_ProductStatus				False	

1-6-1-1-2.Program Body

```

1 //0000-----
2 // Startup functions
3 IF masterState = 0 THEN
4 // Run startup code. Set initial value for variables and initialize function blocks
5 customerQueueLength := 0;
6 connectedPrinters := 0;
7 refreshTimer.In := FALSE;
8
9 IF firstBackgroundRunDone THEN
10     masterState := 100;
11 END_IF;
12
13 //masterState := 10000;
14 END_IF;
15 IF masterState = 100 THEN
16 // Check cutter for connection/errors. If no errors, proceed.
17 IF cutterStatus.done='True' THEN
18     masterState := 200;
19 ELSIF cutterStatus.error='True' THEN
20     masterState := 9010;
21 END_IF;
22 END_IF;
23 IF masterState = 200 THEN
24 // Check how many printers are connected. If at least 1, proceed.
25 FOR index := 0 TO SizeOfAry(printerStatus)-1 DO
26     IF printerStatus[index].connected='True' THEN
27         connectedPrinters := connectedPrinters + 1;
28     END_IF;
29 END_FOR;
30 IF connectedPrinters > 0 THEN
31 //1 or more printers available, proceed.
32     masterState := 300;
33 ELSIF connectedPrinters = 0 THEN
34 //No printer available, error
35     masterState := 9020;
36 END_IF;
37 END_IF;
38 IF masterState = 300 THEN
39 //Connect to LD robots. If successful, proceed, if not, throw error.
40 LDCommands.connect := TRUE;
41 IF LDCommands.connectSuccessful THEN
42     masterState := 400;
43 ELSIF LDCommands.connectError THEN
44     masterState := 9030;
45 END_IF;
46 END_IF;
47 IF masterState = 400 THEN
48 //Connect to TM robots. If successful, proceed, if not, throw error.
49 TMCommands.connect := TRUE;
50 IF TMCommands.connectSuccessful THEN
51     masterState := 500;
52 ELSIF TMCommands.connectError THEN
53     masterState := 9040;
54 END_IF;
55 END_IF;
56

```

```

57 IF masterState = 500 THEN
58     // Check viper, quattro, etc, then go to idle
59
60     masterState := 1000;
61 END_IF;
62
63 //1000-----
-----
64 // Idle, awaiting order
65 IF masterState = 1000 THEN
66
67     IF customerOrder.orderReceived THEN
68         //Order is received from Movicon, copy information to local variable for processing and reset
customerOrder to prevent a loop
69         orderCurrent := customerOrder;
70         customerOrder.orderReceived := FALSE;
71         customerOrder.productID := 0;
72         customerOrder.name := "";
73         masterState := 2000;
74     ELSIF NOT(customerOrder.orderReceived) AND customerQueue[1].orderReceived THEN
75         // There is no new order from Movicon, but the first position in the queue is filled with an order.
76         // Copy the order from the first spot in the queue into the local variable for processing. Signify
that this was an order started from the queue.
77         orderCurrent := customerQueue[1];
78         startedFromQueue := TRUE;
79         masterState := 2000;
80     END_IF;
81
82 END_IF;
83
84 // Add order to the back of the queue when a new order is received while the program is busy processing
another.
85 // Reset customerOrder after completing to open up for a new order if one comes
86 // Note, this always runs in the background.
87 IF NOT(masterState = 1000) AND customerOrder.orderReceived THEN
88     customerQueueLength := customerQueueLength + 1;
89     customerQueue[customerQueueLength] := customerOrder;
90     customerOrder.orderReceived := FALSE;
91     customerOrder.productID := 0;
92     customerOrder.name := "";
93 END_IF;
94
95
96 //2000-----
-----
97 //Process customer order
98 // 2000 = setup
99 // 2100 = process dxf for cutter
100 // 2200 = check kanban storage for printer
101
102 IF masterState = 2000 THEN
103     // Setup
104     masterState := 2100;
105 END_IF;
106 IF masterState = 2100 THEN
107     // Cutter
108     // Depending on the product ID, either apply the preset for the nameplate or the keychain for export to
Dxf, and add the customer's name to it.
109     IF orderCurrent.productID > 0 AND orderCurrent.productID <= SizeOfAry(cutterProductInfo) THEN
110         cutterDxfInfo := cutterProductInfo[orderCurrent.productID];

```

Master Program_5

IPC Program Body

Master

```

111         cutterDxfInfo.textToInsert := orderCurrent.name;
112         masterState := 2200;
113
114     ELSE
115         masterState := 9210;
116     END_IF;
117 END_IF;
118 IF masterState = 2200 THEN
119     //Printer
120     // Check the kanban storage to see if the required part is in storage.
121     IF productStatus[orderCurrent.productID].number > 0 THEN
122         //Product ID exists and there is at least 1 part in storage
123         printerPartsInStore := TRUE;
124         masterState := 3000;
125     ELSIF productStatus[orderCurrent.productID].number = 0 THEN
126         //Product ID exists but there are no parts in storage
127         printerPartsInStore := FALSE;
128         masterState := 3000;
129     ELSIF productStatus[orderCurrent.productID].number < 0 THEN
130         //Fault in number tracking, reset to 0 and continue as if there are no parts in store.
131         printerPartsInStore := FALSE;
132         masterState := 3000;
133     ELSE
134         //Non-existent product ID
135         masterState := 9220;
136     END_IF;
137 END_IF;
138
139
140 //3000-----
141 // Assign work to cutter, printer, robots, etc
142 // 3000 = setup
143 // 3100 = assign cutter task
144 // 3200 = assign printer task
145 // 3300 = assign LD robots
146 // 3400 = assign TM robots
147 IF masterState = 3000 THEN
148     //Setup
149     orderChosenPrinterNr := -1;
150     masterState := 3100;
151 END_IF;
152 IF masterState = 3100 THEN
153     //Cutter
154     //With the information added to the dxfInfo variable and ready for export, start the csv conversion
    program
155     //After that is done, tell Movicon to start the Python script
156     IF NOT(csvBools.writeCutterDxfStart) AND NOT(csvBools.writeCutterDxfDone) THEN
157         csvBools.writeCutterDxfStart := TRUE;
158     END_IF;
159     IF csvBools.writeCutterDxfStart AND csvBools.writeCutterDxfDone THEN
160         csvBools.writeCutterDxfStart := FALSE;
161         moviconSendCommand.startDxfScript := TRUE; //Note, needs to be turned off at some point.
    Could do in cleanup
162         masterState := 3200;
163     END_IF;
164 END_IF;
165 IF masterState = 3200 THEN
166     //Printer
167     //Regardless of whether needed part is or is not in store, print a new one.

```

```

168 //Iterate through the array of printer statuses to find the first printer that is ready and available to use.
169 orderChosenPrinterNr := FindAvailablePrinter(statusArray:=printerStatus);
170 IF orderChosenPrinterNr = -1 THEN
171     //If no available printer was found, go to error.
172     masterState := 9320;
173 ELSE
174     //Otherwise, continue as planned
175     masterState := 3210;
176 END_IF;
177 END_IF;
178 IF masterState = 3210 AND NOT(backgroundWritingCommand) THEN
179     //Now that an available printer has been found, create a command to send to the printer.
180     printerCommand.separator := 'sep='; //Ensures that the file can be opened in excel if required
181     printerCommand.IP := CONCAT(In1:='$R$L', In2:=printerStatus[orderChosenPrinterNr].IP); //R$L is
newline
182     printerCommand.command := 'print';
183     printerCommand.number := '1';
184     printerCommand.argument := printerProductInfo[orderCurrent.productID];
185     printerActivePrints[orderChosenPrinterNr].productID := orderCurrent.productID;
186     printerActivePrints[orderChosenPrinterNr].number := STRING_TO_UINT(printerCommand.number);
187     productStatus[orderCurrent.productID].printing := productStatus[orderCurrent.productID].printing + 1;
188     masterState := 3220;
189 END_IF;
190 IF masterState = 3220 AND NOT(backgroundWritingCommand) THEN
191     //Run the csv conversion program to put the previously created printer command into a csv, then tell
Movicon to run the python script
192     IF NOT(csvBools.writePrinterCommandStart) AND NOT(csvBools.writePrinterCommandDone) THEN
193         csvBools.writePrinterCommandStart := TRUE;
194     END_IF;
195     IF csvBools.writePrinterCommandStart AND csvBools.writePrinterCommandDone THEN
196         csvBools.writePrinterCommandStart := FALSE;
197         moviconSendCommand.startPrinterScript := TRUE; //Note, needs to be turned off at some point,
could do in cleanup
198         masterState := 3300;
199     END_IF;
200 END_IF;
201 IF masterState = 3300 THEN
202     // LD robots
203     IF printerPartsInStore THEN
204         //Command LD robot to pick up parts
205         //May use CASE if rack is too large to only pick from one spot
206         //Could push to 4300?
207         LDCommands.productID := orderCurrent.productID;
208         LDCommands.getFromKanbanStart := TRUE;
209         masterState := 4000;
210     ELSIF NOT(printerPartsInStore) THEN
211         //Part needed is not in store
212         masterState := 9330; //Temporary error state, wait until printing is done and kanban has the
given part.
213     END_IF;
214 END_IF;
215
216 //4000-----
217 // Wait for processes to complete, then continue production
218 // Pick up the different parts and move to assembly table
219 // 4000 initial setup, wait until part is retrieved from kanban
220 // 4100 wait for cutter to be done, then flag it as done
221 // 4200 no printer jobs
222 // 4300 Retrieve item from cutter

```

Master Program_5

IPC Program Body

Master

```

223 IF masterState = 4000 THEN
224     IF LDCommands.getFromKanbanDone THEN
225         orderPrinterDone := TRUE;
226         LDCommands.getFromKanbanStart := FALSE;
227         productStatus[orderCurrent.productID].number := productStatus
[orderCurrent.productID].number - 1;
228
229         masterState := 4100;
230     END_IF;
231 END_IF;
232 IF masterState = 4100 THEN
233     //Cutter
234     //Cutter status csv is being continuously read/refreshed in Background. Wait until it reads that the cutting
is done
235     IF cutterStatus.done='True' THEN
236         //Cutter is done. Move to next task
237         orderCutterDone := TRUE;
238         masterState := 4200;
239     ELSIF cutterStatus.error='True' THEN
240         //Error happened, go to appropriate state.
241         masterState := 9410;
242     END_IF;
243 END_IF;
244 IF masterState = 4200 THEN
245     //No more necessary printer tasks to do
246     masterState := 4300;
247 END_IF;
248 IF masterState = 4300 THEN
249     //LD robots
250     masterState := 4310;
251 END_IF;
252 IF masterState = 4310 THEN
253     //Retrieve item from cutter
254     IF orderCutterDone AND NOT(LDCommands.getFromKanbanStart) AND NOT
(LDCommands.getFromCutterDone) THEN
255         LDCommands.productID := orderCurrent.productID;
256         LDCommands.getFromCutterStart := TRUE;
257     END_IF;
258     IF orderCutterDone AND LDCommands.getFromCutterStart AND LDCommands.getFromCutterDone
THEN
259         LDCommands.getFromCutterStart := FALSE;
260         masterState := 5000;
261     END_IF;
262 END_IF;
263
264 //5000-----
265 // Order assembly and delivery
266 IF masterState = 5000 THEN
267     TMCommands.productID := orderCurrent.productID;
268
269     TMCommands.arm1_jobNr := orderCurrent.productID;
270     TMCommands.arm1_jobStart := TRUE;
271
272     TMCommands.arm2_jobNr := orderCurrent.productID;
273     TMCommands.arm2_jobStart := TRUE;
274
275     masterState := 5100;
276 END_IF;
277 IF masterState = 5100 THEN

```

12 / 68


```

278     IF TMCommands.arm1_jobDone AND TMCommands.arm2_jobDone THEN
279         TMCommands.arm1_jobStart := FALSE;
280         TMCommands.arm2_jobStart := FALSE;
281         TMCommands.productID := 0;
282         LDCommands.productID := orderCurrent.productID;
283         LDCommands.deliverProductStart := TRUE;
284         masterState := 5200;
285     END_IF;
286 END_IF;
287
288 IF masterState = 5200 THEN
289     IF LDCommands.deliverProductDone THEN
290         LDCommands.deliverProductStart := FALSE;
291         masterState := 6000;
292     END_IF;
293 END_IF;
294
295
296 //6000-----
297 // Update status and clean up
298
299 IF masterState = 6000 THEN
300     //Reset local variables
301     printerPartsInStore := FALSE;
302     orderCutterDone := FALSE;
303     orderPrinterDone := FALSE;
304     moviconSendCommand.startDxfScript := FALSE;
305     moviconSendCommand.startPrinterScript := FALSE;
306     orderCurrent.orderReceived := FALSE;
307     orderCurrent.productID := 0;
308     orderCurrent.name := "";
309
310     IF startedFromQueue THEN
311         masterState := 6050;
312     ELSIF NOT(startedFromQueue) THEN
313         masterState := 6100;
314     END_IF;
315 END_IF;
316 IF masterState = 6050 THEN
317     //If the current process was started from a queue, then move all spots in the queue down one level and
318     //reset the last.
319     FOR index := 1 TO customerQueueLength-1 DO
320         customerQueue[index] := customerQueue[index+1];
321     END_FOR;
322     customerQueue[customerQueueLength].orderReceived := FALSE;
323     customerQueue[customerQueueLength].productID := 0;
324     customerQueue[customerQueueLength].name := "";
325     customerQueueLength := customerQueueLength - 1;
326     startedFromQueue := FALSE;
327
328     //Queue process complete, continue with order cleanup
329     masterState := 6100;
330 END_IF;
331 IF masterState = 6100 THEN
332     //Cleanup complete, return to idle
333     masterState := 1000;
334 END_IF;
335
336 //9000-----

```

```
337 // Error handling
338 // Actual errors, handling "no printers/cutters available"/"can't complete task", etc
339 // 9X00, replace X with whatever X000 master state the program was in when error occurred.
340 // 90Y0, replace Y with whatever 0Y00 master state the program was in when error occurred.
341 // 900Z, replace Z with whatever 00Z0 master state the program was in when error occurred.
342
343 IF masterState = 9010 THEN
344     //Cutter is unable to connect
345     //Send to Movicon?
346     //Wait a while, then set master state to 100 to try again.
347     //Maybe send "Unrecoverable error" status to Movicon if this happens too many times
348     IF NOT(refreshTimer.In) THEN
349         refreshTimer.In := TRUE;
350         refreshTimer.PT := T#5s;
351     END_IF;
352     IF refreshTimer.In AND refreshTimer.Q THEN
353         refreshTimer.In := FALSE;
354         masterState := 100;
355     END_IF;
356 END_IF;
357 IF masterState = 9020 THEN
358     //No printers available in rack
359     //Send to movicon?
360     //Wait a while, then set master state to 200 to try again
361     //Maybe send "Unrecoverable error" status to Movicon if this happens too many times
362     IF NOT(refreshTimer.In) THEN
363         refreshTimer.In := TRUE;
364         refreshTimer.PT := T#5s;
365     END_IF;
366     IF refreshTimer.In AND refreshTimer.Q THEN
367         refreshTimer.In := FALSE;
368         masterState := 200;
369     END_IF;
370 END_IF;
371 IF masterState = 9030 THEN
372     //Unable to connect to LDs
373     //Send to movicon?
374     //Wait a while, then set master state to 300.
375     //Maybe send "Unrecoverable error" status to Movicon if this happens too many times
376     IF NOT(refreshTimer.In) THEN
377         refreshTimer.In := TRUE;
378         refreshTimer.PT := T#5s;
379         LDCommands.connect := FALSE;
380     END_IF;
381     IF refreshTimer.In AND refreshTimer.Q THEN
382         refreshTimer.In := FALSE;
383         masterState := 300;
384     END_IF;
385 END_IF;
386 IF masterState = 9040 THEN
387     //Unable to connect to TMs
388     //Send to movicon?
389     //Wait a while, then set master state to 300.
390     //Maybe send "Unrecoverable error" status to Movicon if this happens too many times
391     IF NOT(refreshTimer.In) THEN
392         refreshTimer.In := TRUE;
393         refreshTimer.PT := T#5s;
394         TMCommands.connect := FALSE;
395     END_IF;
```

Master Program_5

IPC Program Body

Master

```

396     IF refreshTimer.In AND refreshTimer.Q THEN
397         refreshTimer.In := FALSE;
398         masterState := 400;
399     END_IF;
400 END_IF;
401
402 IF masterState = 9210 THEN
403     //Invalid product ID from cutter, return to idle.
404     masterState := 1000;
405 END_IF;
406 IF masterState = 9220 THEN
407     //Invalid product ID from printer, return to idle.
408     masterState := 1000;
409 END_IF;
410
411
412 IF masterState = 9320 THEN
413     //No available printer is found, do something about this.
414     //Set printer text to "no available printer"? Prompt asking whether to cancel or wait?
415     //For now, loop printer search until a printer becomes available.
416     IF NOT(refreshTimer.In) THEN
417         refreshTimer.In := TRUE;
418         refreshTimer.PT := T#1s;
419     END_IF;
420     IF refreshTimer.In AND refreshTimer.Q THEN
421         refreshTimer.In := FALSE;
422         orderChosenPrinterNr := FindAvailablePrinter(statusArray:=printerStatus);
423         IF NOT(orderChosenPrinterNr = -1) THEN
424             masterState := 3210;
425         END_IF;
426     END_IF;
427 END_IF;
428 IF masterState = 9330 THEN
429     //Part required to pick up is not in stock. Wait until part has finished printing and added to storage
430     //Ask in movicon whether the customer wants to wait or cancel in the meantime?
431     IF orderCurrent.productID = 1 AND moviconSendStatus.ID1BracketPairNumber > 0 THEN
432         LDCommands.getFromKanbanStart := TRUE;
433         LDCommands.productID := orderCurrent.productID;
434         printerPartsInStore := TRUE;
435         masterState := 4000;
436     ELSIF orderCurrent.productID = 2 AND moviconSendStatus.ID2KeychainNumber > 0 THEN
437         LDCommands.getFromKanbanStart := TRUE;
438         LDCommands.productID := orderCurrent.productID;
439         printerPartsInStore := TRUE;
440         masterState := 4000;
441     END_IF;
442 END_IF;
443
444 IF masterState = 9410 THEN
445     //Cutter has given off an error before completing.
446     //Cannot be fixed without access to physical hardware to know what I can and cannot do about it .
447     masterState := 9410;
448 END_IF;

```

APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC Variables

Background

1-6-1-2. Background

1-6-1-2-1. Variables

Name	Data Type	Initial Value	AT	Retain	Constant	Comment
VAR						
index	INT			False	False	Basic index used for various iteration tasks
printerTempArray	ARRAY[0..19] OF INT			False	False	Temporary array to store information while it's converted to a format that Movicon is capable of reading (can't do arrays or structures inside structures)
printerNr	INT			False	False	Integer containing the number of the currently selected, available printer
partsToPrint	INT			False	False	Number of parts that needs to be printed, ranging from 1-5
firstCutterReadDone	BOOL			False	False	Cutter status has been successfully read for the first time.
firstPrinterReadDone	BOOL			False	False	Printer status has been successfully read for the first time.
VAR EXTERNAL						
printerStatus	ARRAY[0..19] OF ST_3DPrinterStatus				False	
moviconSendStatus	ST_MoviconStatus				False	
printerActivePrints	ARRAY[0..19] OF ST_3DPrinterCurrentPrintInfo				False	
printerCommand	ST_3DPrinterCommand				False	
moviconSendCommand	ST_MoviconCommands				False	
csvBools	ST_CsvTools				False	
firstBackgroundRunDone	BOOL				False	
printerProductInfo	ARRAY[1..2] OF STRING[256]				True	
masterState	INT				False	
LDCommands	ST_LDTools				False	
LDIakloStatus	ST_LDStatus				False	
LDFinssonStatus	ST_LDStatus				False	
LDRingstadStatus	ST_LDStatus				False	
backgroundWritingCommand	BOOL				False	
TMCommands	ST_TMTools				False	
cutterStatus	ST_CutterStatus				False	
productStatus	ARRAY[1..2] OF ST_ProductStatus				False	

1-6-1-2-2.Program Body

```

1  //Reset a few write commands once they are completed.
2  IF csvBools.writeCutterDxfStart AND csvBools.writeCutterDxfDone THEN
3      csvBools.writeCutterDxfStart := FALSE;
4  END_IF;
5  IF csvBools.writePrinterCommandStart AND csvBools.writePrinterCommandDone THEN
6      csvBools.writePrinterCommandStart := FALSE;
7  END_IF;
8
9  //Refresh cutter status
10 IF NOT(csvBools.readCutterStatusStart) AND NOT(csvBools.readCutterStatusDone) THEN
11     csvBools.readCutterStatusStart := TRUE;
12 END_IF;
13 IF csvBools.readCutterStatusStart AND csvBools.readCutterStatusDone THEN
14     csvBools.readCutterStatusStart := FALSE;
15     firstCutterReadDone := TRUE;
16 END_IF;
17
18 //Refresh printer status
19 IF NOT(csvBools.readPrinterStatusStart) AND NOT(csvBools.readPrinterStatusDone) THEN
20     csvBools.readPrinterStatusStart := TRUE;
21 END_IF;
22 IF csvBools.readPrinterStatusStart AND csvBools.readPrinterStatusDone THEN
23     csvBools.readPrinterStatusStart := FALSE;
24     firstPrinterReadDone := TRUE;
25 END_IF;
26
27 //Initial run of background program done
28 IF firstCutterReadDone AND firstPrinterReadDone THEN
29     firstBackgroundRunDone := TRUE;
30 END_IF;
31
32 IF firstBackgroundRunDone THEN
33     //Update list of current prints
34     FOR index := 0 TO SizeOfAry(printerStatus)-1 DO
35         IF printerStatus[index].printing='True' OR printerStatus[index].finished='True' THEN
36             //Something is being printed in this printer, update status array on the given index.
37             SubDelimiter(In:=ProcessGcodeName(gcodeName:=printerStatus[index].printJob),
OutStruct:=printerActivePrints[index], Delimiter:=_eDELIMITER#_SEMICOLON);
38             ELSIF printerStatus[index].printing='False' AND printerStatus[index].finished='False' THEN
39                 //Nothing is being printed here, update status.
40                 printerActivePrints[index].productID := 0;
41                 printerActivePrints[index].number := 0;
42             END_IF;
43         END_FOR;
44
45
46     //Confirm that buffer doesn't use unrealistic values, correct if it does
47     FOR index := 1 TO SizeOfAry(productStatus) DO
48         IF productStatus[index].buffer > 20 THEN
49             productStatus[index].buffer := 20;
50         ELSIF productStatus[index].buffer < 0 THEN
51             productStatus[index].buffer := 0;
52         END_IF;
53     END_FOR;
54
55     // Check if printer's bed and nozzle temperatures aren't too high. If yes, tell printer to do an emergency
shutdown.
56     // Protect against malicious or stupid intentions causing high enough temperatures to damage or

```

```

destroy equipment
57   FOR printerNr := 0 TO SizeOfAry(printerStatus)-1 DO
58       IF STRING_TO_INT(printerStatus[printerNr].bedTemp) > 100 OR STRING_TO_INT(printerStatus
[printerNr].nozzleTemp) > 400 AND NOT(masterState = 3210) AND NOT(masterState = 3220) THEN
59           //Printer with too large temperature detected
60           printerCommand.separator := 'sep=';
61           printerCommand.IP := CONCAT(In1:='$R$L', In2:=printerStatus[printerNr].IP);
62           printerCommand.command := 'shutdown';
63           //printerCommand.argument := 'emergency';
64           csvBools.writePrinterCommandStart := TRUE;
65       END_IF;
66   END_FOR;
67
68   // Check if a printer has finished printing. If yes, bring the part to kanban storage
69   FOR printerNr := SizeOfAry(printerStatus)-1 TO 0 BY -1 DO
70       IF (printerStatus[printerNr].finished = 'True') AND NOT(LDCommands.getFromPrinterStart) AND
NOT(LDCommands.getFromPrinterDone) THEN
71           // This printer has finished a print and hasn't been retrieved.
72           LDCommands.getFromPrinterStart := TRUE;
73           LDCommands.printerNr := index;
74       END_IF;
75       IF (printerStatus[printerNr].finished = 'True') AND LDCommands.getFromPrinterStart AND
LDCommands.getFromPrinterDone THEN
76           //The print has been picked up
77           productStatus[printerActivePrints[printerNr].productID].number := productStatus
[printerActivePrints[printerNr].productID].number + printerActivePrints[printerNr].number;
78           productStatus[printerActivePrints[printerNr].productID].printing := productStatus
[printerActivePrints[printerNr].productID].printing - printerActivePrints[printerNr].number;
79
80           //Tell printer that the part is picked up and that the printer is ready for use again.
Doing this will set finished to False.
81           printerCommand.separator := 'sep=';
82           printerCommand.IP := CONCAT(In1:='$R$L', In2:=printerStatus[printerNr].IP);
83           printerCommand.command := 'retrievedPrint';
84           csvBools.writePrinterCommandStart := TRUE;
85           LDCommands.getFromPrinterStart := FALSE;
86
87           //To prevent issues for another finished print, exit for-loop to prevent further finished
prints from being found.
88           //This way, it's guaranteed to only work on one print at a time.
89           EXIT;
90       END_IF;
91   END_FOR;
92
93
94
95   //Check buffers, print up to 5 new parts (per printer) if current stock is smaller than buffer
96   FOR index := 1 TO SizeOfAry(productStatus) DO
97       IF (productStatus[index].number+productStatus[index].printing) < productStatus[index].buffer
AND NOT(masterState=3210) AND NOT(masterState=3220) THEN
98           //Current number of parts in store + parts printing is not enough to fill the buffer
99           IF NOT(csvBools.writePrinterCommandStart) AND NOT
(csvBools.writePrinterCommandDone) THEN
100              //IF statements ensure that no other part of the program is currently trying to
write to the same file
101              printerNr := -1; //Set unreachable default value to check for no available
printer
102              printerNr := FindAvailablePrinter(statusArray:=printerStatus);
103              IF NOT(printerNr = -1) THEN

```

Master Program_5

IPC Program Body

Background

```

104                                     //If there is an available printer, use start sending write commands to
it
105                                     printerCommand.separator := 'sep=';
106                                     printerCommand.IP := CONCAT(In1:='$R$L', In2:=printerStatus
[printerNr].IP);
107                                     printerCommand.command := 'print';
108                                     printerCommand.argument := printerProductInfo[index];
109                                     //Find number of parts required left to print, counting current in
storage and currently printing
110                                     partsToPrint := productStatus[index].buffer - productStatus
[index].number - productStatus[index].printing;
111                                     //Ensure that no more than 5 are printed. If more than 5 parts are
needed, then 5 are subtracted and the next 1-5 will be printed next time Background loops.
112                                     printerCommand.number := INT_TO_STRING(LIMIT(MN:=INT#1,
In:=partsToPrint, MX:=INT#5));
113                                     csvBools.writePrinterCommandStart := TRUE;
114                                     backgroundWritingCommand := TRUE;
115                                     END_IF;
116                                     END_IF;
117                                     IF csvBools.writePrinterCommandStart AND csvBools.writePrinterCommandDone
THEN
118                                     //Note, method for adding to local database of prints removed because it
now updates in the top.
119                                     productStatus[index].printing := productStatus[index].printing +
STRING_TO_INT(printerCommand.number);
120                                     moviconSendCommand.startPrinterScript := TRUE; //Note, must be turned off
eventually
121                                     csvBools.writePrinterCommandStart := FALSE;
122                                     backgroundWritingCommand := FALSE;
123                                     END_IF;
124                                     EXIT;
125                                     END_IF;
126                                     END_FOR;
127
128
129                                     //-----
130
131                                     //Send updated statuses to Movicon
132                                     //Master state
133                                     moviconSendStatus.masterState := masterState;
134
135                                     //Product statuses
136                                     moviconSendStatus.ID1BracketPairNumber := productStatus[1].number;
137                                     moviconSendStatus.ID1BracketPairPrinting := productStatus[1].printing;
138                                     moviconSendStatus.ID2KeychainNumber := productStatus[2].number;
139                                     moviconSendStatus.ID2KeychainPrinting := productStatus[2].printing;
140
141                                     //Update buffer sizes in program
142                                     productStatus[1].buffer := moviconSendStatus.ID1BracketPairBuffer;
143                                     productStatus[2].buffer := moviconSendStatus.ID2KeychainBuffer;
144
145                                     //LD robots
146                                     IF LDFinssonStatus.connected AND NOT(LDFinssonStatus.busy) THEN
147                                         moviconSendStatus.mobileRobot01 := 1;
148                                     END_IF;
149                                     IF LDFinssonStatus.busy THEN
150                                         moviconSendStatus.mobileRobot01 := 2;
151                                     END_IF;
152                                     IF LDFinssonStatus.error OR NOT(LDFinssonStatus.connected) THEN

```

Master Program_5

IPC Program Body

Background

```

153     moviconSendStatus.mobileRobot01 := 0;
154 END_IF;
155
156 IF LDTakloStatus.connected AND NOT(LDTakloStatus.busy) THEN
157     moviconSendStatus.mobileRobot00 := 1;
158 END_IF;
159 IF LDTakloStatus.busy THEN
160     moviconSendStatus.mobileRobot00 := 2;
161 END_IF;
162 IF LDTakloStatus.error OR NOT(LDTakloStatus.connected) THEN
163     moviconSendStatus.mobileRobot00 := 0;
164 END_IF;
165
166 IF LDRingstadStatus.connected AND NOT(LDRingstadStatus.busy) THEN
167     moviconSendStatus.mobileRobot02 := 1;
168 END_IF;
169 IF LDRingstadStatus.busy THEN
170     moviconSendStatus.mobileRobot02 := 2;
171 END_IF;
172 IF LDRingstadStatus.error OR NOT(LDRingstadStatus.connected) THEN
173     moviconSendStatus.mobileRobot02 := 0;
174 END_IF;
175
176 //TM Arms
177 IF TMCommands.connectSuccessful AND NOT(TMCommands.armLD_jobStart) THEN
178     moviconSendStatus.cobot00 := 1;
179 END_IF;
180 IF TMCommands.connectSuccessful AND NOT(TMCommands.arm1_jobStart) THEN
181     moviconSendStatus.cobot01 := 1;
182 END_IF;
183 IF TMCommands.connectSuccessful AND NOT(TMCommands.arm2_jobStart) THEN
184     moviconSendStatus.cobot02 := 1;
185 END_IF;
186 IF TMCommands.armLD_jobStart THEN
187     moviconSendStatus.cobot00 := 2;
188 END_IF;
189 IF TMCommands.arm1_jobStart THEN
190     moviconSendStatus.cobot01 := 2;
191 END_IF;
192 IF TMCommands.arm2_jobStart THEN
193     moviconSendStatus.cobot02 := 2;
194 END_IF;
195 IF TMCommands.connectError OR NOT(TMCommands.connectSuccessful) THEN
196     moviconSendStatus.cobot00 := 0;
197     moviconSendStatus.cobot01 := 0;
198     moviconSendStatus.cobot02 := 0;
199 END_IF;
200
201 //Cutter
202 IF cutterStatus.done='True' AND cutterStatus.working='False' THEN
203     moviconSendStatus.laserCutter00 := 1;
204 END_IF;
205 IF cutterStatus.working='True' THEN
206     moviconSendStatus.laserCutter00 := 2;
207 END_IF;
208 IF cutterStatus.error='True' THEN
209     moviconSendStatus.laserCutter00 := 0;
210 END_IF;

```



```
211
212 //Printers
213 FOR index :=0 TO SizeOfAry(printerStatus)-1 DO
214     //Update printer status
215     // 0 = error
216     // 1 = idle
217     // 2 = printing
218     IF printerStatus[index].ready='True' THEN
219         printerTempArray[index] := 1;
220     END_IF;
221     IF printerStatus[index].printing='True' OR printerStatus[index].finished='True' THEN
222         printerTempArray[index] := 2;
223     END_IF;
224     IF printerStatus[index].connected='False' OR printerStatus[index].operational='False' OR
printerStatus[index].pausing='True' OR printerStatus[index].paused='True' THEN
225         printerTempArray[index] := 0;
226     END_IF;
227 END_FOR;
228 moviconSendStatus.printer00 := printerTempArray[0]; moviconSendStatus.printer10 := printerTempArray
[10];
229 moviconSendStatus.printer01 := printerTempArray[1]; moviconSendStatus.printer11 := printerTempArray
[11];
230 moviconSendStatus.printer02 := printerTempArray[2]; moviconSendStatus.printer12 := printerTempArray
[12];
231 moviconSendStatus.printer03 := printerTempArray[3]; moviconSendStatus.printer13 := printerTempArray
[13];
232 moviconSendStatus.printer04 := printerTempArray[4]; moviconSendStatus.printer14 := printerTempArray
[14];
233 moviconSendStatus.printer05 := printerTempArray[5]; moviconSendStatus.printer15 := printerTempArray
[15];
234 moviconSendStatus.printer06 := printerTempArray[6]; moviconSendStatus.printer16 := printerTempArray
[16];
235 moviconSendStatus.printer07 := printerTempArray[7]; moviconSendStatus.printer17 := printerTempArray
[17];
236 moviconSendStatus.printer08 := printerTempArray[8]; moviconSendStatus.printer18 := printerTempArray
[18];
237 moviconSendStatus.printer09 := printerTempArray[9]; moviconSendStatus.printer19 := printerTempArray
[19];
238 // ^Thanks Movicon for neither accepting array inputs nor structs inside other structs, would use the
earlier for loop for assigning if it did.
239 END_IF;
```

APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC Variables

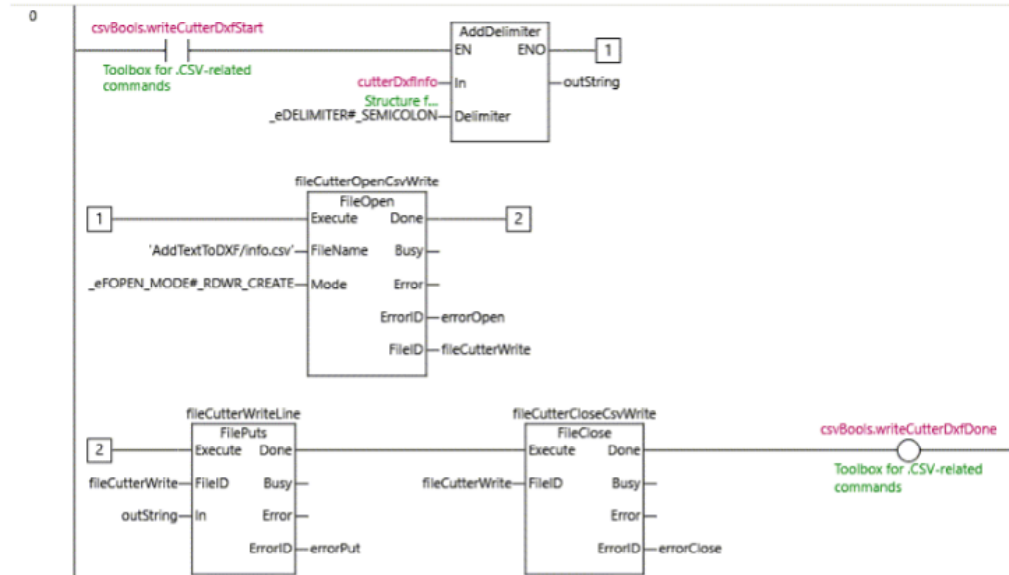
commsCSV

1-6-1-3.commsCSV

1-6-1-3-1.Variables

Name	Data Type	Initial Value	AT	Retain	Constant	Comment
VAR						
outString	STRING[256]			False	False	
fileCutterWrite	DWORD			False	False	
fileCutterRead	DWORD			False	False	
filePrinterWrite	DWORD			False	False	
filePrinterRead	DWORD			False	False	
offset	UINT			False	False	
state	UINT			False	False	
index	UINT			False	False	
errorOpen	WORD			False	False	
errorSeek	WORD			False	False	
errorPut	WORD			False	False	
errorRead	WORD			False	False	
errorClose	WORD			False	False	
fileCutterOpenCsvWrite	FileOpen			False	False	
fileCutterWriteLine	FilePuts			False	False	
fileCutterCloseCsvWrite	FileClose			False	False	
fileCutterOpenCsvRead	FileOpen			False	False	
fileCutterSetStartPoint	FileSeek			False	False	
fileCutterReadLine	FileGets			False	False	
fileCutterCloseCsvRead	FileClose			False	False	
filePrinterOpenCsvRead	FileOpen			False	False	
filePrinterSeek	FileSeek			False	False	
filePrinterReadLine	FileGets			False	False	
filePrinterCloseCsvRead	FileClose			False	False	
filePrinterOpenCsvWrite	FileOpen			False	False	
filePrinterWriteLine	FilePuts			False	False	
filePrinterCloseCsvWrite	FileClose			False	False	
VAR_EXTERNAL						
cutterDxfInfo	ST_CutterDxfExport				False	
cutterStatus	ST_CutterStatus				False	
csvBools	ST_CsvTools				False	
printerStatus	ARRAY[0..19] OF ST_3DPrinterStatus				False	
CardIReady	BOOL				True	
printerCommand	ST_3DPrinterCommand				False	

1-6-1-3-2.writeCutterDxf

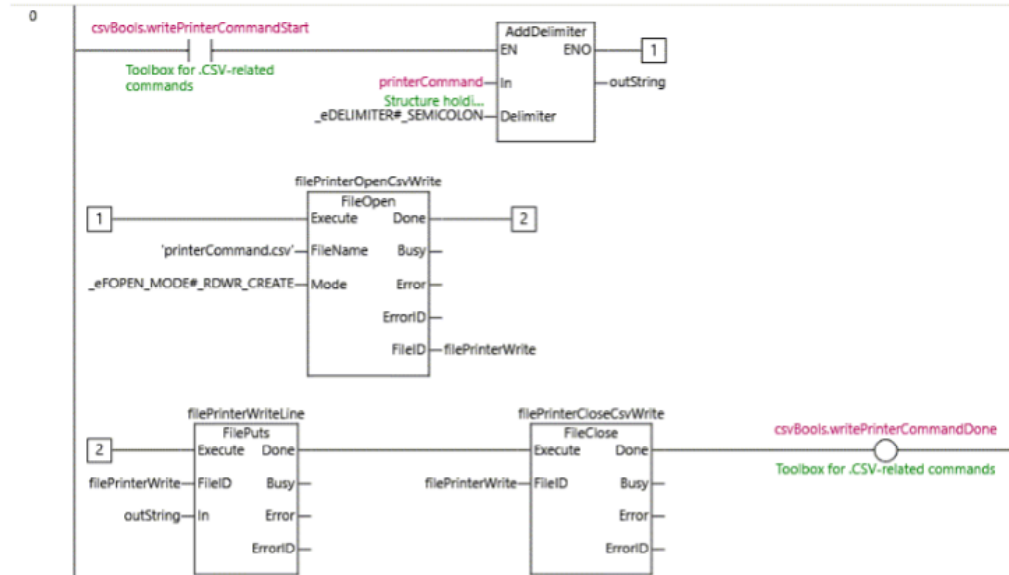


Master Program_5

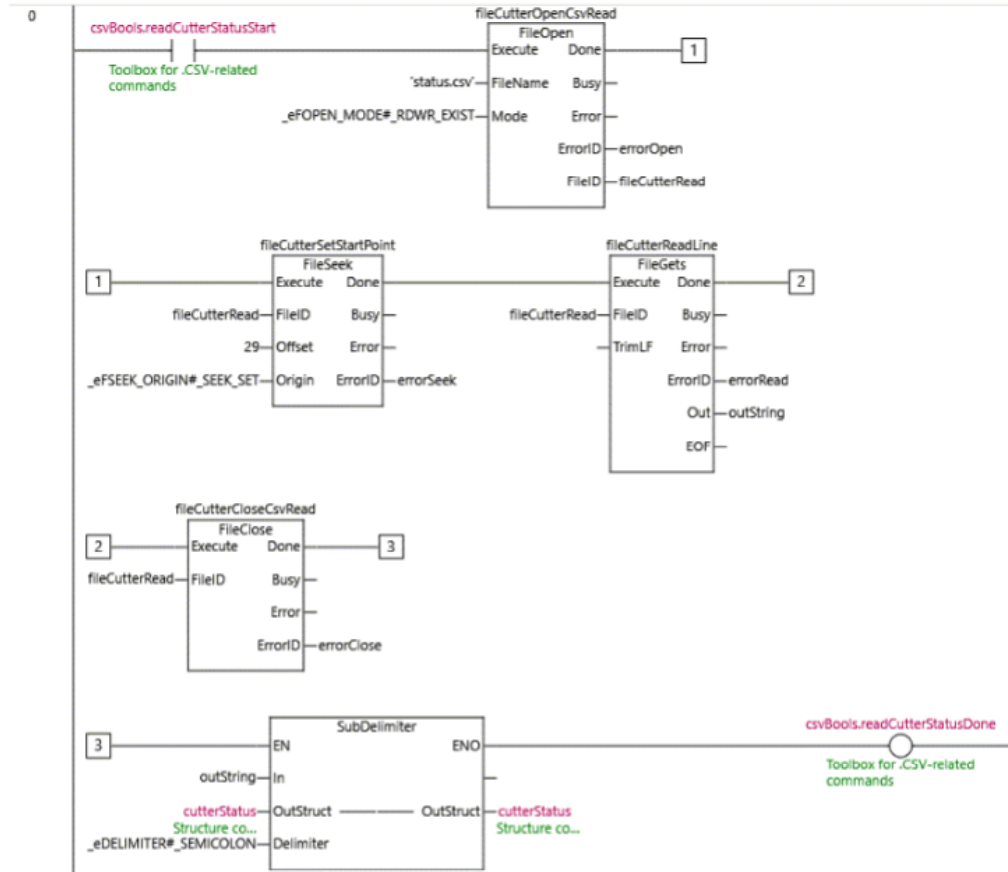
IPC writePrinterCommand

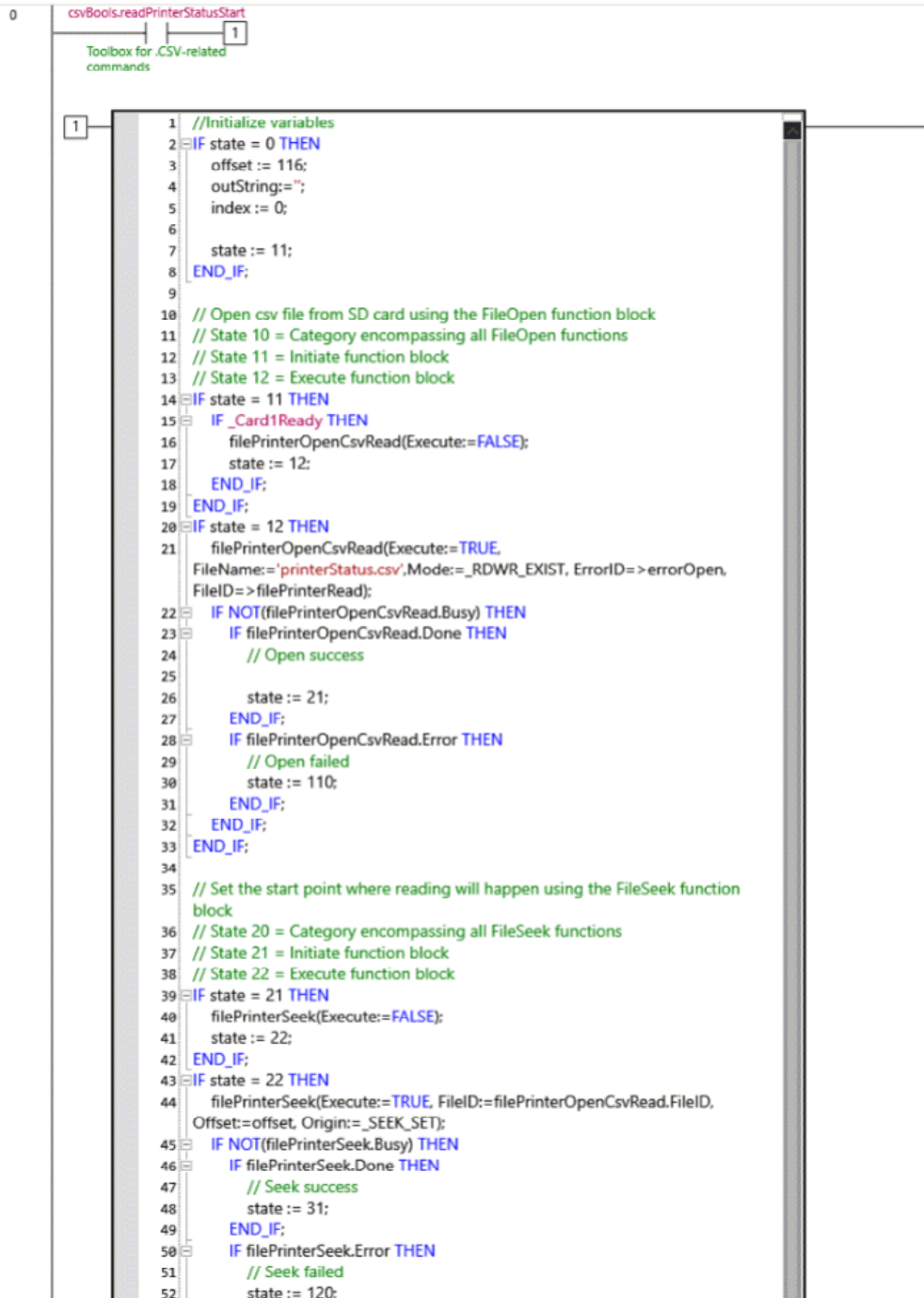
commsCSV writePrinterCommand

1-6-1-3-3.writePrinterCommand



1-6-1-3-4.readCutterStatus



1-6-1-3-5.readPrinterStatus

Master Program_5

IPC readPrinterStatus

commsCSV readPrinterStatus

```

53 |     END_IF;
54 | END_IF;
55 | END_IF;
56 |
57 | // Read the rest of the line from the start point set in state 20
58 | // State 30 = Category encompassing all FileGets functions
59 | // State 31 = Initiate function block
60 | // State 32 = Execute function block
61 | IF state = 31 THEN
62 |     filePrinterReadLine(Execute:=FALSE);
63 |     state := 32;
64 | END_IF;
65 | IF state = 32 THEN
66 |     filePrinterReadLine(Execute:=TRUE, FileID:=filePrinterOpenCsvRead.FileID);
67 |     Out:=>outString);
68 |     IF NOT(filePrinterReadLine.Busy) THEN
69 |         IF filePrinterReadLine.Done THEN
70 |             // Read success
71 |             state := 40;
72 |             END_IF;
73 |         IF filePrinterReadLine.Error THEN
74 |             // Read failed
75 |             state := 130;
76 |             END_IF;
77 |         END_IF;
78 |     END_IF;
79 | // 40 = Process data.
80 | // Increase the offset to go to the end of the line, so that the next FileSeek will
81 | // start from the next line.
82 | // Separate the string at each ';' and put each string segment into the
83 | // printerStatus struct.
84 | // Remove the $R$L linebreak from the end of the last string segment
85 | // Increase index by 1.
86 | // - If index is still within range of the printerStatus array, go back to state 20 and
87 | // repeat until this is no longer the case.
88 | // - If index becomes larger than total amount of elements in the printerStatus
89 | // array, continue with the program.
90 | IF state = 40 THEN
91 |     offset := offset + GetByteLen(In:=filePrinterReadLine.Out);
92 |     SubDelimiter(In:=filePrinterReadLine.Out, OutStruct:=printerStatus[index],
93 |     Delimiter:=_eDELIMITER#_SEMICOLON);
94 |     index := index + 1;
95 |     IF index > (SizeOfAry(printerStatus)-1) THEN
96 |         state := 51;
97 |     ELSIF index <= (SizeOfAry(printerStatus)-1) THEN
98 |         printerStatus[index].posY := LEFT(In:=printerStatus[index].posY, L:=LEN
99 |         (printerStatus[index].posY)-2);
100 |         state := 21;
101 |     END_IF;
102 | END_IF;
103 | END_IF;
104 |
105 | // Now that reading is complete, close the file with the FileClose function block
106 | // State 50 = Category encompassing all FileClose functions
107 | // State 51 = Initiate function block
108 | // State 52 = Execute function block
109 | IF state = 51 THEN
110 |     filePrinterCloseCsvRead(Execute:=FALSE);
111 |     state := 52;
112 | END_IF;
113 | IF state = 52 THEN
114 |     filePrinterCloseCsvRead(Execute:=TRUE, FileID:=filePrinterOpenCsvRead.FileID,

```

APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC readPrinterStatus

commsCSV readPrinterStatus

```
Done=>csvBools.readPrinterStatusDone);
109 IF NOT(filePrinterCloseCsvRead.Busy) THEN
110 IF filePrinterCloseCsvRead.Done THEN
111 // Closed successfully
112 state := 60;
113 END_IF;
114 IF filePrinterCloseCsvRead.Error THEN
115 // Close failed
116 state := 150;
117 END_IF;
118 END_IF;
119 END_IF;
```


APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC Variables

commsLD

1-6-1-4.commsLD

1-6-1-4-1.Variables

Name	Data Type	Initial Value	AT	Retain	Constant	Comment
VAR						
Taklo	ARCL Comms Li b\ARCL_ST_Conn ection			False	False	
Taklo_Status	STRING[30]			False	False	
Taklo_ErrorID	WORD			False	False	
Taklo_ErrorDesc	STRING[30]			False	False	
Taklo_LastMsg	STRING[200]			False	False	
connectToTaklo	ARCL Comms Li b\ARCL_Connect			False	False	
connectToFinns on	ARCL Comms Li b\ARCL_Connect			False	False	
Finns on	ARCL Comms Li b\ARCL_ST_Conn ection			False	False	
Finns on_ErrorI D	WORD			False	False	
Finns on_ErrorD esc	STRING[30]			False	False	
Finns on_Status	STRING[30]			False	False	
Finns on_LastMs g	STRING[200]			False	False	
connectToRingst ad	ARCL Comms Li b\ARCL_Connect			False	False	
Ringst ad	ARCL Comms Li b\ARCL_ST_Conn ection			False	False	
Ringst ad_ErrorI D	WORD			False	False	
Ringst ad_Error Desc	STRING[30]			False	False	
Ringst ad_Status	STRING[30]			False	False	
Ringst ad_jobName	STRING[25]			False	False	
Ringst ad_LastM sg	STRING[200]			False	False	
printerPickup	ARCL Comms Li b\ARCL_QPickup			False	False	
printerConveyor Arrived	BOOL			False	False	
printerDropoff	ARCL Comms Li b\ARCL_QDropoff			False	False	
kanbanPickup	ARCL Comms Li b\ARCL_QPickup			False	False	
kanbanDropoff	ARCL Comms Li b\ARCL_QDropoff			False	False	
cutterPickup	ARCL Comms Li b\ARCL_QPickup			False	False	
cutterDropoff	ARCL Comms Li b\ARCL_QDropoff			False	False	
deliveryPickup	ARCL Comms Li b\ARCL_QPickup			False	False	
deliveryDropoff	ARCL Comms Li b\ARCL_QDropoff			False	False	
printerPickupCo nv	ARCL Comms Li b\ARCL_QPickup			False	False	
printerDropoffC onv	ARCL Comms Li b\ARCL_QDropoff			False	False	
VAR EXTERNAL						
LDCCommands	ST_LDTools				False	
TMCCommands	ST_TMTTools				False	

APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

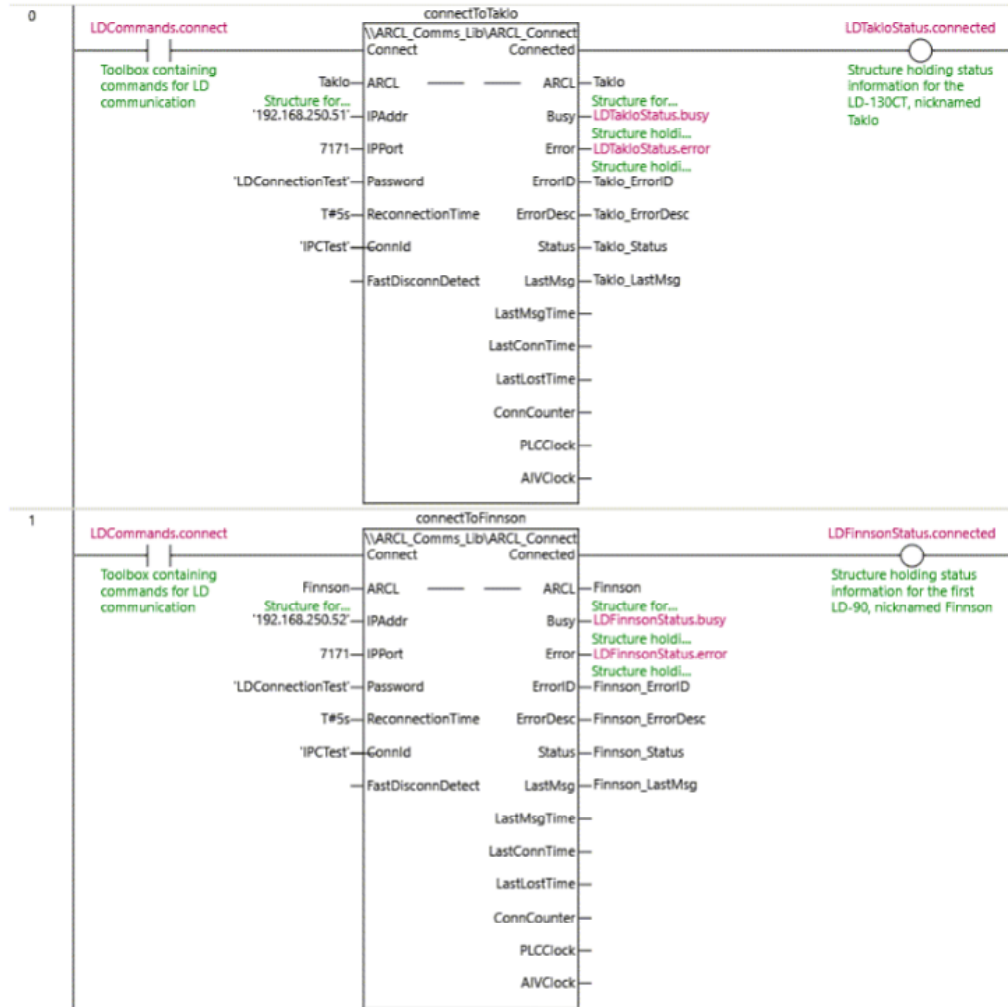
Master Program_5

IPC Variables

commsLD

printerStatus	ARRAY[0..19] OF ST_3DPrinterStatus				False	
LDTakloStatus	ST_LDStatus				False	
LDFinnsenStatus	ST_LDStatus				False	
LDRingstadStatus	ST_LDStatus				False	

1-6-1-4-2.ConnectToLDs

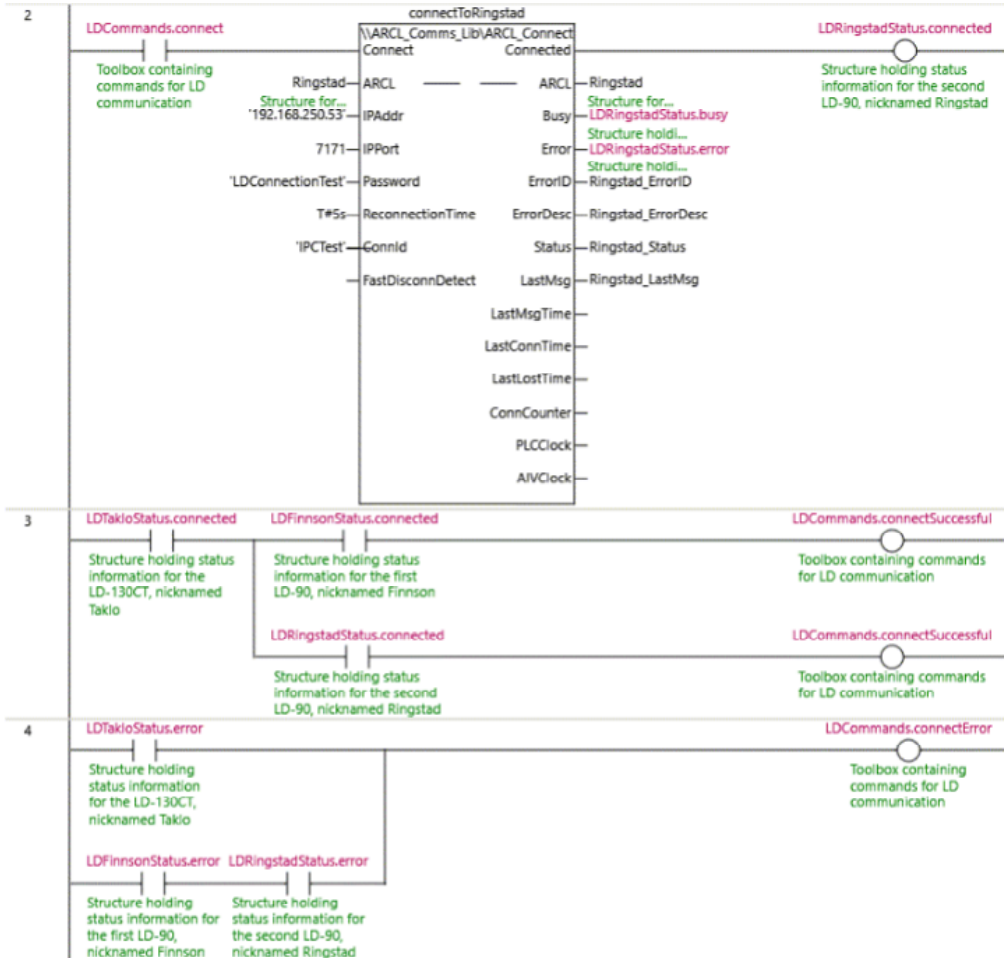


APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

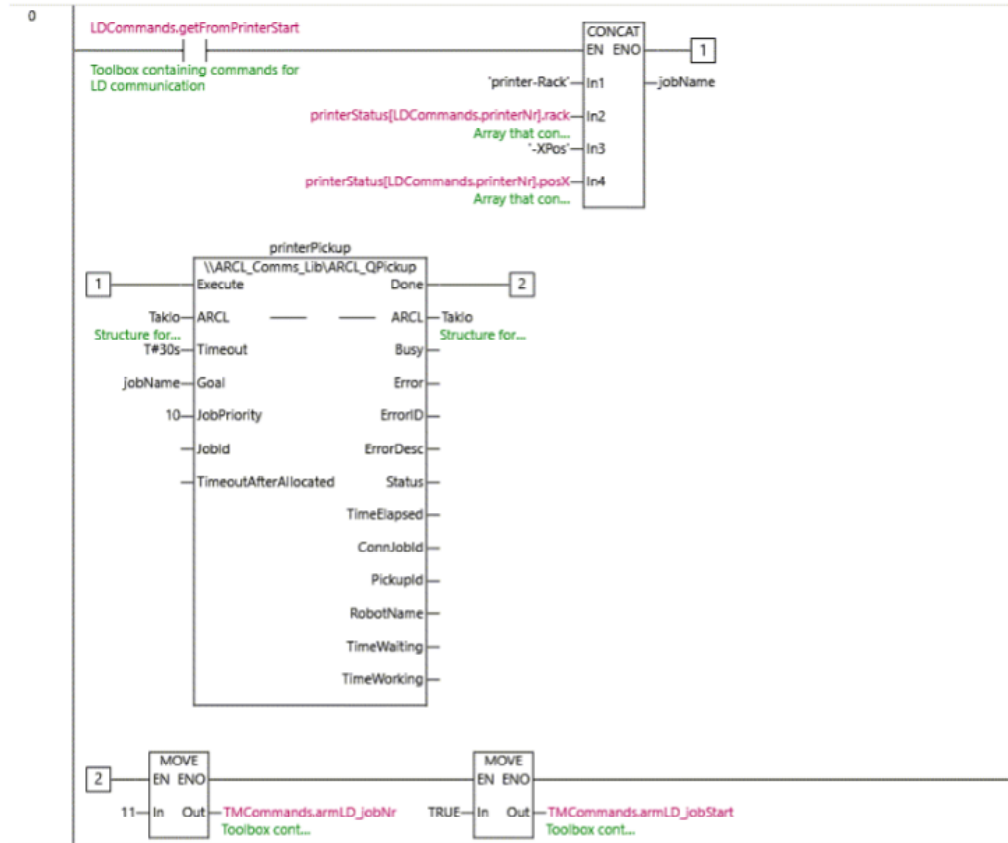
Master Program_5

IPC ConnectToLDs

commsLD ConnectToLDs



1-6-1-4-3.getFromPrinter

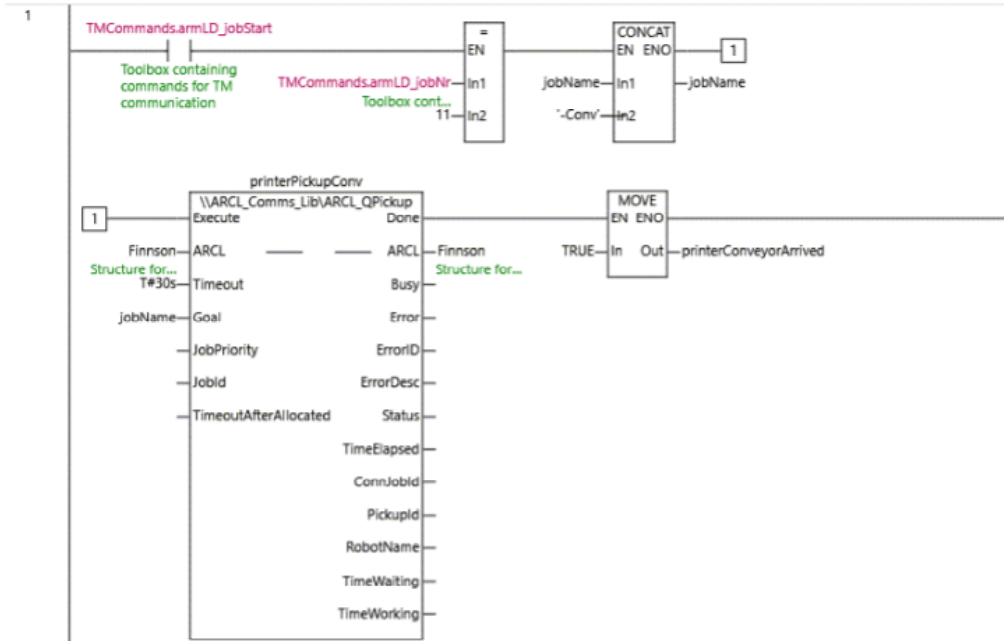


APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC getFromPrinter

commsLD getFromPrinter

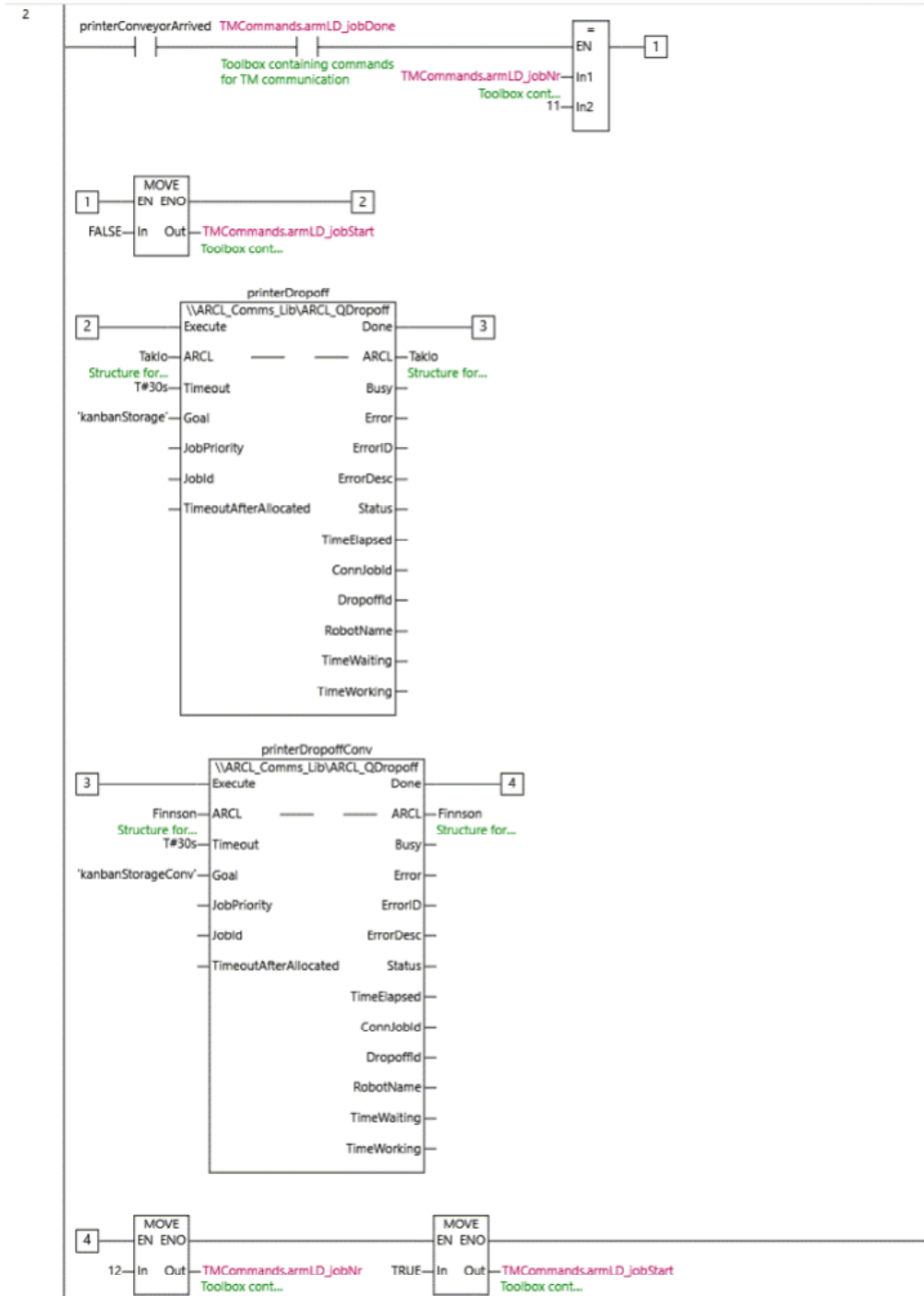


APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC getFromPrinter

commsLD getFromPrinter

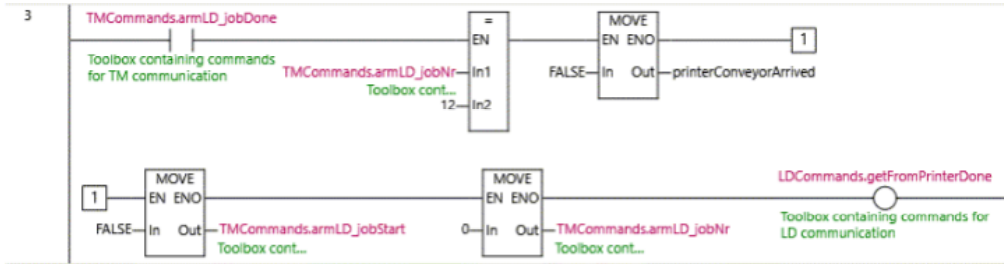


APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC getFromPrinter

commsLD getFromPrinter

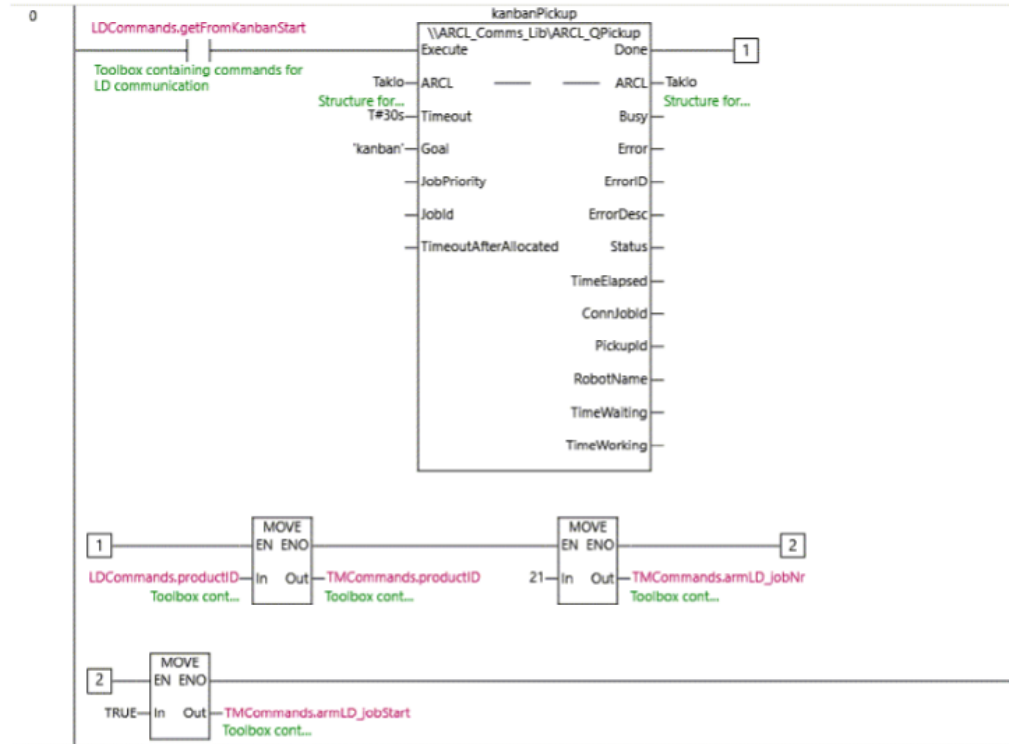


Master Program_5

IPC getFromKanban

commsLD getFromKanban

1-6-1-4-4.getFromKanban

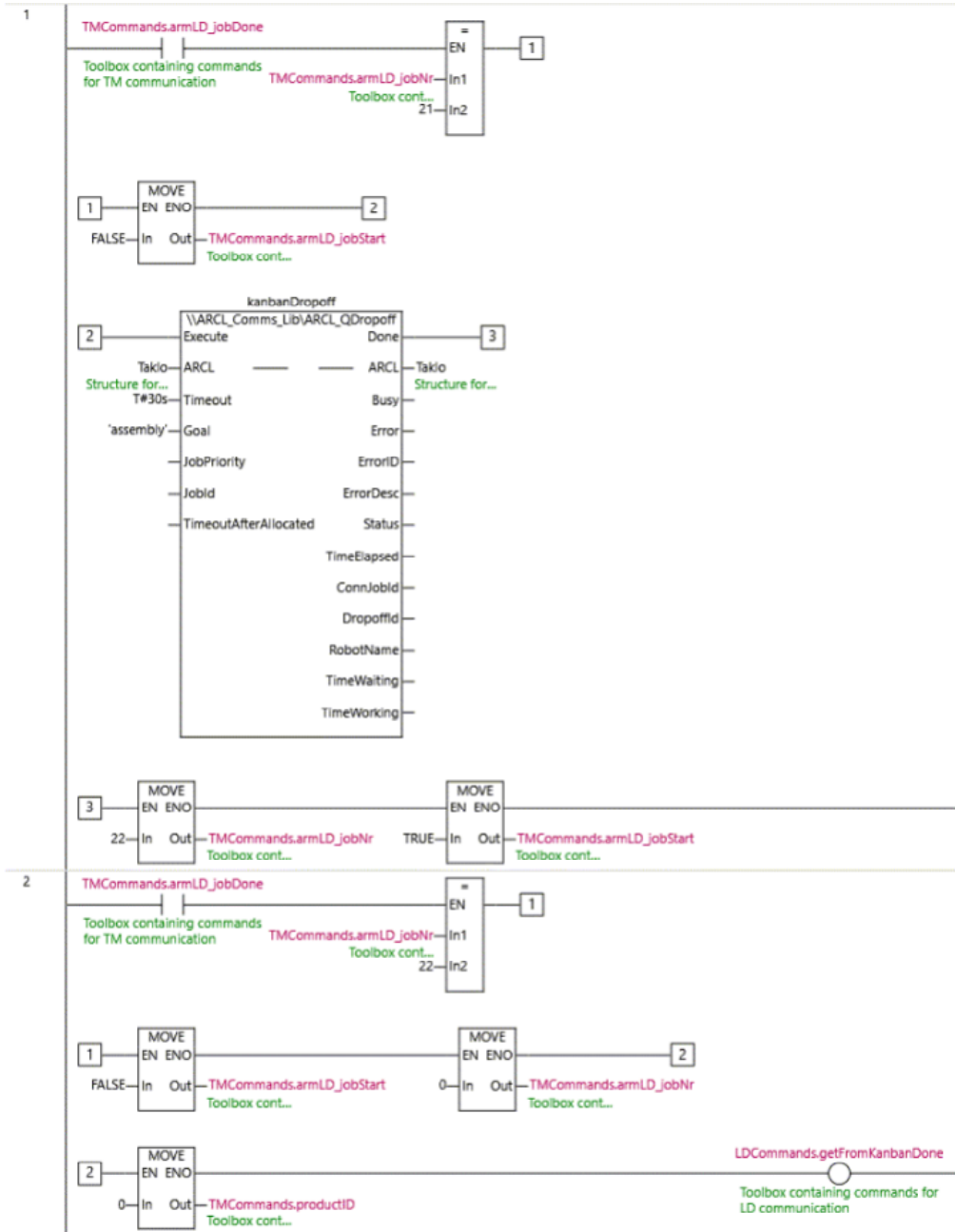


APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC getFromKanban

commsLD getFromKanban

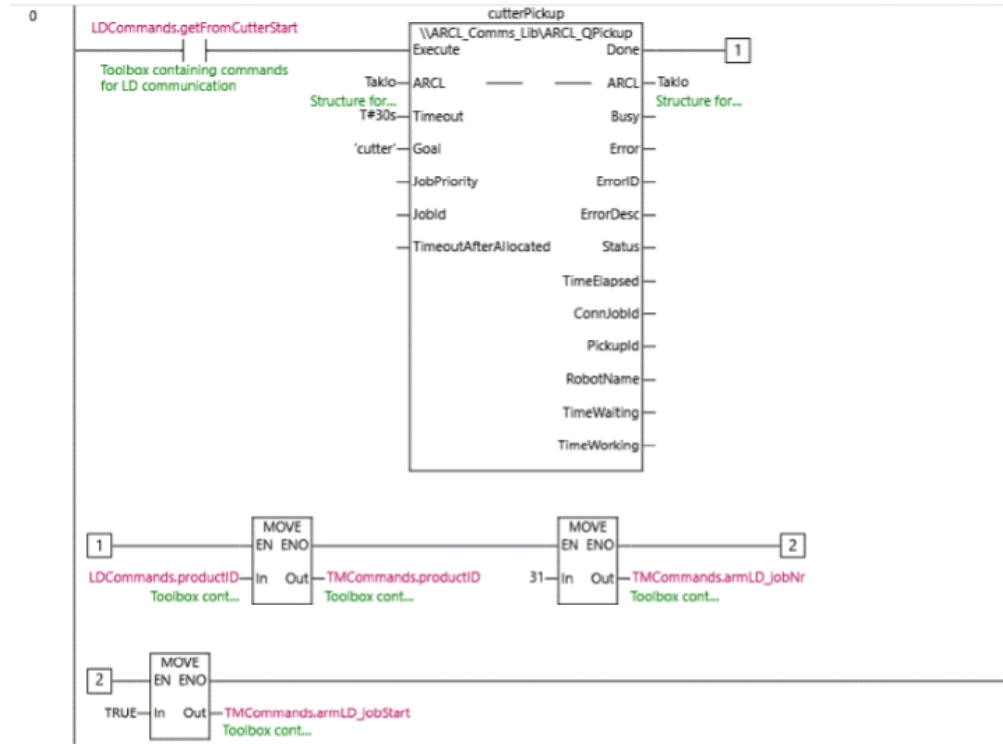


Master Program_5

IPC getFromCutter

commsLD getFromCutter

1-6-1-4-5.getFromCutter

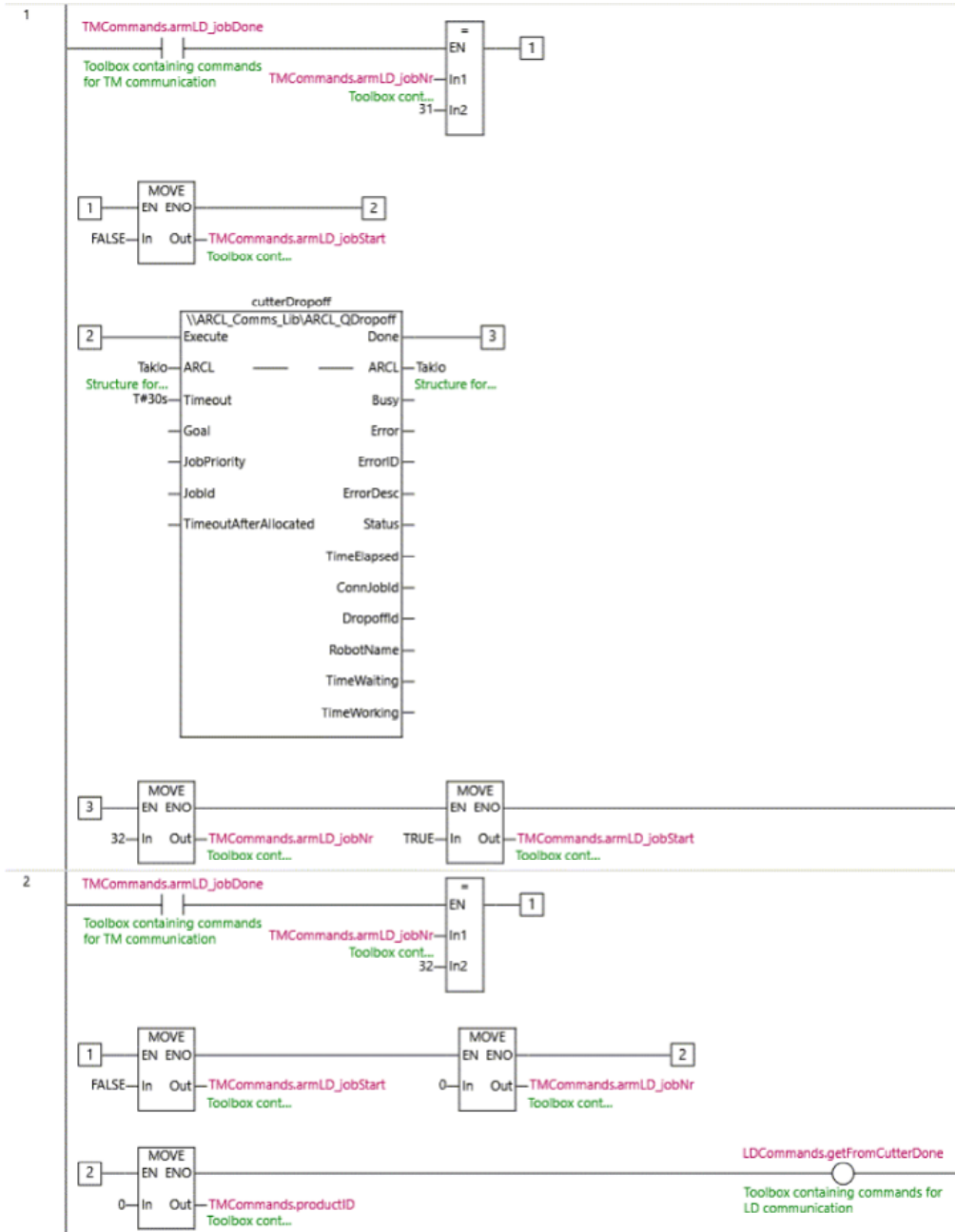


APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC getFromCutter

commsLD getFromCutter

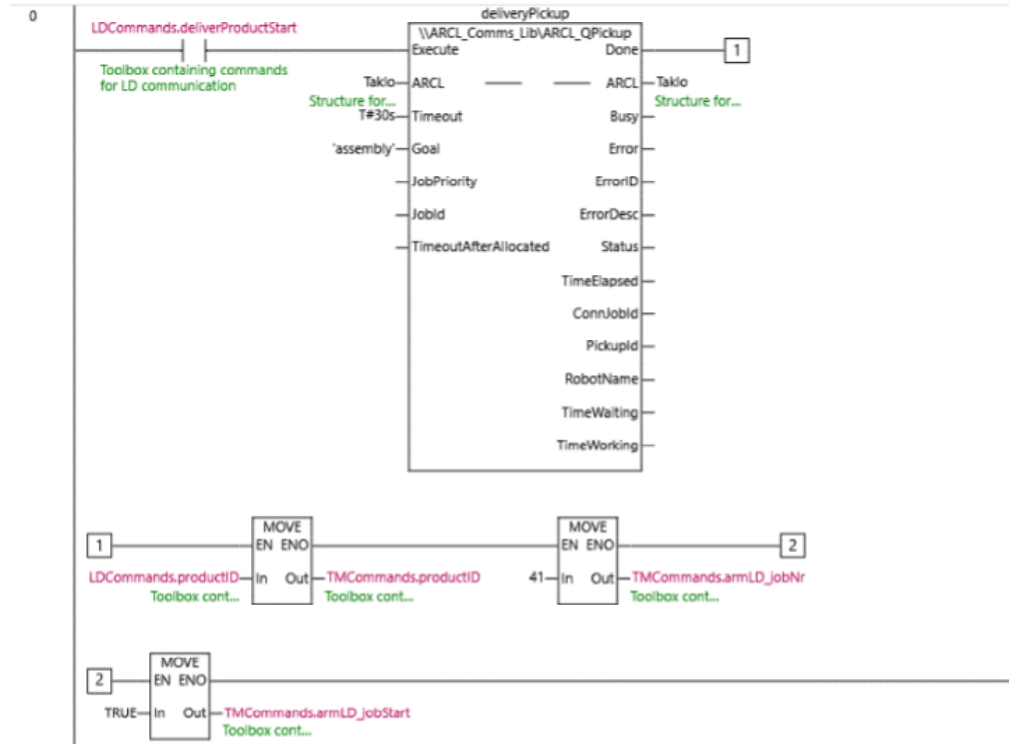


Master Program_5

IPC deliverProduct

commsLD deliverProduct

1-6-1-4-6.deliverProduct

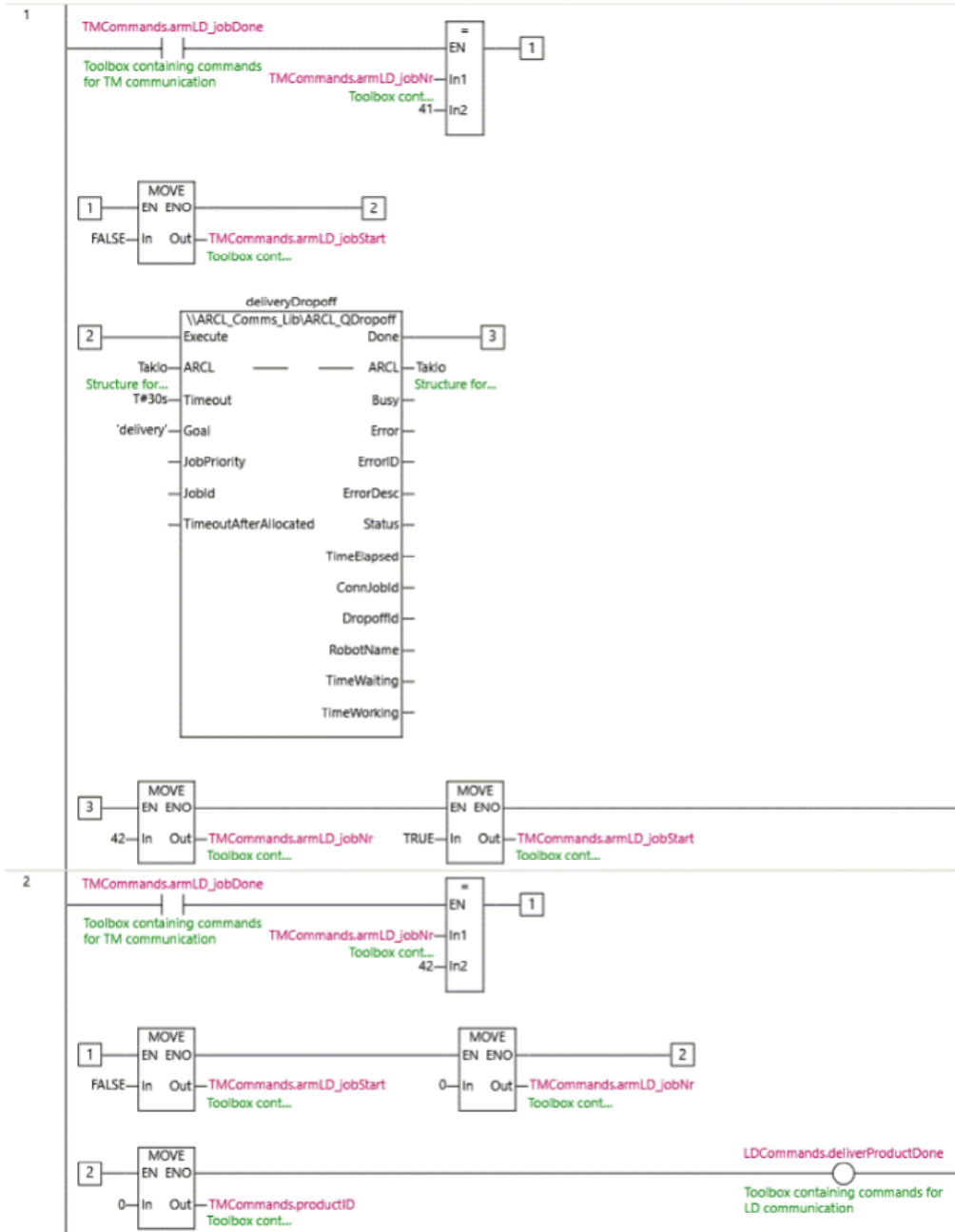


APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC deliverProduct

commsLD deliverProduct



APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC Variables

commsTM

1-6-1-5.commsTM

1-6-1-5-1.Variables

Name	Data Type	Initial Value	AT	Retain	Constant	Comment
VAR						
FBModbus_ArmlD	OEN\Eth\TCP_Connect			False	False	
FBModbus_ArmlD_State	eCONNECTION_STATE			False	False	
FBModbus_ArmlD_SocketError	BOOL.			False	False	
FBModbus_ArmlD_ErrorID	WORD			False	False	
FBModbus_ArmlD_ErrorIDtxt	STRING[100]			False	False	
FB_TM_ArmlD	OEN\Robot\TM\TM_ModbusTCP			False	False	
TM_testconnect	BOOL.			False	False	
FB_TMControl	OEN\Robot\TM\TM_Control			False	False	
ArmlD_PlayPause	BOOL.			False	False	
ArmlD_Stop	BOOL.			False	False	
ArmlD_ToggleAuto	BOOL.			False	False	
ArmlD_Mode	OEN\Robot\TM\TM_eMode			False	False	
ArmlD_Connected	BOOL.			False	False	
ArmlD_AttemptConnect	BOOL.			False	False	
jobStarted	BOOL.			False	False	
jobNr	INT			False	False	
jobStart	BOOL.			False	False	
jobDone	BOOL.			False	False	
ConnectTimer	TON			False	False	
Arm1_AttemptConnect	BOOL.			False	False	
FBModbus_Arm1	OEN\Eth\TCP_Connect			False	False	
FB_TM_Arm1	OEN\Robot\TM\TM_ModbusTCP			False	False	
FB_TMControl_Arm1D	OEN\Robot\TM\TM_Control			False	False	
FB_TMControl_Arm1	OEN\Robot\TM\TM_Control			False	False	
Arm1_PlayPause	BOOL.			False	False	
Arm1_Stop	BOOL.			False	False	
Arm1_ToggleAuto	BOOL.			False	False	
Arm1_Mode	OEN\Robot\TM\TM_eMode			False	False	
Arm1_Connected	BOOL.			False	False	
Arm2_AttemptConnect	BOOL.			False	False	
FBModbus_Arm2	OEN\Eth\TCP_Connect			False	False	
FBModbus_Arm2_State	eCONNECTION_STATE			False	False	
FBModbus_Arm2_SocketError	BOOL.			False	False	

APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC Variables

commsTM

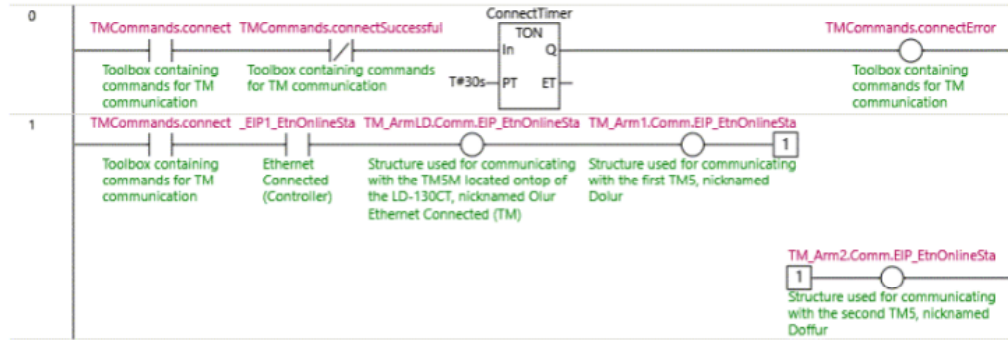
FBModbus Arm2_ErrorID	WORD			False	False	
FBModbus Arm2_ErrorIDtxt	STRING[100]			False	False	
FB_TM_Arm2	OENRobotTM\TM_ModbusTCP			False	False	
Arm2_PlayPause	BOOL			False	False	
Arm2_Stop	BOOL			False	False	
Arm2_ToggleAuto	BOOL			False	False	
Arm2_Mode	OENRobotTM\TM_eMode			False	False	
Arm2_Connected	BOOL			False	False	
armLD_jobStarted	BOOL			False	False	
arm1_jobStarted	BOOL			False	False	
arm2_jobStarted	BOOL			False	False	
VAR_EXTERNAL						
EIP1_EtnOnlineSta	BOOL			True		Ethernet Connected (Controller)
TM_ArmLD	OENRobotTM\sRobot			False		
TMCommands	ST_TMTools			False		
TM_Arm1	OENRobotTM\sRobot			False		
TM_Arm2	OENRobotTM\sRobot			False		

Master Program_5

IPC ModbusTCP_Setup

commsTM ModbusTCP_Setup

1-6-1-5-2.ModbusTCP_Setup



APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC ModbusTCP_Setup

commsTM ModbusTCP_Setup

```

2  TM_ArmLD.Comm.EIP_EtnOnlineSta
   Structure used for communicating
   with the TMSM located ontop of
   the LD-130CT, nicknamed Olur
   Ethernet Connected (TM)
1  1  TM_ArmLD.Comm.MBus_Port_TM:=UINT#502;           //Robot TCP-port for
   Modbus communication
2  TM_ArmLD.Comm.IPAddress:='192.168.250.150';
3
4  //Activate Modbus Reading
5  TM_ArmLD.DataSelect.Coordinates_Reg1B59thru1B94:=TRUE; //Activate reading
   of coordinates
6  TM_ArmLD.DataSelect.Others_Reg1CACthru1CB7:=TRUE; // Activate reading of
   general signals
7  TM_ArmLD.DataSelect.Status:=TRUE;                 //Activate reading of
   Status signals
8  TM_ArmLD.DataSelect.ErrorCode:=TRUE;              //Activate reading of
   Last Error Code
9  TM_ArmLD.DataSelect.Stick:=TRUE;                  // Read Speed and
   Mode
10 TM_ArmLD.DataSelect.Light:=TRUE;                  // Read Light Color
11 TM_ArmLD.DataSelect.UserINT:=TRUE;                // Read
   RegisterOutput#9000-9031 (Write has RegisterOutput#9100-9131)
12
13 TM_ArmLD.DataSelect.CBoxDI:=TRUE;                 // Read ControlBox
   Input signals Status
14 TM_ArmLD.DataSelect.CBoxDO:=TRUE;                 // Read ControlBox
   Output signals Status
15
16 // More Modbus Data available below if needed,but each one will slightly reduce
   comm response
17 // If you set all to TRUE, the total responsetime is about 0.5s
18 TM_ArmLD.DataSelect.CBoxAI:=FALSE; TM_ArmLD.DataSelect.CBoxAO:=FALSE;
19 TM_ArmLD.DataSelect.EndAI:=FALSE; TM_ArmLD.DataSelect.EndDI:=FALSE;
   TM_ArmLD.DataSelect.EndDO:=FALSE;
20 TM_ArmLD.DataSelect.Inertia_MassCenter:=FALSE;TM_ArmLD.DataSelect.JointTor
   que:=FALSE; TM_ArmLD.DataSelect.TouchStop:=FALSE;
21 TM_ArmLD.DataSelect.CollabMode:=FALSE;
   TM_ArmLD.DataSelect.SafetyStopCriteria:=FALSE;
22 TM_ArmLD.DataSelect.UserBOOL:=FALSE;
   TM_ArmLD.DataSelect.UserFloat:=FALSE;
23
24 //Modbus Writing is on change except UserINT and UserFloat
25
26 ArmLD_AttemptConnect := TRUE;
27
28

```



```

4  TM_Arm2.Comm.EIP_EtrOnlineSta
   Structure used for communicating
   with the second TMS, nicknamed
   Doffur
1  1  TM_Arm2.Comm.MBus_Port_TM:=UINT#502;           //Robot TCP-port for
   Modbus communication
2  TM_Arm2.Comm.IPAddress:='192.168.250.152';
3
4  //Activate Modbus Reading
5  TM_Arm2.DataSelect.Coordinates_Reg1B59thru1B94:=TRUE; //Activate reading
   of coordinates
6  TM_Arm2.DataSelect.Others_Reg1CACthru1CB7:=TRUE; // Activate reading of
   general signals
7  TM_Arm2.DataSelect.Status:=TRUE;                //Activate reading of
   Status signals
8  TM_Arm2.DataSelect.ErrorCode:=TRUE;            //Activate reading of
   Last Error Code
9  TM_Arm2.DataSelect.Stick:=TRUE;                // Read Speed and
   Mode
10 TM_Arm2.DataSelect.Light:=TRUE;                // Read Light Color
11 TM_Arm2.DataSelect.UserINT:=TRUE;              // Read
   RegisterOutput#9000-9031 (Write has RegisterOutput#9100-9131)
12
13 TM_Arm2.DataSelect.CBoxDI:=TRUE;               // Read ControlBox Input
   signals Status
14 TM_Arm2.DataSelect.CBoxDO:=TRUE;              // Read ControlBox
   Output signals Status
15
16 // More Modbus Data available below if needed,but each one will slightly reduce
   comm response
17 // If you set all to TRUE, the total responsetime is about 0.5s
18 TM_Arm2.DataSelect.CBoxAI:=FALSE; TM_Arm2.DataSelect.CBoxAO:=FALSE;
19 TM_Arm2.DataSelect.EndAI:=FALSE; TM_Arm2.DataSelect.EndDI:=FALSE;
   TM_Arm2.DataSelect.EndDO:=FALSE;
20 TM_Arm2.DataSelect.Inertia_MassCenter:=FALSE;TM_Arm2.DataSelect.JointTorqu
   e:=FALSE; TM_Arm2.DataSelect.TouchStop:=FALSE;
21 TM_Arm2.DataSelect.CollabMode:=FALSE;
   TM_Arm2.DataSelect.SafetyStopCriteria:=FALSE;
22 TM_Arm2.DataSelect.UserBOOL:=FALSE; TM_Arm2.DataSelect.UserFloat:=FALSE;
23
24 //Modbus Writing is on change except UserINT and UserFloat
25
26 Arm2_AttemptConnect := TRUE;
27
28

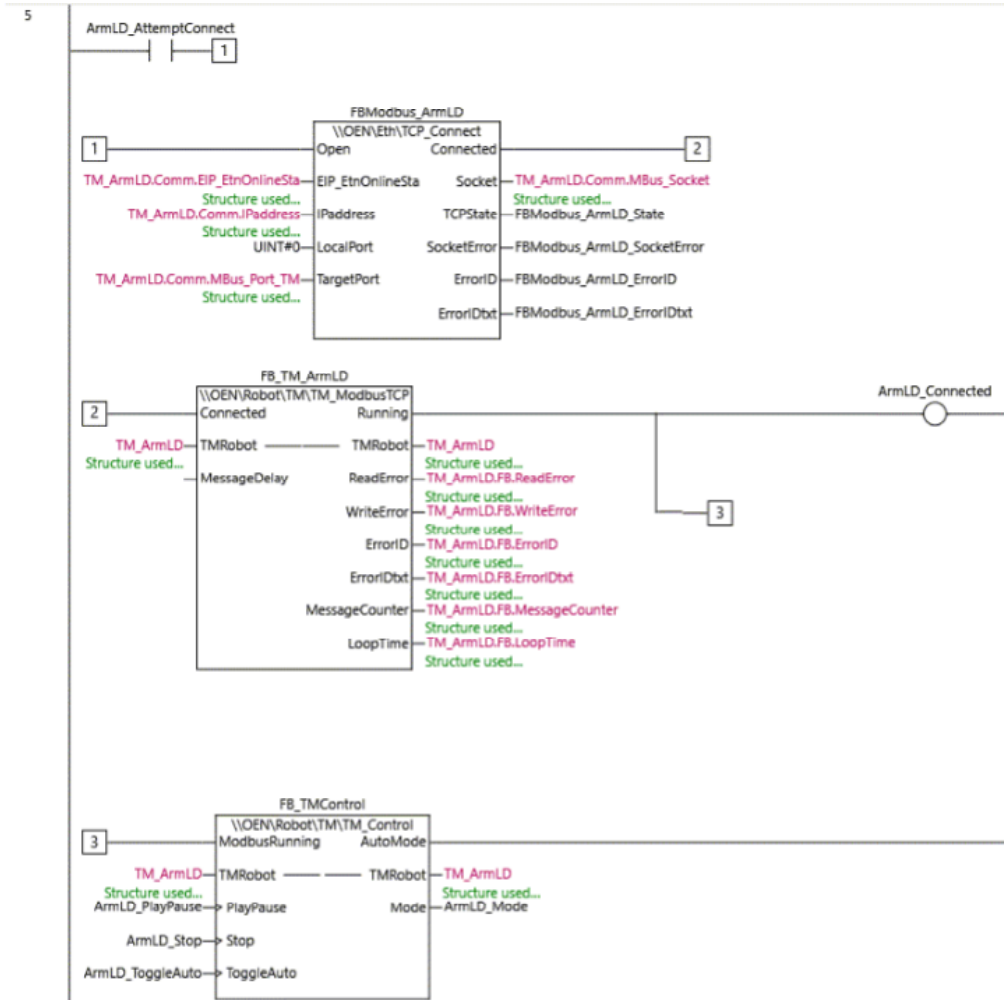
```

APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC ModbusTCP_Setup

commsTM ModbusTCP_Setup



APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC ModbusTCP_Setup

commsTM ModbusTCP_Setup

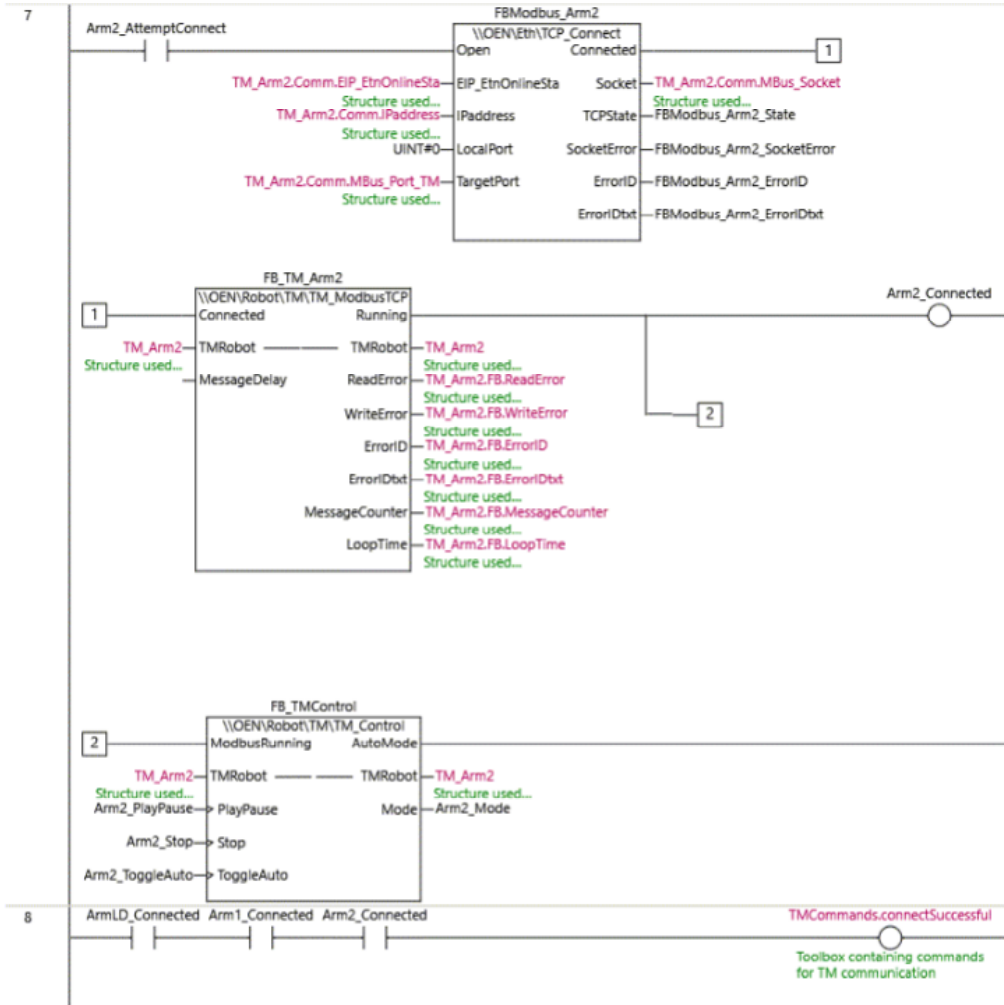


APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

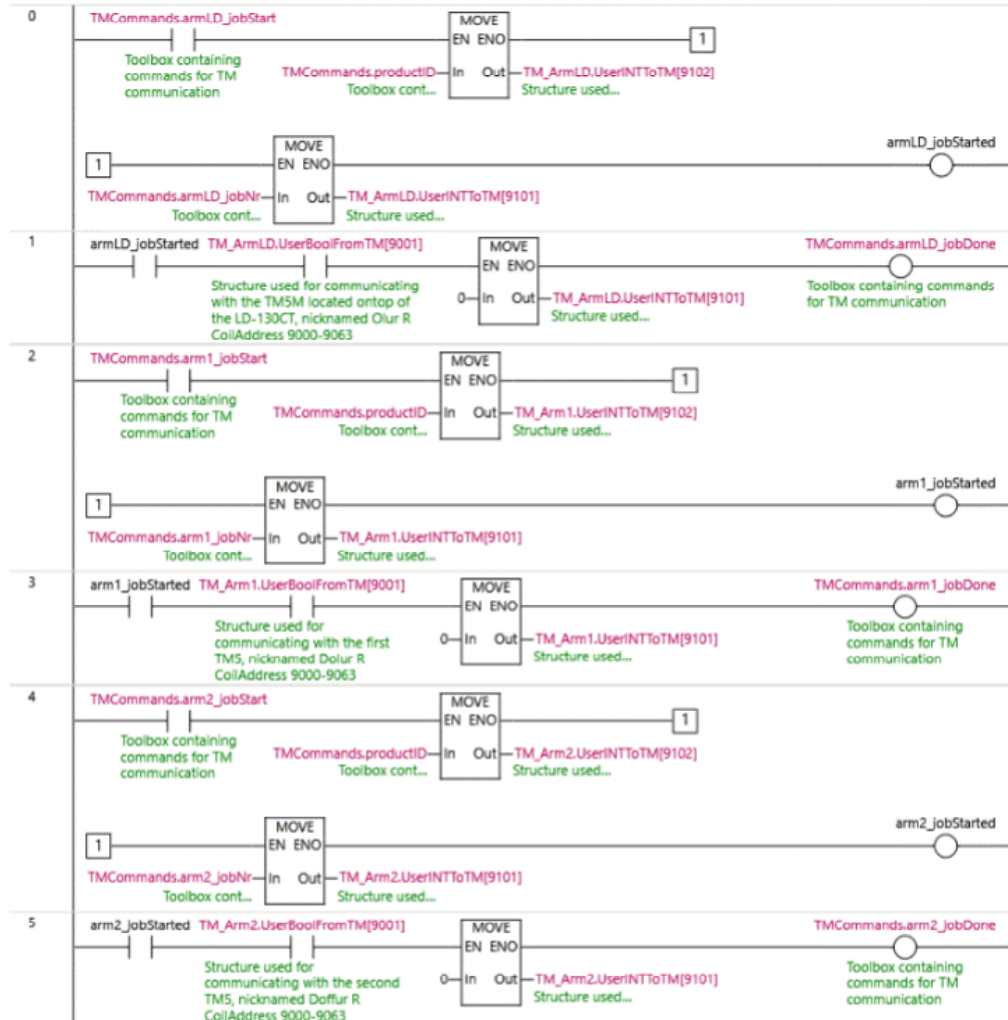
Master Program_5

IPC ModbusTCP_Setup

commsTM ModbusTCP_Setup



1-6-1-5-3.doTask



APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC Variables

UDP_Socket

1-6-1-6.UDP Socket

1-6-1-6-1.Variables

Name	Data Type	Initial Value	AT	Retain	Constant	Comment
VAR						
start	BOOL.			False	False	
iSkICreate	SkIUDPCreate			False	False	
state	DINT			False	False	
iSkISend	SkIUDPSend			False	False	
iSkIReceive	SkIUDPRcv			False	False	
iSkIClose	SkIClose			False	False	
iSkIClearBuf	SkIClearBuf			False	False	
Socket	sSOCKET			False	False	
SendStr	STRING[256]			False	False	
Packet	ARRAY[0..1999] OF byte			False	False	
SendSize	UINT			False	False	
RcvPacket	ARRAY[0..1999] OF BYTE			False	False	
RcvStr	string[256]			False	False	
Success	BOOL.			False	False	
VAR EXTERNAL						
FIP_EtnOnline Sta	BOOL.				True	

Master Program_5

IPC Program Body

UDP_Socket

1-6-1-6-2.Program Body

```

1  IF state = 0 THEN
2    IF _EIP_EtnOnlineSta THEN
3      iSktCreate(Execute := FALSE, SrcUdpPort := 6000, Socket => Socket);
4      state := 10;
5    END_IF;
6  END_IF;
7
8  IF state = 10 THEN
9    iSktCreate(Execute := TRUE, SrcUdpPort := 6000, Socket => Socket);
10   IF NOT iSktCreate.Busy THEN
11     IF iSktCreate.Done THEN
12       Socket.DstAdr.IpAdr := '192.168.250.200';
13       Socket.DstAdr.PortNo := 6000;
14       state := 20;
15     END_IF;
16
17     IF iSktCreate.Error THEN
18       state := 9000;
19     END_IF;
20   END_IF;
21 END_IF;
22
23 IF state = 20 THEN
24   iSktClearBuf(Execute := FALSE, Socket := Socket);
25   state := 30;
26 END_IF;
27
28 IF state = 30 THEN
29   iSktClearBuf(Execute := TRUE, Socket := Socket);
30   IF NOT iSktClearBuf.Busy THEN
31     state := 40;
32   END_IF;
33 END_IF;
34
35 IF state = 40 THEN
36   iSktReceive(Execute := FALSE, Socket := Socket, Size := 2000, RcvDat := RcvPacket[0]);
37   state := 50;
38 END_IF;
39
40 IF state = 50 THEN
41   iSktReceive(Execute := TRUE, Socket := Socket, TimeOut := 20, Size := 2000, RcvDat := RcvPacket[0]);
42   IF NOT iSktReceive.Busy THEN
43     IF iSktReceive.Done THEN
44       AryByteTo(In := RcvPacket[0], Size := iSktReceive.RcvSize, OutVal := RcvStr);
45       IF RcvStr = 'Hello World' THEN
46         Success := TRUE;
47       ELSE
48         Success := FALSE;
49       END_IF;
50       state := 60;
51     END_IF;
52
53     IF iSktReceive.Error THEN
54       state := 9000;
55     END_IF;
56   END_IF;
57 END_IF;

```

Master Program_5

IPC Program Body

UDP_Socket

```
58
59 IF state = 60 THEN
60   SendStr := 'Hello World';
61   SendSize := ToAryByte(In := SendStr, AryOut := Packet[0]);
62   iSkSend(Execute := FALSE, Socket := Socket, SendDat := Packet[0], Size := SendSize);
63   state := 70;
64 END_IF;
65
66 IF state = 70 THEN
67   iSkSend(Execute := TRUE, Socket := Socket, SendDat := Packet[0], Size := SendSize);
68   IF NOT iSkSend.Busy THEN
69     IF iSkSend.Done THEN
70       state := 20;
71     END_IF;
72
73     IF iSkSend.Error THEN
74       state := 9000;
75     END_IF;
76   END_IF;
77 END_IF;
78
79 IF state = 9000 THEN
80   IF Socket.Handle <> 0 THEN
81     state := 9010;
82   ELSE
83     state := 0;
84   END_IF;
85 END_IF;
86
87 IF state = 9010 THEN
88   iSkClose(Execute := FALSE, Socket := Socket);
89   state := 9020;
90 END_IF;
91
92 IF state = 9020 THEN
93   iSkClose(Execute := TRUE, Socket := Socket);
94
95   IF NOT iSkClose.Busy THEN
96     state := 0;
97   END_IF;
98 END_IF;
```

Master Program_5

IPC Variables

FindAvailablePrinter

1-6-2.Functions

1-6-2-1.FindAvailablePrinter

1-6-2-1-1.Variables

Name	Data Type	Edge	Initial Value	AT	Retain	Constant	Comment
VAR							
index	INT				False	False	Index for iterating
printerNr	INT				False	False	Available printer number
VAR INPUTOUTPUT							
EN	BOOL	No Edge			False	False	
statusArray	ARRAY[0..19] OF ST_3DPrinterStat us	No Edge			False	False	
RETURN							
FindAvailable Printer	INT						The array number for the available printer

Master Program_5

IPC Program Body

FindAvailablePrinter

1-6-2-1-2.Program Body

```
1 //Set default return value to -1, since this cannot naturally be achieved.
2 //Should the function return a -1, then there is no available printer to use.
3 //Usage of this function should thus be followed by an IF statement to check for -1.
4 printerNr := -1;
5
6 //Iterate over the array containing all printer statuses.
7 FOR index := SizeOfAry(statusArray)-1 TO 0 BY -1 DO
8 //Check to see if there is a printer that is ready for a new print and isn't in the "finished" state where the
  print is yet to be retrieved.
9   IF (statusArray[index].ready = 'True') AND (statusArray[index].finished = 'False') THEN
10     // This printer is ready to use, choose it as printer to assign work
11     printerNr := index;
12   END_IF;
13 END_FOR;
14
15 //Return either available printer or -1.
16 FindAvailablePrinter := printerNr;
```

APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC Variables

ProcessGcodeName

1-6-2-2.ProcessGcodeName

1-6-2-2-1.Variables

Name	Data Type	Edge	Initial Value	AT	Retain	Constant	Comment
VAR							
cursor	UINT				False	False	The current position in the string that is being worked from
productID	STRING[256]				False	False	The extracted product ID text
number	STRING[256]				False	False	The extracted number of products being printed
tempStr	STRING[256]				False	False	Temporary string used for processing
VAR INPUTOUTPUT							
EN	BOOL	No Edge			False	False	
gcodeName	STRING[256]	No Edge			False	False	.GCODE name
RETURN							
ProcessGcode Name	STRING[256]						Returned string containing product ID and number. -1 if unknown. Separate into a struct with SubDelimiter afterwards

1-6-2-2-2.Program Body

```

1 // EXPECTED FORMAT: "P[ID]_ProductName_[number]pcs.gcode"
2 // For example:
3 // - P1_BracketPair_1pcs.gcode
4 // - P192_WaluigiBust_10000pcs.gcode
5 // Function is able to detect letters inside the numbers, and can remove a single comma or period, but no
other symbols.
6
7 //Check the name of the gcode to see if it can be recognized. It must pass these tests:
8 // - First letter is 'P'
9 // - Last 9 letters is 'pcs.gcode'
10 // - There exists an underscore in the gcode name
11 // - There exists a second underscore in the remaining text after the end of the last underscore.
12 // - There does not exist a third underscore in the remaining text after the end of the second underscore
13 //If it does not pass, return "-1;-1".
14 cursor := FIND(In1:=gcodeName, In2:='_');
15 IF LEFT(In:=gcodeName, L:=1)='P' AND RIGHT(In:=gcodeName, L:=9)='pcs.gcode' AND NOT(cursor=0) AND
NOT(FIND(In1:=RIGHT(In:=gcodeName, L:=LEN(gcodeName)-cursor), In2:='_')=0) AND FIND(In1:=RIGHT
(In:=RIGHT(In:=gcodeName, L:=LEN(gcodeName)-cursor), L:=LEN(RIGHT(In:=gcodeName, L:=LEN(gcodeName)-
cursor))-FIND(In1:=RIGHT(In:=gcodeName, L:=LEN(gcodeName)-cursor), In2:='_'), In2:='_')=0) THEN
16 //A recognized part is being printed, update the status of this printer.
17
18 //Find product ID of what is being printed.
19 //Find first underscore
20 cursor := FIND(In1:=gcodeName, In2:='_');
21 //Remove P at the start and everything after (and including) the underscore.
22 tempStr := MID(In:=gcodeName, L:=(cursor-2), P:=2);
23 //Check if there are any letters in the remaining text. If yes, returning -1.
24 IF (tempStr = ToLCase(In:=tempStr)) AND (tempStr = ToUCase(In:=tempStr)) THEN
25 //No letter in the remaining numbers. Check for period or comma, and if one is detected,
remove it.
26 IF NOT(FIND(In1:=tempStr, In2:=',')=0) THEN
27 productID := DELETE(In:=tempStr, L:=1, P:=FIND(In1:=tempStr, In2:=','));
28 ELSIF NOT(FIND(In1:=tempStr, In2:=',')=0) THEN
29 productID := DELETE(In:=tempStr, L:=1, P:=FIND(In1:=tempStr, In2:=','));
30 ELSE
31 productID := tempStr;
32 END_IF;
33 ELSE
34 productID := '-1';
35 END_IF;
36
37 tempStr := RIGHT(In:=gcodeName, L:=(LEN(gcodeName)-cursor));
//Extracts everything after the underscore
38 cursor := FIND(In1:=tempStr, In2:='_');
//Find next
underscore
39 tempStr := MID(In:=tempStr, L:=(LEN(tempStr)-cursor-LEN('pcs.gcode')), P:=(cursor+1)); //Removes
everything up until and including underscore, and pcs.gcode.
40 IF (tempStr = ToLCase(In:=tempStr)) AND (tempStr = ToUCase(In:=tempStr)) THEN
41 IF NOT(FIND(In1:=tempStr, In2:=',')=0) THEN
42 number := DELETE(In:=tempStr, L:=1, P:=FIND(In1:=tempStr, In2:=','));
43 ELSIF NOT(FIND(In1:=tempStr, In2:=',')=0) THEN
44 number := DELETE(In:=tempStr, L:=1, P:=FIND(In1:=tempStr, In2:=','));
45 ELSE
46 number := tempStr;
47 END_IF;

```

APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC Program Body

ProcessGcodeName

```
48 ELSE
49     number := '-1';
50 END_IF;
51
52 //Combine into a format that SubDelimiter can split up
53 ProcessGcodeName := CONCAT(ln1:=productID, ln2:=';', ln3:=number);
54 ELSE
55     //Name is not recognized. Return unattainable value for ID and number.
56     ProcessGcodeName := '-1;-1';
57 END_IF;
```


APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC Data Types

1-7.Data

1-7-1.Data Types

Name	Type		Offset Byte	Offset Bit	Comment
ST_OrderDetails	STRUCT	NJ			
orderReceived	BOOL				
productID	UINT				
name	STRING[256]				
ST_MoviconCommands	STRUCT	NJ			
startDxfScript	BOOL				
startPrinterScript	BOOL				
ST_MoviconStatus	STRUCT	NJ			
ID1BracketPairNumber	INT				
ID1BracketPairPrinting	INT				
ID1BracketPairBuffer	INT				
ID2KeychainNumber	INT				
ID2KeychainPrinting	INT				
ID2KeychainBuffer	INT				
masterState	INT				
laserCutter00	INT				
mobileRobot00	INT				
mobileRobot01	INT				
mobileRobot02	INT				
cobot00	INT				
cobot01	INT				
cobot02	INT				
quattro00	INT				
viper00	INT				
viper01	INT				
printer00	INT				
printer01	INT				
printer02	INT				
printer03	INT				
printer04	INT				
printer05	INT				
printer06	INT				
printer07	INT				
printer08	INT				
printer09	INT				
printer10	INT				
printer11	INT				
printer12	INT				
printer13	INT				
printer14	INT				
printer15	INT				
printer16	INT				
printer17	INT				
printer18	INT				
printer19	INT				
ST_CsvTools	STRUCT	NJ			
writeCutterDxfStart	BOOL				
writeCutterDxfDone	BOOL				
writePrinterCommandStart	BOOL				

APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC Data Types

	writePrinterCommandDone	BOOL			
	readCutterStatusStart	BOOL			
	readCutterStatusDone	BOOL			
	readPrinterStatusStart	BOOL			
	readPrinterStatusDone	BOOL			
ST	CutterDxfExport	STRUCT	NJ	Offset Byte	Offset Bit
	separator	STRING[256]			
	finalFileName	STRING[256]			
	templateName	STRING[256]			
	logoFileName	STRING[256]			
	xInsertPointLogo	STRING[256]			
	yInsertPointLogo	STRING[256]			
	xPixelSize	STRING[256]			
	yPixelSize	STRING[256]			
	xSizeInMm	STRING[256]			
	ySizeInMm	STRING[256]			
	rotationLogo	STRING[256]			
	textToInsert	STRING[256]			
	textStyle	STRING[256]			
	xInsertPointText	STRING[256]			
	yInsertPointText	STRING[256]			
	rotationText	STRING[256]			
	alignmentText	STRING[256]			
	textWidth	STRING[256]			
	textHeight	STRING[256]			
	strNameLen	STRING[256]			
ST	CutterStatus	STRUCT	NJ	Offset Byte	Offset Bit
	working	STRING[256]			
	done	STRING[256]			
	error	STRING[256]			
ST	LDStatus	STRUCT	NJ	Offset Byte	Offset Bit
	connected	BOOL			
	busy	BOOL			
	error	BOOL			
ST	LDTools	STRUCT	NJ	Offset Byte	Offset Bit
	fleetManager	STRING[256]			
	connect	BOOL			
	taskList	ARRAY[0..10] OF STRING[256]			
	getFromKanbanStart	BOOL			
	getFromCutterStart	BOOL			
	getFromPrinterStart	BOOL			
	getFromKanbanDone	BOOL			
	getFromCutterDone	BOOL			
	getFromPrinterDone	BOOL			
	deliverProductStart	BOOL			
	deliverProductDone	BOOL			
	printerNr	INT			
	productID	INT			
	connectError	BOOL			
	connectSuccessful	BOOL			
ST	TMTools	STRUCT	NJ	Offset Byte	Offset Bit
	connect	BOOL			
	connectSuccessful	BOOL			

APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC Data Types

connectError	BOOL				
assemblyStart	BOOL				
assemblyDone	BOOL				
productID	INT				
armLD_jobStart	BOOL				
armLD_jobDone	BOOL				
arm1_jobStart	BOOL				
arm1_jobDone	BOOL				
arm2_jobStart	BOOL				
arm2_jobDone	BOOL				
armLD_jobNr	INT				
arm1_jobNr	INT				
arm2_jobNr	INT				
ST_3DPrinterCommand	STRUCT	NJ	Offset Byte	Offset Bit	
separator	STRING[256]				
IP	STRING[256]				
argument	STRING[256]				
command	STRING[256]				
number	STRING[256]				
ST_3DPrinterStatus	STRUCT	NJ	Offset Byte	Offset Bit	
IP	STRING[256]				
connected	STRING[256]				
printing	STRING[256]				
ready	STRING[256]				
operational	STRING[256]				
pausing	STRING[256]				
paused	STRING[256]				
finished	STRING[256]				
nozzleTemp	STRING[256]				
bedTemp	STRING[256]				
printJob	STRING[256]				
rack	STRING[256]				
posX	STRING[256]				
posY	STRING[256]				
ST_3DPrinterCurrentPrintInfo	STRUCT	NJ	Offset Byte	Offset Bit	
productID	INT				
number	INT				
ST_ProductStatus	STRUCT	NJ	Offset Byte	Offset Bit	
number	INT				
printing	INT				
buffer	INT				
sHMI	STRUCT	NJ	Offset Byte	Offset Bit	
Button	sHMI_Buttons				
Lamp	sHMI_Lamps				
sHMI_Buttons	STRUCT	NJ	Offset Byte	Offset Bit	
Auto	BOOL				
Start	BOOL				
Pause	BOOL				
Resume	BOOL				
Stop	BOOL				
Home	BOOL				
Connect	BOOL				
sHMI_Lamps	STRUCT	NJ	Offset Byte	Offset Bit	
ModbusOK	BOOL				

APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC Data Types

Connected	BOOL				
RobotReady	BOOL				

APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC Global Variables

1-7-2.Global Variables

Name	Data Type	Initial Value	AT	Retain	Constant	Network Publish	Comment
VAR GLOBAL							
HMI Comm	sHMI			True	False	Do not publish	
masterState	INT	0		False	False	Do not publish	Used to keep track of current production state
firstBackgroudRunDone	BOOL			False	False	Do not publish	Goes true when initial equipment status information has been obtained
customerOrder	ST_OrderDetails			False	False	Publish Only	Structure that holds order info coming from Movicon
csvBools	ST_CsvTools			False	False	Do not publish	Toolbox for .CSV-related commands
cutterDxfInfo	ST_CutterDxfExport			False	False	Do not publish	Structure for storing .DXF cutting information
cutterProductInfo	ARRAY[1..2] OF ST_CutterDxfExport	[(separator := 'sep=', finalFileName := 'SR\$TestPlate.dxf', templateName := 'NTNU_Plate_Template_v1.dxf', logoFileName := 'logo_ntnu_manulab.png', xInsertPoint Logo := '1', yInsertPoint Logo := '60', xPixelSize := '1157', yPixelSize := '456', xSizeInMm := '98', ySizeInMm := '38.6', rotationLogo := '0', textToInsert := 'Hans Christian H. Finnson', textStyle := 'NTNU-DIN', xInsertPoint Text := '50', yInsertPoint Text := '55', rotationText := '0', alignmentText := '2', textWidth := '78', textHeight := '7', strNameLen := '40'), (sep		False	True	Do not publish	Array holding default .DXF cutting information for each product

APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC Global Variables

		<pre> separator := 'sep=', finalFileName := '\$RSLtestPlate.dxf', templateName := 'NTNU_KeyChain_Template_v2.dxf', logoFileName := 'ntnu_logo_svarf.png', xInsertPointLogo := '1.505', yInsertPointLogo := '18.7732', xPixelSize := '260', yPixelSize := '348', xSizeInMm := '22', ySizeInMm := '29.5', rotationLogo := '0', textToInsert := 'H C F', textStyle := 'NTNU-DIN', xInsertPointText := '12.5', yInsertPointText := '14', rotationText := '0', alignmentText := '2', textWidth := '25', textHeight := '2.5', strNameLen := '8'] </pre>					
cutterStatus	ST_CutterStatus			False	False	Do not publish	Structure containing the status of the cutters
printerCommand	ST_3DPrinterCommand	<pre> (separator := 'sep=', \$RSLIP_Address; Command', IP := 'Argument\$RSL.192.168.250.200', argument := 'print', command := '/api/files/local/test/home.gcode', number := '1') </pre>		False	False	Do not publish	Structure holding text that is written as a command for the printers

APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC Global Variables

printerStatus	ARRAY[0..19] OF ST_3DPrinterStatus	[20((IP := ", connected := 'False', printing := ", ready := ", operational := ", pausing := ", paused := ", finished := ", nozzleTemp := ", bedTemp := ", printJob := ", rack := ", posX := ", posY := "))]		False	False	Do not publish	Array that contains status of all 20 printers
printerActivePrints	ARRAY[0..19] OF ST_3DPrinterCurrentPrintInfo			False	False	Do not publish	Array that contains the ID and number of parts being printed at all the different printers
printerProductInfo	ARRAY[1..2] OF STRING [256]	['P1_Bracket Pair 1pcs.gcode', 'P2_KeychainFrame 1pc.s.gcode']		False	True	Do not publish	Array containing .GCODE filepaths for all product IDs
LDCommands	ST_LDTools			False	False	Do not publish	Toolbox containing commands for LD communication
TMCommands	ST_TMTools			False	False	Output	Toolbox containing commands for TM communication
moviconSendStatus	ST_MoviconStatus			False	False	Publish Only	Structure holding all status information that is transmitted to Movicon
moviconSendCommand	ST_MoviconCommands			False	False	Publish Only	Structure holding commands that are sent to Movicon. Used for starting Python scripts
productStatus	ARRAY[1..2] OF ST_ProductStatus			False	False	Do not publish	Structure holding the current status for number of a products in store, being printed and its buffer
backgroundWritingCommand	BOOL			False	False	Do not publish	Bool used for preventing simultaneous .CSV writing in both Master and Background
LDTakloStatus	ST_LDStatus			False	False	Do not publish	Structure holding status information for the LD-130CT, nicknamed Taklo
LDFinssonStatus	ST_LDStatus			False	False	Do not publish	Structure holding status information for the first LD-90, nicknamed Finsson
LDRingstadStatus	ST_LDStatus			False	False	Do not publish	Structure holding status information for the second LD-90, nicknamed Ringstad
TM_ArmLD	OEN\nRobot\nTMsRobot			False	False	Do not publish	Structure used for communicating with the TMSM located ontop of the LD-130CT, nicknamed Olur
TM_ArmI	OEN\nRobot\nTMsRobot			False	False	Do not publish	Structure used for communicating with the first TM5, nicknamed Dolor

APPENDIX F. SYSMAC MASTER PROGRAM SOURCE CODE

Master Program_5

IPC Global Variables

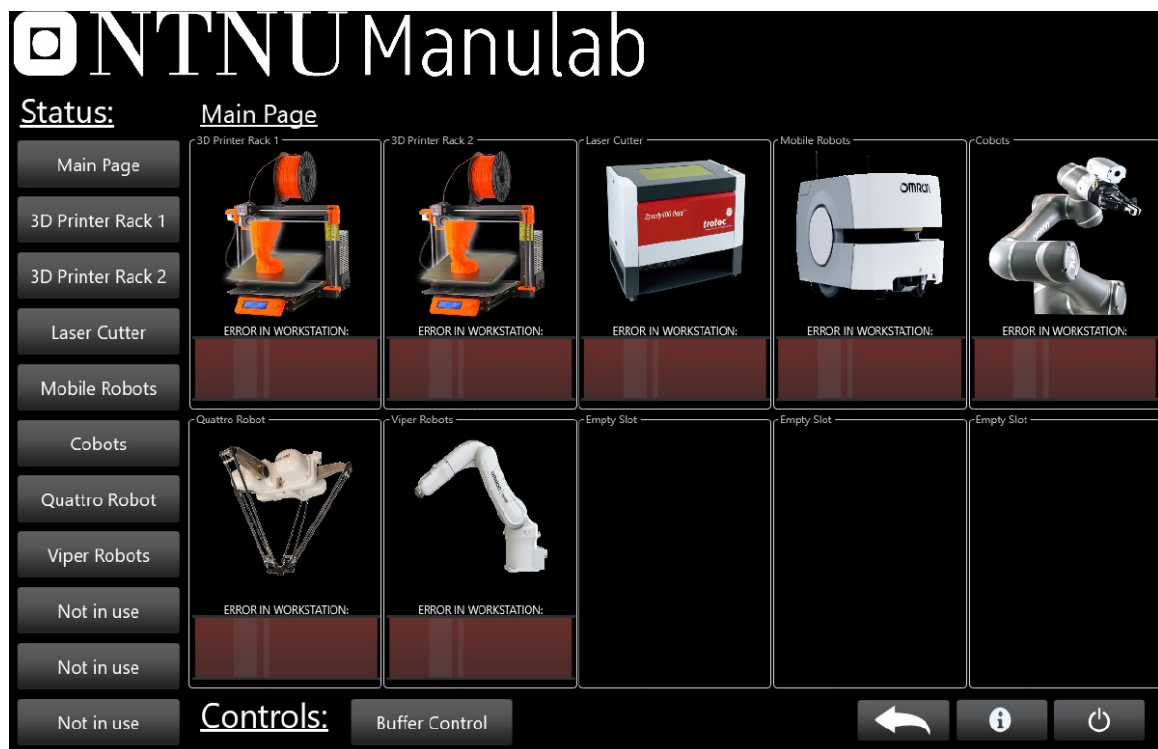
TM_Arm2	OEN\nRobot\nT MsRobot		False	False	Do not publish	Structure used for communicating with the second TM5, nicknamed Doffur
---------	--------------------------	--	-------	-------	----------------	---

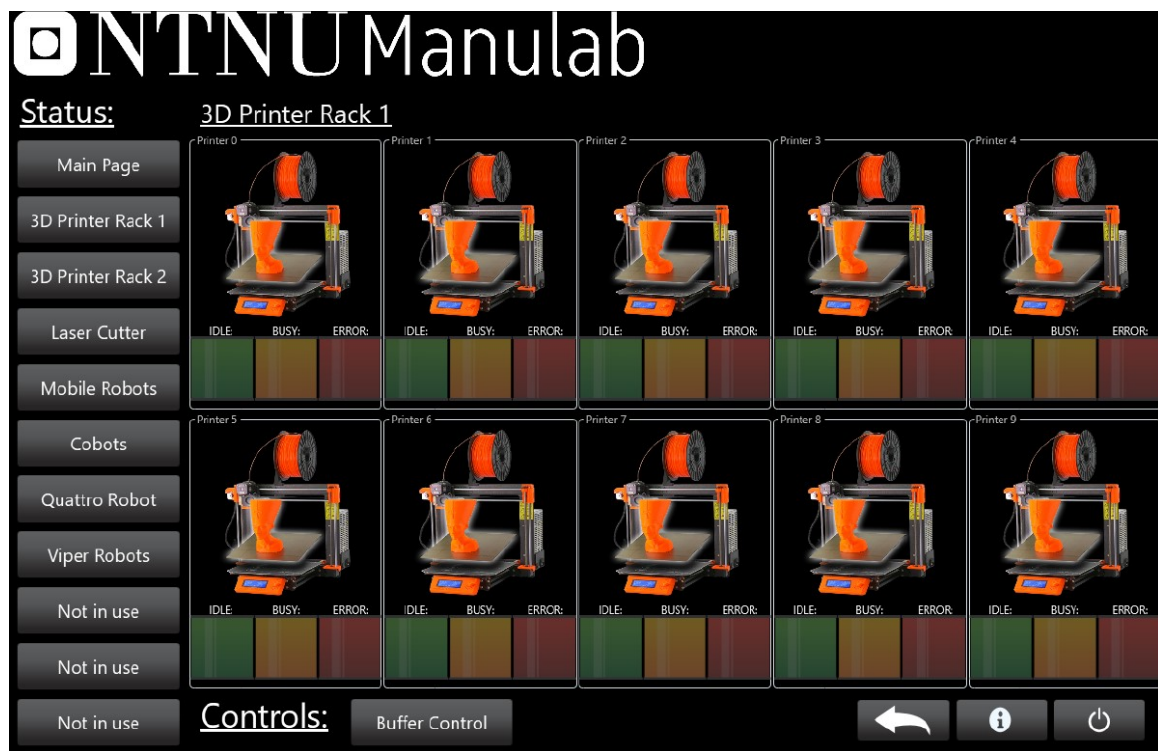
Appendix G

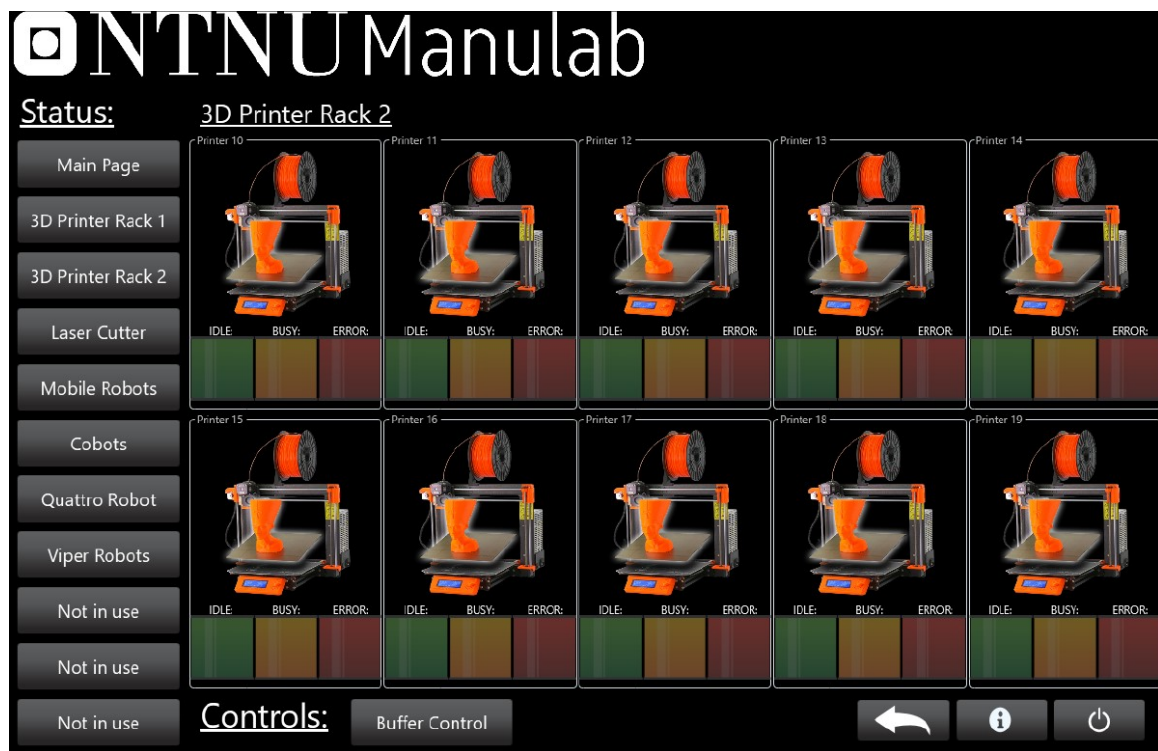
Movicon.NExT HMI Source Code



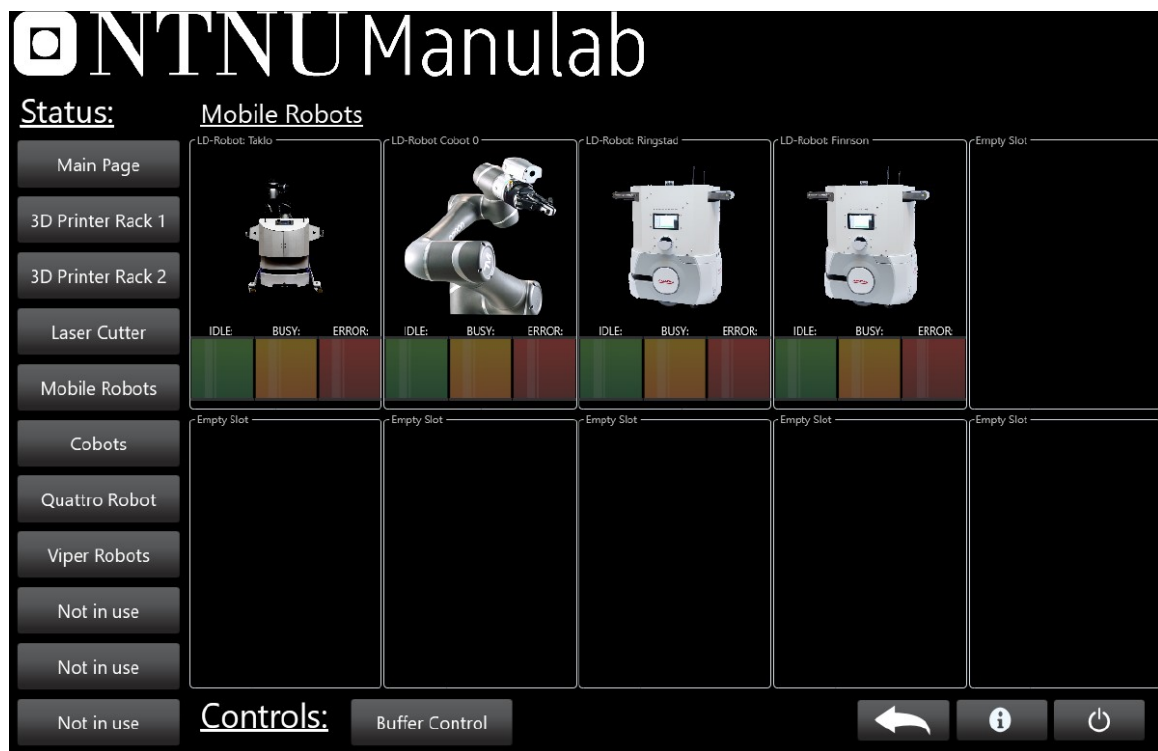


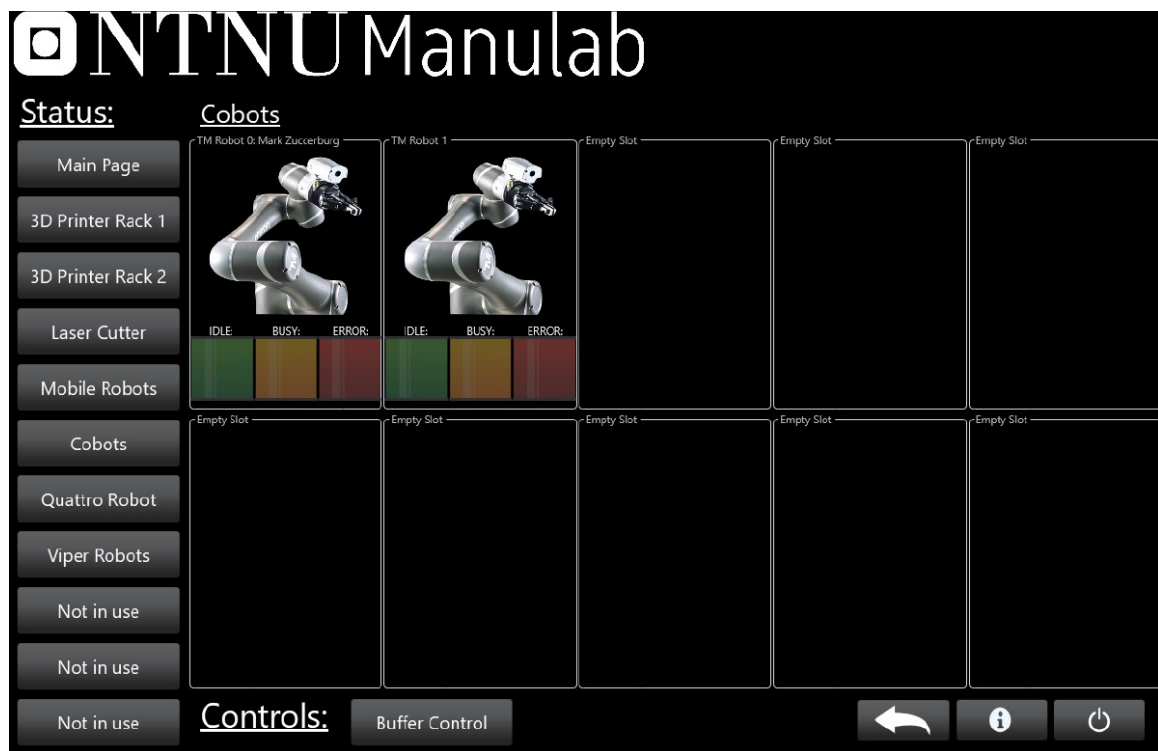


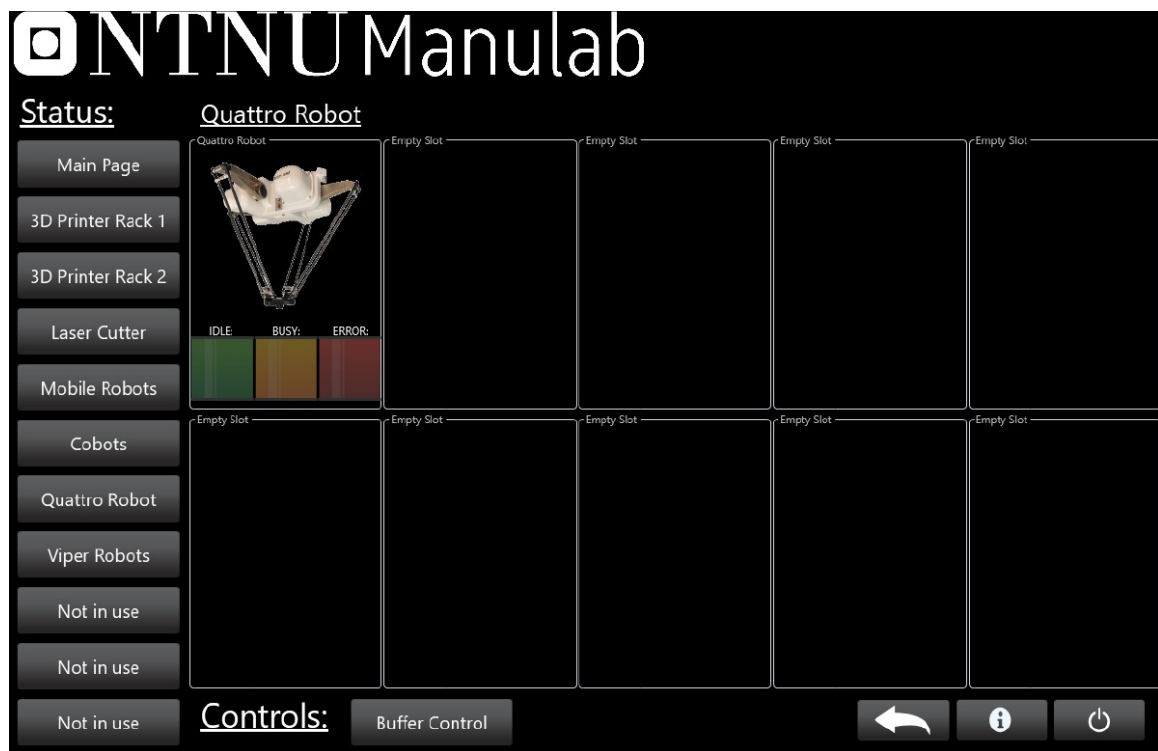


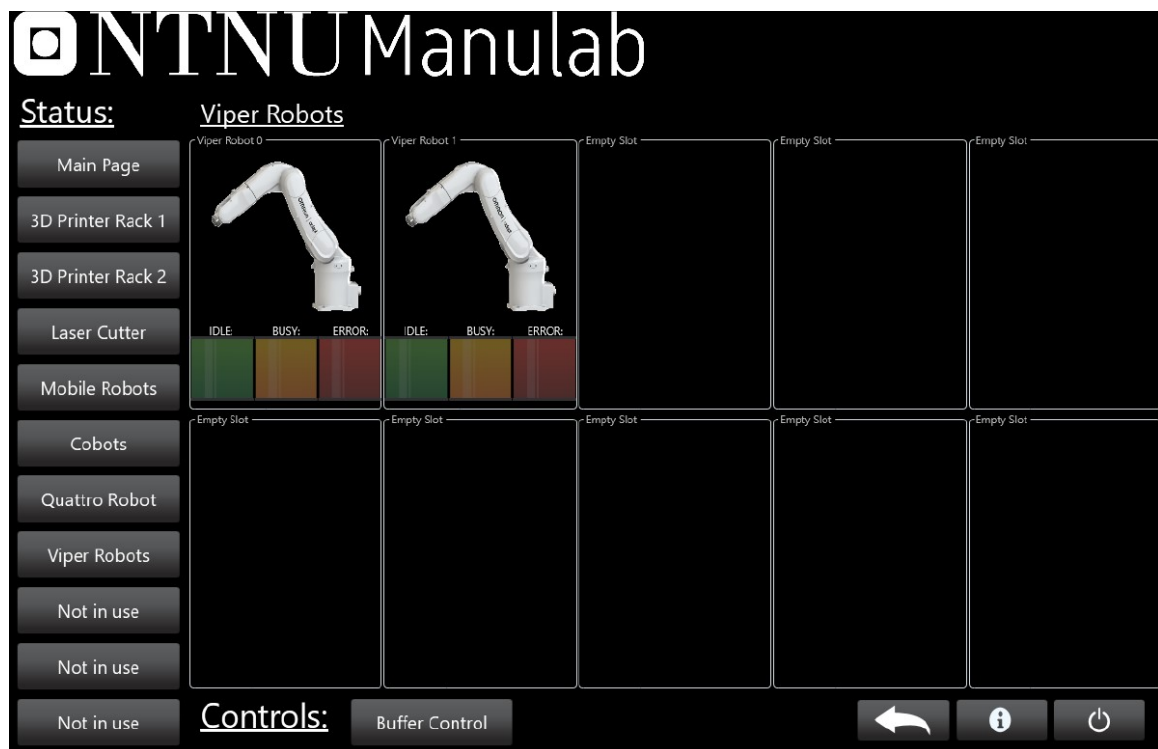


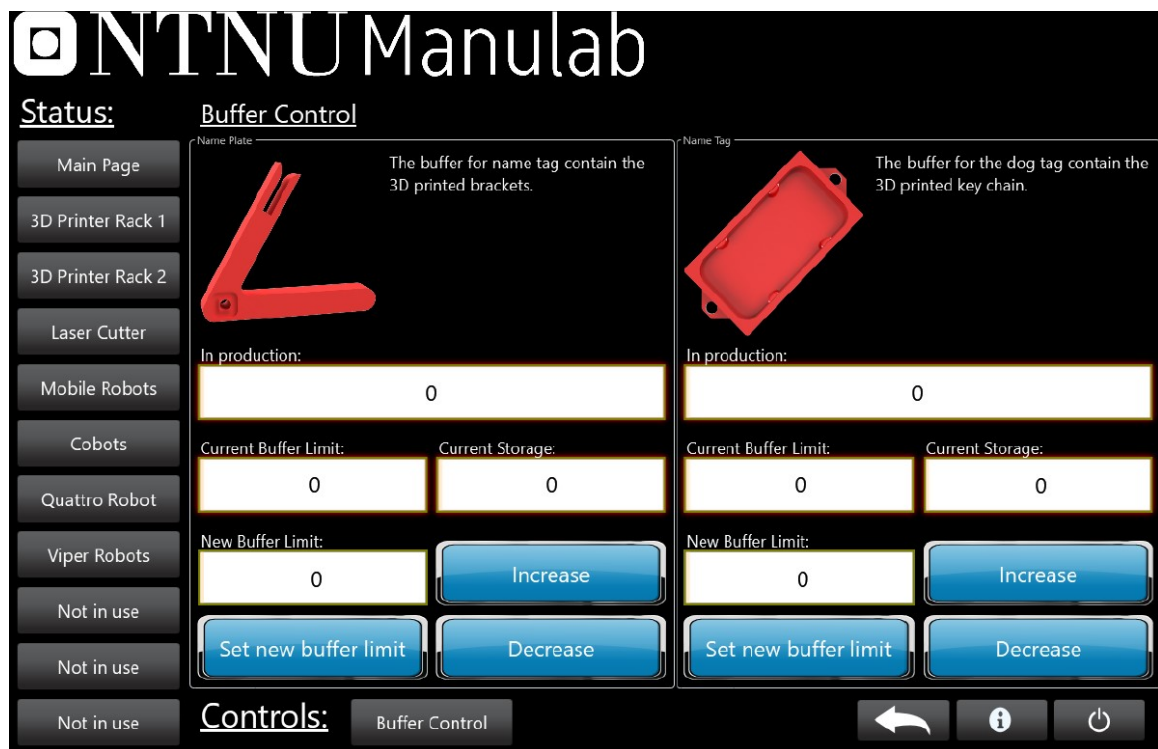


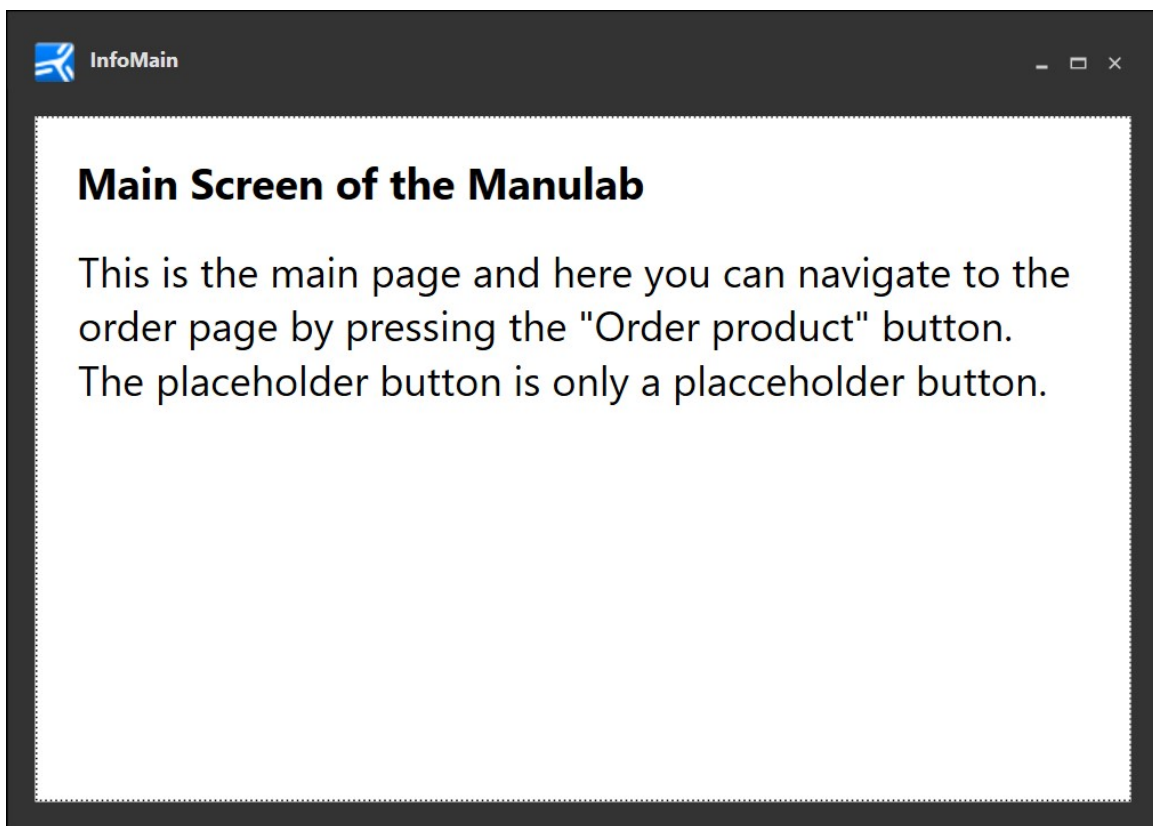


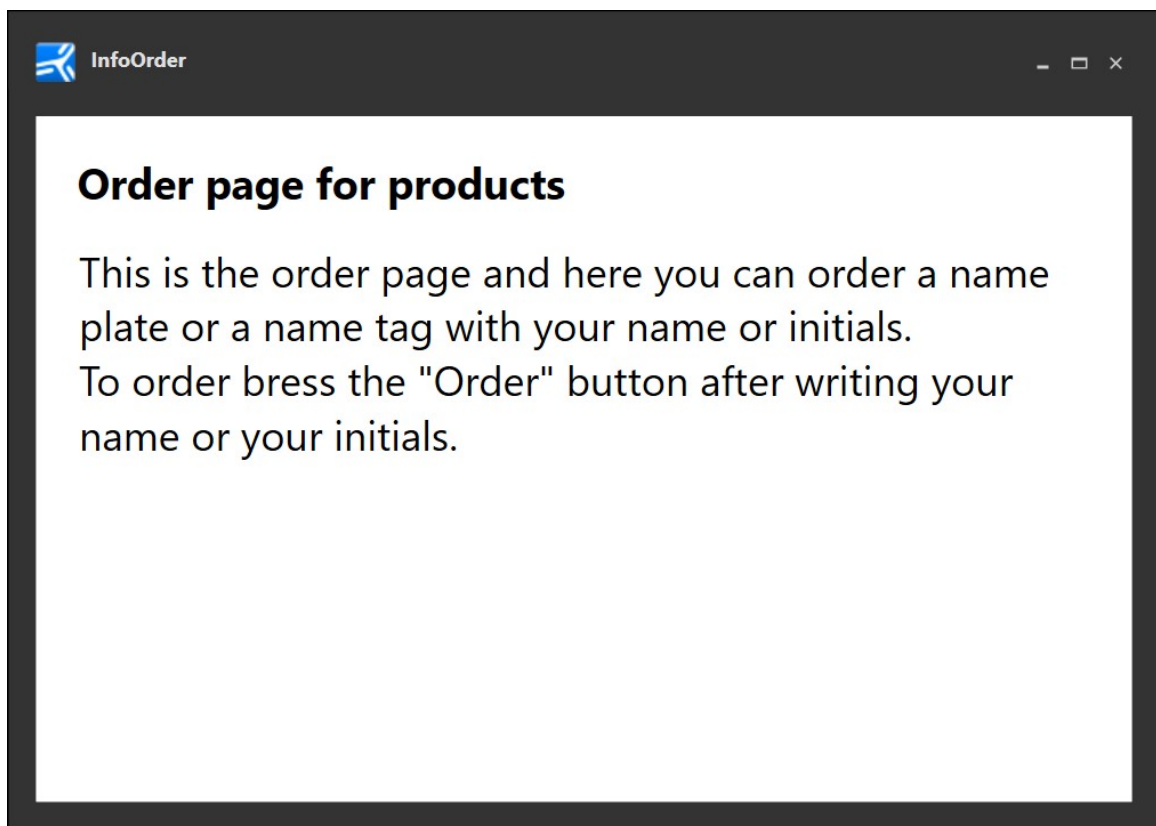


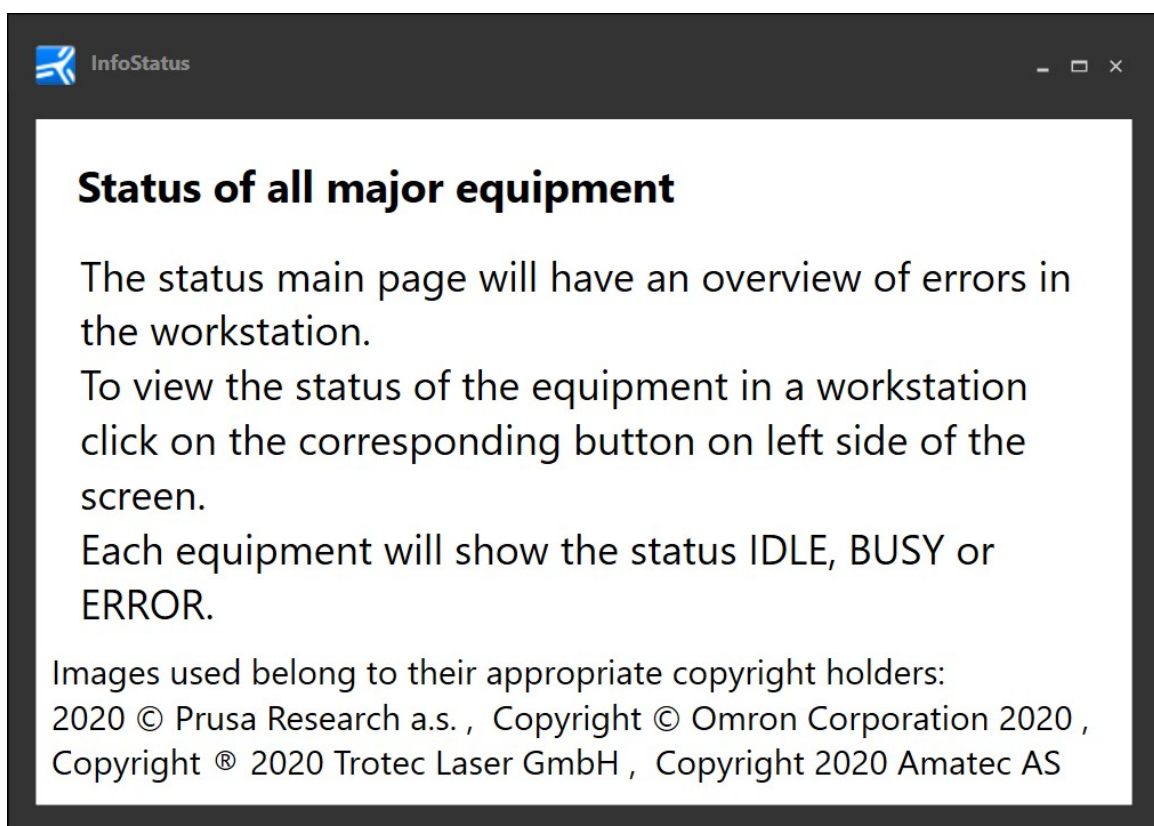


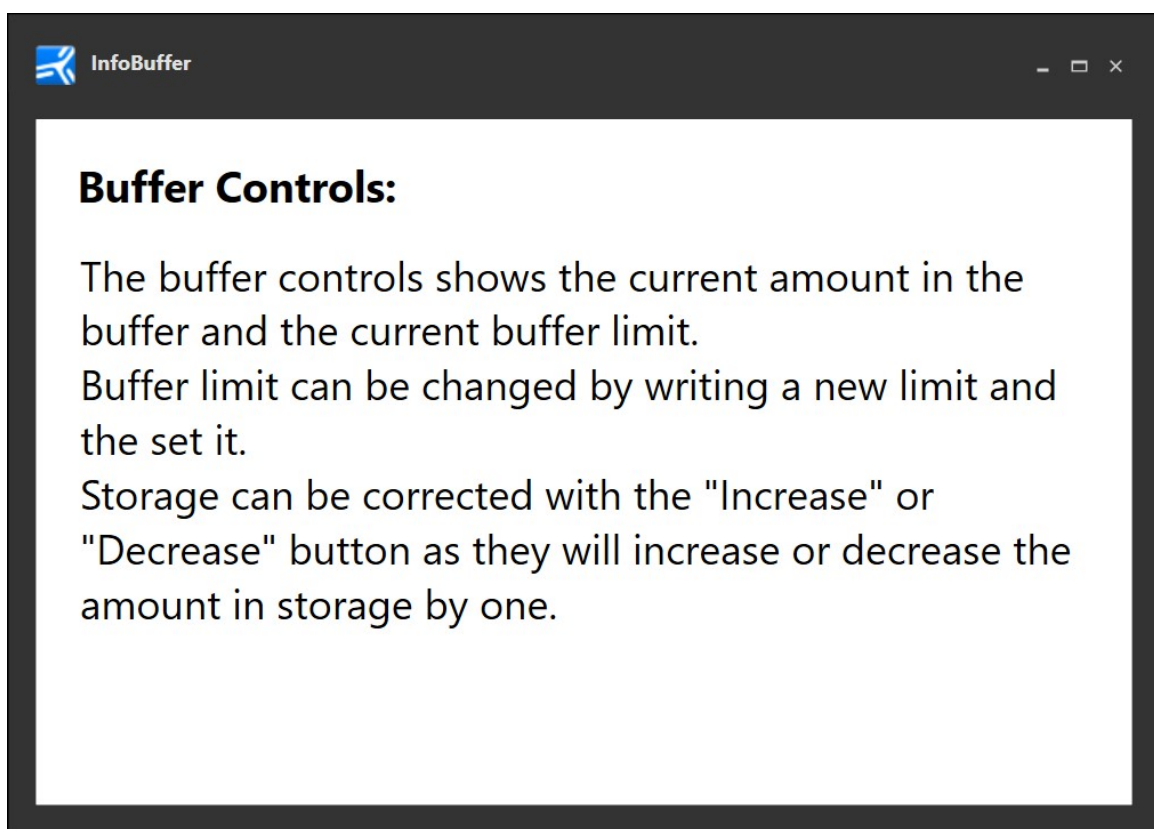












Texts (ManulabHMIWeb)	
ID	en-GB
Buffer_Control	Buffer Control
Cobot_0	TM Robot 0: Mark Zuccherburg
Cobot_1	TM Robot 1
Cobot_Workstation	Cobots
ID_Buffer_Curr_Lim	Current Buffer Limit:
ID_Buffer_New_Lim	New Buffer Limit:
ID_Curr_Storage	Current Storage:
ID_Down	Decrease
ID_In_Prod	In production:
ID_Info_Buffer	The buffer controls shows the current amount in the buffer and the
ID_Info_Buffer_Title	Buffer Controls:
ID_Info_Main	This is the main page and here you can navigate to the order page by
ID_Info_Main_Title	Main Screen of the Manulab
ID_Info_Order	This is the order page and here you can order a name plate or a nam
ID_Info_Order_Title	Order page for products
ID_Info_Status	The status main page will have an overview of errors in the workstat
ID_Info_Status_Copyright	Images used belong to their appropriate copyright holders:\n2020 ©
ID_Info_Status_Title	Status of all major equipment
ID_Not_In_Use	Not in use
ID_Order	Order
ID_Placeholder	Placeholder button
ID_Product_1	A name plate with an NTNU Manulab logo and your name.
ID_Product_1_Buffer	The buffer for name tag contain the 3D printed brackets.
ID_Product_2	A name tag with an NTNU logo and your initial.
ID_Product_2_Buffer	The buffer for the dog tag contain the 3D printed key chain.
ID_Set_Buffer	Set new buffer limit
ID_To_Order_Page	Order product
ID_To_Status_Page	Status of Manulab
ID_Up	Increase
ID_Write_Initial	Write your initials here:
ID_Write_Name	Write your name here:
Laser_Cutter_0	Laser Cutter
Laser_Cutter_Workstation	Laser Cutter
Mobile_Robot_0	LD-Robot: Taklo
Mobile_Robot_1	LD-Robot: Ringstad
Mobile_Robot_2	LD-Robot: Finnson
Mobile_Robot_Cobot_0	LD-Robot Cobot 0
Mobile_Robot_Fleet	Mobile Robots
Printer_00	Printer 0
Printer_01	Printer 1
Printer_02	Printer 2
Printer_03	Printer 3
Printer_04	Printer 4
Printer_05	Printer 5
Printer_06	Printer 6
Printer_07	Printer 7
Printer_08	Printer 8
Printer_09	Printer 9

Printer_10	Printer 10
Printer_11	Printer 11
Printer_12	Printer 12
Printer_13	Printer 13
Printer_14	Printer 14
Printer_15	Printer 15
Printer_16	Printer 16
Printer_17	Printer 17
Printer_18	Printer 18
Printer_19	Printer 19
Printer_Rack_1	3D Printer Rack 1
Printer_Rack_2	3D Printer Rack 2
Product_1	Name Plate
Product_2	Name Tag
Quattro_Robot_0	Quattro Robot
Quattro_Robot_Workstation	Quattro Robot
Status_Btns	Status:
Status_Busy	BUSY:
Status_Empty_Slot	Empty Slot
Status_Error	ERROR:
Status_Error_Main	ERROR IN WORKSTATION:
Status_Idle	IDLE:
Status_Main	Main Page
Viper_Robot_0	Viper Robot 0
Viper_Robot_1	Viper Robot 1
Viper_Robot_Workstation	Viper Robots

current buffer limit.\nBuffer limit can be changed by writing a new limit and the set it.\nStorage can be correc
/ pressing the "Order product" button.\nThe placeholder button is only a placeholder button.
e tag with your name or initials.\nTo order press the "Order" button after writing your name or your initials.
ion.\nTo view the status of the equipment in a workstation click on the corresponding button on left side of t
) Prusa Research a.s. , Copyright © Omron Corporation 2020 , Copyright ® 2020 Trotec Laser GmbH , Copyrig

ted with the "Increase" or "Decrease" button as they will increase or decrease the amount in storage by one.

re screen.\nEach equipment will show the status IDLE, BUSY or ERROR.
ht 2020 Amatec AS

Prototypes (ManulabHMIWeb)
UFUAModel.UFUATagPrototype

Name	Description
customerOrder	
moviconSendCommand	
moviconSendStatus	
status	


```
artnr  
artnr  
buffer  
  
moviconSendStatus Station0
```


EnumStringsFlat	MemberOrderId	UseShared	Name
	-1	True	statusPrinter01
	-1	True	statusPrinter14
	-1	True	statusPrinter17
	-1	True	statusCobot00
	-1	True	statusMobileRobot00
	-1	True	statusMobileRobot01
	-1	True	statusMobileRobot02
	-1	True	statusViper01
	-1	True	statusMobileRobotCobot00
	-1	True	printerRack00
	-1	True	printerRack01
	-1	True	viperWorkstation
	-1	True	namePro01
	-1	True	namePro02
	-1	True	boolTrue
	-1	True	buffLimPro02
	-1	True	moviconSendCommand
	-1	True	masterWorkstation
	-1	True	statusMaster00
	-1	True	customerOrder
	-1	True	statusPrinter00
	-1	True	statusPrinter02
	-1	True	statusPrinter03
	-1	True	statusPrinter04
	-1	True	statusPrinter05
	-1	True	statusPrinter06
	-1	True	statusPrinter07
	-1	True	statusPrinter08
	-1	True	statusPrinter09
	-1	True	statusPrinter10
	-1	True	statusPrinter11
	-1	True	statusPrinter12
	-1	True	statusPrinter13
	-1	True	statusPrinter15
	-1	True	statusPrinter16
	-1	True	statusPrinter18
	-1	True	statusPrinter19
	-1	True	statusCobot01
	-1	True	statusLaserCutter00
	-1	True	statusQuattro00
	-1	True	statusViper00
	-1	True	stopVar
	-1	True	num
	-1	True	laserCutterWorkstation
	-1	True	mobileRobotFleet
	-1	True	cobotWorkstation
	-1	True	quattroWorkstation

-1 True	boolFalse
-1 True	artnrProduct01
-1 True	artnrProduct02
-1 True	buffLimPro01
-1 True	statUpdateStatus
-1 True	moviconSendStatus

Description	ModelType	DataType	ArrayDimension
Status for printer 01	ObjectType		0
Status for printer 14	ObjectType		0
Status for printer 17	ObjectType		0
Status for cobot 00	ObjectType		0
Status for mobile robot 00	ObjectType		0
Status for mobile robot 01	ObjectType		0
Status for mobile robot 02	ObjectType		0
Status for viper robot 01	ObjectType		0
Status for mobile robot cobot 00	ObjectType		0
True if an error has occurred in the rack, false if not	Analog	Boolean	0
True if an error has occurred in the rack, false if not	Analog	Boolean	0
True if an error has occurred in the workstation, false if not	Analog	Boolean	0
Contain the user input name used for product 1 name	Analog	String	0
Contain the user input initial used for product 2 name	Analog	String	0
This boolean is always true	Analog	Boolean	0
Limit for the buffer for product 2	Analog	Int16	0
	ObjectType		0
True if an error has occurred in the workstation, false if not	Analog	Boolean	0
Status for the master 00	ObjectType		0
	ObjectType		0
Status for printer 00	ObjectType		0
Status for printer 02	ObjectType		0
Status for printer 03	ObjectType		0
Status for printer 04	ObjectType		0
Status for printer 05	ObjectType		0
Status for printer 06	ObjectType		0
Status for printer 07	ObjectType		0
Status for printer 08	ObjectType		0
Status for printer 09	ObjectType		0
Status for printer 10	ObjectType		0
Status for printer 11	ObjectType		0
Status for printer 12	ObjectType		0
Status for printer 13	ObjectType		0
Status for printer 15	ObjectType		0
Status for printer 16	ObjectType		0
Status for printer 18	ObjectType		0
Status for printer 19	ObjectType		0
Status for cobot 01	ObjectType		0
Status for laser cutter 00	ObjectType		0
Status for quattro robot 00	ObjectType		0
Status for viper robot 00	ObjectType		0
Variable to stop the UpdateStatus	Analog	Boolean	0
	Analog	Int16	0
True if an error has occurred in the workstation, false if not	Analog	Boolean	0
True if an error has occurred in the fleet, false if not	Analog	Boolean	0
True if an error has occurred in the workstation, false if not	Analog	Boolean	0
True if an error has occurred in the workstation, false if not	Analog	Boolean	0

This boolean is always false	Analog	Boolean	0
Article number for product 1	Analog	Int16	0
Article number for product 2	Analog	Int16	0
Limit for the buffer for product 1	Analog	Int16	0
State the script UpdateStatus is in	Analog	String	0
	ObjectType		0

DynamicSettings

OmronEthernetIP.Station=Station0|LinkType=1|ABA=moviconSendCommand|TEFRM=10

OmronEthernetIP.Station=Station0|LinkType=1|ABA=customerOrder|TEFRM=10|SSFL=:256

OmronEthernetIP.Station=Station0|LinkType=1|ABA=moviconSendStatus|TEFRM=10

False	False	True	False	False
False	False	True	False	False
False	False	True	False	False
False	False	True	False	False
False	False	True	False	False
False	False	True	False	False

False	0 1.00:00:00	False	
False	0 1.00:00:00		1
False	0 1.00:00:00		2
False	0 1.00:00:00		
False	0 1.00:00:00	Not initialized	
False	0 1.00:00:00		

0	0	0 CurrentReadOrWrite
0	0	0 CurrentReadOrWrite
0	0	0 CurrentReadOrWrite
0	0	0 CurrentReadOrWrite
0	0	0 CurrentReadOrWrite
0	0	0 CurrentReadOrWrite

UFUAEngineeringUnit AssignedHistorian AlarmList Views ListProcedures ScriptCode

```
<?xml version="1.0" encoding="utf-8"?>
<ScriptDocument xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://progea.com">
  <bEnableLog>true</bEnableLog>
  <bEnableSysLog>true</bEnableSysLog>
  <breakpoints
xmlns:d2p1="http://schemas.microsoft.com/2003/10/Serialization/Arrays" />
  <currentStatusTag i:nil="true" />
  <cycleTimeTag i:nil="true" />
  <disableWhenNotUsed>true</disableWhenNotUsed>
  <fastSamplingInterval>500</fastSamplingInterval>
  <forceWritingOnServer>true</forceWritingOnServer>
  <id>00000000-0000-0000-0000-000000000000</id>
  <listVariableUsed
xmlns:d2p1="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
    <d2p1:string>statUpdateStatus</d2p1:string>
    <d2p1:string>stopVar</d2p1:string>
    <d2p1:string>num</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-printer00</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-printer01</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-printer02</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-printer03</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-printer04</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-printer05</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-printer06</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-printer07</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-printer08</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-printer09</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-printer10</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-printer11</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-printer12</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-printer13</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-printer14</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-printer15</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-printer16</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-printer17</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-printer18</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-printer19</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-laserCutter00</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-mobileRobot00</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-mobileRobot01</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-mobileRobot02</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-cobot00</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-cobot01</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-cobot02</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-quattro00</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-viper00</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-viper01</d2p1:string>
    <d2p1:string>Station0_moviconSendStatus-masterState</d2p1:string>
    <d2p1:string>statusVar_printers_statusPrinter00-IDLE</d2p1:string>
    <d2p1:string>statusVar_printers_statusPrinter00-BUSY</d2p1:string>
    <d2p1:string>statusVar_printers_statusPrinter00-FAULT</d2p1:string>
    <d2p1:string>statusVar_printers_statusPrinter01-IDLE</d2p1:string>
    <d2p1:string>statusVar_printers_statusPrinter01-BUSY</d2p1:string>
```



```

<d2p1:string>statusVar_printers_statusPrinter19-FAULT</d2p1:string>
<d2p1:string>statusVar_laserCutter_statusLaserCutter00-IDLE</d2p1:string>
<d2p1:string>statusVar_laserCutter_statusLaserCutter00-BUSY</d2p1:string>
<d2p1:string>statusVar_laserCutter_statusLaserCutter00-FAULT</d2p1:string>
<d2p1:string>statusVar_mobileRobot_statusMobileRobot00-IDLE</d2p1:string>
<d2p1:string>statusVar_mobileRobot_statusMobileRobot00-BUSY</d2p1:string>
<d2p1:string>statusVar_mobileRobot_statusMobileRobot00-FAULT</d2p1:string>
<d2p1:string>statusVar_mobileRobot_statusMobileRobot01-IDLE</d2p1:string>
<d2p1:string>statusVar_mobileRobot_statusMobileRobot01-BUSY</d2p1:string>
<d2p1:string>statusVar_mobileRobot_statusMobileRobot01-FAULT</d2p1:string>
<d2p1:string>statusVar_mobileRobot_statusMobileRobot02-IDLE</d2p1:string>
<d2p1:string>statusVar_mobileRobot_statusMobileRobot02-BUSY</d2p1:string>
<d2p1:string>statusVar_mobileRobot_statusMobileRobot02-FAULT</d2p1:string>

<d2p1:string>statusVar_mobileRobot_statusMobileRobotCobot00-IDLE</d2p1:string>

<d2p1:string>statusVar_mobileRobot_statusMobileRobotCobot00-BUSY</d2p1:string>

<d2p1:string>statusVar_mobileRobot_statusMobileRobotCobot00-FAULT</d2p1:string>
<d2p1:string>statusVar_cobot_statusCobot00-IDLE</d2p1:string>
<d2p1:string>statusVar_cobot_statusCobot00-BUSY</d2p1:string>
<d2p1:string>statusVar_cobot_statusCobot00-FAULT</d2p1:string>
<d2p1:string>statusVar_cobot_statusCobot01-IDLE</d2p1:string>
<d2p1:string>statusVar_cobot_statusCobot01-BUSY</d2p1:string>
<d2p1:string>statusVar_cobot_statusCobot01-FAULT</d2p1:string>
<d2p1:string>statusVar_quattro_statusQuattro00-IDLE</d2p1:string>
<d2p1:string>statusVar_quattro_statusQuattro00-BUSY</d2p1:string>
<d2p1:string>statusVar_quattro_statusQuattro00-FAULT</d2p1:string>
<d2p1:string>statusVar_viper_statusViper00-IDLE</d2p1:string>
<d2p1:string>statusVar_viper_statusViper00-BUSY</d2p1:string>
<d2p1:string>statusVar_viper_statusViper00-FAULT</d2p1:string>
<d2p1:string>statusVar_viper_statusViper01-IDLE</d2p1:string>
<d2p1:string>statusVar_viper_statusViper01-BUSY</d2p1:string>
<d2p1:string>statusVar_viper_statusViper01-FAULT</d2p1:string>
<d2p1:string>statusVar_master_statusMaster00-IDLE</d2p1:string>
<d2p1:string>statusVar_master_statusMaster00-BUSY</d2p1:string>
<d2p1:string>statusVar_master_statusMaster00-FAULT</d2p1:string>
<d2p1:string>statusVar_workstation_printerRack00</d2p1:string>
<d2p1:string>statusVar_workstation_printerRack01</d2p1:string>
<d2p1:string>statusVar_workstation_laserCutterWorkstation</d2p1:string>
<d2p1:string>statusVar_workstation_mobileRobotFleet</d2p1:string>
<d2p1:string>statusVar_workstation_cobotWorkstation</d2p1:string>
<d2p1:string>statusVar_workstation_quattroWorkstation</d2p1:string>
<d2p1:string>statusVar_workstation_viperWorkstation</d2p1:string>
<d2p1:string>statusVar_workstation_masterWorkstation</d2p1:string>
</listVariableUsed>
<maxCleanCount>2</maxCleanCount>
<nSellLength>0</nSellLength>
<nSleepTime>50</nSleepTime>
<nStartSel>3268</nStartSel>
<publishingInterval>1000</publishingInterval>
<removeDisabledItemAfterSecs>30</removeDisabledItemAfterSecs>
<sCode>'#Language "WWB.NET"
Imports System

```



```
Imports System.Threading

Option Explicit

Const statCodeFault As Int16 = 0
Const statCodeIdle As Int16 = 1
Const statCodeBusy As Int16 = 2

Const sleepTime As Int16 = 2000

Sub Main Handles .Main
    statUpdateStatus = "Initilasing"
    stopVar = False
    num = 0
    Do
        ' DoEvents
        '
        'num = num + 1
        '
        ' Set all status external codes to a local variables
        '
        statUpdateStatus = "Get status"
        '
        ' 3D Printers:
        Dim currStatCodePrinter00 As Int16 =
Station0_moviconSendStatus.printer00
        Dim currStatCodePrinter01 As Int16 =
Station0_moviconSendStatus.printer01
        Dim currStatCodePrinter02 As Int16 =
Station0_moviconSendStatus.printer02
        Dim currStatCodePrinter03 As Int16 =
Station0_moviconSendStatus.printer03
        Dim currStatCodePrinter04 As Int16 =
Station0_moviconSendStatus.printer04
        Dim currStatCodePrinter05 As Int16 =
Station0_moviconSendStatus.printer05
        Dim currStatCodePrinter06 As Int16 =
Station0_moviconSendStatus.printer06
        Dim currStatCodePrinter07 As Int16 =
Station0_moviconSendStatus.printer07
        Dim currStatCodePrinter08 As Int16 =
Station0_moviconSendStatus.printer08
        Dim currStatCodePrinter09 As Int16 =
Station0_moviconSendStatus.printer09
        Dim currStatCodePrinter10 As Int16 =
Station0_moviconSendStatus.printer10
        Dim currStatCodePrinter11 As Int16 =
Station0_moviconSendStatus.printer11
        Dim currStatCodePrinter12 As Int16 =
Station0_moviconSendStatus.printer12
        Dim currStatCodePrinter13 As Int16 =
Station0_moviconSendStatus.printer13
        Dim currStatCodePrinter14 As Int16 =
Station0_moviconSendStatus.printer14
```

```
        Dim currStatCodePrinter15 As Int16 =
Station0_moviconSendStatus.printer15
        Dim currStatCodePrinter16 As Int16 =
Station0_moviconSendStatus.printer16
        Dim currStatCodePrinter17 As Int16 =
Station0_moviconSendStatus.printer17
        Dim currStatCodePrinter18 As Int16 =
Station0_moviconSendStatus.printer18
        Dim currStatCodePrinter19 As Int16 =
Station0_moviconSendStatus.printer19
    ,
    ' Laser cutter:
    Dim currStatCodeLaserCutter00 As Int16 =
Station0_moviconSendStatus.laserCutter00
    ,
    ' Mobile Robots:
    Dim currStatCodeMobileRobot00 As Int16 =
Station0_moviconSendStatus.mobileRobot00
    Dim currStatCodeMobileRobot01 As Int16 =
Station0_moviconSendStatus.mobileRobot01
    Dim currStatCodeMobileRobot02 As Int16 =
Station0_moviconSendStatus.mobileRobot02
    ,
    ' Cobots:
    Dim currStatCodeCobot00 As Int16 = Station0_moviconSendStatus.cobot00
    Dim currStatCodeCobot01 As Int16 = Station0_moviconSendStatus.cobot01
    Dim currStatCodeCobot02 As Int16 = Station0_moviconSendStatus.cobot02
    ,
    ' Quattro:
    Dim currStatCodeQuattro00 As Int16 =
Station0_moviconSendStatus.quattro00
    ,
    ' Viper:
    Dim currStatCodeViper00 As Int16 = Station0_moviconSendStatus.viper00
    Dim currStatCodeViper01 As Int16 = Station0_moviconSendStatus.viper01
    ,
    ' Master:
    Dim currStatCodeMaster00 As Int16 =
Station0_moviconSendStatus.masterState
    ,
    ' Set status according to status code
    ,
    statUpdateStatus = "Set indicators"
    ,
    Dim workstationErr00 As Boolean = False
    Dim workstationErr01 As Boolean = False
    Dim workstationErr02 As Boolean = False
    Dim workstationErr03 As Boolean = False
    Dim workstationErr04 As Boolean = False
    Dim workstationErr05 As Boolean = False
    Dim workstationErr06 As Boolean = False
    Dim workstationErr07 As Boolean = False
    ,
    ' Status for 3D Printers
```

```
,
' Status for 3D Printer 00
Select Case currStatCodePrinter01
  Case statCodeIdle
    statusVar_printers_statusPrinter00.IDLE = True
    statusVar_printers_statusPrinter00.BUSY = False
    statusVar_printers_statusPrinter00.FAULT = False
  Case statCodeBusy
    statusVar_printers_statusPrinter00.IDLE = False
    statusVar_printers_statusPrinter00.BUSY = True
    statusVar_printers_statusPrinter00.FAULT = False
  Case statCodeFault
    statusVar_printers_statusPrinter00.IDLE = False
    statusVar_printers_statusPrinter00.BUSY = False
    statusVar_printers_statusPrinter00.FAULT = True
    workstationErr00 = True
  Case Else
    statusVar_printers_statusPrinter00.IDLE = False
    statusVar_printers_statusPrinter00.BUSY = False
    statusVar_printers_statusPrinter00.FAULT = True
    workstationErr00 = True
End Select

' Status for 3D Printer 01
Select Case currStatCodePrinter01
  Case statCodeIdle
    statusVar_printers_statusPrinter01.IDLE = True
    statusVar_printers_statusPrinter01.BUSY = False
    statusVar_printers_statusPrinter01.FAULT = False
  Case statCodeBusy
    statusVar_printers_statusPrinter01.IDLE = False
    statusVar_printers_statusPrinter01.BUSY = True
    statusVar_printers_statusPrinter01.FAULT = False
  Case statCodeFault
    statusVar_printers_statusPrinter01.IDLE = False
    statusVar_printers_statusPrinter01.BUSY = False
    statusVar_printers_statusPrinter01.FAULT = True
    workstationErr00 = True
  Case Else
    statusVar_printers_statusPrinter01.IDLE = False
    statusVar_printers_statusPrinter01.BUSY = False
    statusVar_printers_statusPrinter01.FAULT = True
    workstationErr00 = True
End Select

' Status for 3D Printer 02
Select Case currStatCodePrinter02
  Case statCodeIdle
    statusVar_printers_statusPrinter02.IDLE = True
    statusVar_printers_statusPrinter02.BUSY = False
    statusVar_printers_statusPrinter02.FAULT = False
  Case statCodeBusy
    statusVar_printers_statusPrinter02.IDLE = False
    statusVar_printers_statusPrinter02.BUSY = True
```

```

        statusVar_printers_statusPrinter02.FAULT = False
    Case statCodeFault
        statusVar_printers_statusPrinter02.IDLE = False
        statusVar_printers_statusPrinter02.BUSY = False
        statusVar_printers_statusPrinter02.FAULT = True
        workstationErr00 = True
    Case Else
        statusVar_printers_statusPrinter02.IDLE = False
        statusVar_printers_statusPrinter02.BUSY = False
        statusVar_printers_statusPrinter02.FAULT = True
        workstationErr00 = True
End Select

' Status for 3D Printer 03
Select Case currStatCodePrinter03
    Case statCodeIdle
        statusVar_printers_statusPrinter03.IDLE = True
        statusVar_printers_statusPrinter03.BUSY = False
        statusVar_printers_statusPrinter03.FAULT = False
    Case statCodeBusy
        statusVar_printers_statusPrinter03.IDLE = False
        statusVar_printers_statusPrinter03.BUSY = True
        statusVar_printers_statusPrinter03.FAULT = False
    Case statCodeFault
        statusVar_printers_statusPrinter03.IDLE = False
        statusVar_printers_statusPrinter03.BUSY = False
        statusVar_printers_statusPrinter03.FAULT = True
        workstationErr00 = True
    Case Else
        statusVar_printers_statusPrinter03.IDLE = False
        statusVar_printers_statusPrinter03.BUSY = False
        statusVar_printers_statusPrinter03.FAULT = True
        workstationErr00 = True
End Select

' Status for 3D Printer 04
Select Case currStatCodePrinter04
    Case statCodeIdle
        statusVar_printers_statusPrinter04.IDLE = True
        statusVar_printers_statusPrinter04.BUSY = False
        statusVar_printers_statusPrinter04.FAULT = False
    Case statCodeBusy
        statusVar_printers_statusPrinter04.IDLE = False
        statusVar_printers_statusPrinter04.BUSY = True
        statusVar_printers_statusPrinter04.FAULT = False
    Case statCodeFault
        statusVar_printers_statusPrinter04.IDLE = False
        statusVar_printers_statusPrinter04.BUSY = False
        statusVar_printers_statusPrinter04.FAULT = True
        workstationErr00 = True
    Case Else
        statusVar_printers_statusPrinter04.IDLE = False
        statusVar_printers_statusPrinter04.BUSY = False
        statusVar_printers_statusPrinter04.FAULT = True

```

```
        workstationErr00 = True
    End Select

    ' Status for 3D Printer 05
    Select Case currStatCodePrinter05
        Case statCodeIdle
            statusVar_printers_statusPrinter05.IDLE = True
            statusVar_printers_statusPrinter05.BUSY = False
            statusVar_printers_statusPrinter05.FAULT = False
        Case statCodeBusy
            statusVar_printers_statusPrinter05.IDLE = False
            statusVar_printers_statusPrinter05.BUSY = True
            statusVar_printers_statusPrinter05.FAULT = False
        Case statCodeFault
            statusVar_printers_statusPrinter05.IDLE = False
            statusVar_printers_statusPrinter05.BUSY = False
            statusVar_printers_statusPrinter05.FAULT = True
            workstationErr00 = True
        Case Else
            statusVar_printers_statusPrinter05.IDLE = False
            statusVar_printers_statusPrinter05.BUSY = False
            statusVar_printers_statusPrinter05.FAULT = True
            workstationErr00 = True
    End Select

    ' Status for 3D Printer 06
    Select Case currStatCodePrinter06
        Case statCodeIdle
            statusVar_printers_statusPrinter06.IDLE = True
            statusVar_printers_statusPrinter06.BUSY = False
            statusVar_printers_statusPrinter06.FAULT = False
        Case statCodeBusy
            statusVar_printers_statusPrinter06.IDLE = False
            statusVar_printers_statusPrinter06.BUSY = True
            statusVar_printers_statusPrinter06.FAULT = False
        Case statCodeFault
            statusVar_printers_statusPrinter06.IDLE = False
            statusVar_printers_statusPrinter06.BUSY = False
            statusVar_printers_statusPrinter06.FAULT = True
            workstationErr00 = True
        Case Else
            statusVar_printers_statusPrinter06.IDLE = False
            statusVar_printers_statusPrinter06.BUSY = False
            statusVar_printers_statusPrinter06.FAULT = True
            workstationErr00 = True
    End Select

    ' Status for 3D Printer 07
    Select Case currStatCodePrinter07
        Case statCodeIdle
            statusVar_printers_statusPrinter07.IDLE = True
            statusVar_printers_statusPrinter07.BUSY = False
            statusVar_printers_statusPrinter07.FAULT = False
        Case statCodeBusy
```

```
        statusVar_printers_statusPrinter07.IDLE = False
        statusVar_printers_statusPrinter07.BUSY = True
        statusVar_printers_statusPrinter07.FAULT = False
    Case statCodeFault
        statusVar_printers_statusPrinter07.IDLE = False
        statusVar_printers_statusPrinter07.BUSY = False
        statusVar_printers_statusPrinter07.FAULT = True
        workstationErr00 = True
    Case Else
        statusVar_printers_statusPrinter07.IDLE = False
        statusVar_printers_statusPrinter07.BUSY = False
        statusVar_printers_statusPrinter07.FAULT = True
        workstationErr00 = True
End Select

' Status for 3D Printer 08
Select Case currStatCodePrinter08
    Case statCodeIdle
        statusVar_printers_statusPrinter08.IDLE = True
        statusVar_printers_statusPrinter08.BUSY = False
        statusVar_printers_statusPrinter08.FAULT = False
    Case statCodeBusy
        statusVar_printers_statusPrinter08.IDLE = False
        statusVar_printers_statusPrinter08.BUSY = True
        statusVar_printers_statusPrinter08.FAULT = False
    Case statCodeFault
        statusVar_printers_statusPrinter08.IDLE = False
        statusVar_printers_statusPrinter08.BUSY = False
        statusVar_printers_statusPrinter08.FAULT = True
        workstationErr00 = True
    Case Else
        statusVar_printers_statusPrinter08.IDLE = False
        statusVar_printers_statusPrinter08.BUSY = False
        statusVar_printers_statusPrinter08.FAULT = True
        workstationErr00 = True
End Select

' Status for 3D Printer 09
Select Case currStatCodePrinter09
    Case statCodeIdle
        statusVar_printers_statusPrinter09.IDLE = True
        statusVar_printers_statusPrinter09.BUSY = False
        statusVar_printers_statusPrinter09.FAULT = False
    Case statCodeBusy
        statusVar_printers_statusPrinter09.IDLE = False
        statusVar_printers_statusPrinter09.BUSY = True
        statusVar_printers_statusPrinter09.FAULT = False
    Case statCodeFault
        statusVar_printers_statusPrinter09.IDLE = False
        statusVar_printers_statusPrinter09.BUSY = False
        statusVar_printers_statusPrinter09.FAULT = True
        workstationErr00 = True
    Case Else
        statusVar_printers_statusPrinter09.IDLE = False
```

```
        statusVar_printers_statusPrinter09.BUSY = False
        statusVar_printers_statusPrinter09.FAULT = True
        workstationErr00 = True
    End Select

' Status for 3D Printer 10
Select Case currStatCodePrinter10
    Case statCodeIdle
        statusVar_printers_statusPrinter10.IDLE = True
        statusVar_printers_statusPrinter10.BUSY = False
        statusVar_printers_statusPrinter10.FAULT = False
    Case statCodeBusy
        statusVar_printers_statusPrinter10.IDLE = False
        statusVar_printers_statusPrinter10.BUSY = True
        statusVar_printers_statusPrinter10.FAULT = False
    Case statCodeFault
        statusVar_printers_statusPrinter10.IDLE = False
        statusVar_printers_statusPrinter10.BUSY = False
        statusVar_printers_statusPrinter10.FAULT = True
        workstationErr01 = True
    Case Else
        statusVar_printers_statusPrinter10.IDLE = False
        statusVar_printers_statusPrinter10.BUSY = False
        statusVar_printers_statusPrinter10.FAULT = True
        workstationErr01 = True
End Select

' Status for 3D Printer 11
Select Case currStatCodePrinter11
    Case statCodeIdle
        statusVar_printers_statusPrinter11.IDLE = True
        statusVar_printers_statusPrinter11.BUSY = False
        statusVar_printers_statusPrinter11.FAULT = False
    Case statCodeBusy
        statusVar_printers_statusPrinter11.IDLE = False
        statusVar_printers_statusPrinter11.BUSY = True
        statusVar_printers_statusPrinter11.FAULT = False
    Case statCodeFault
        statusVar_printers_statusPrinter11.IDLE = False
        statusVar_printers_statusPrinter11.BUSY = False
        statusVar_printers_statusPrinter11.FAULT = True
        workstationErr01 = True
    Case Else
        statusVar_printers_statusPrinter11.IDLE = False
        statusVar_printers_statusPrinter11.BUSY = False
        statusVar_printers_statusPrinter11.FAULT = True
        workstationErr01 = True
End Select

' Status for 3D Printer 12
Select Case currStatCodePrinter12
    Case statCodeIdle
        statusVar_printers_statusPrinter12.IDLE = True
        statusVar_printers_statusPrinter12.BUSY = False
```

```
        statusVar_printers_statusPrinter12.FAULT = False
    Case statCodeBusy
        statusVar_printers_statusPrinter12.IDLE = False
        statusVar_printers_statusPrinter12.BUSY = True
        statusVar_printers_statusPrinter12.FAULT = False
    Case statCodeFault
        statusVar_printers_statusPrinter12.IDLE = False
        statusVar_printers_statusPrinter12.BUSY = False
        statusVar_printers_statusPrinter12.FAULT = True
        workstationErr01 = True
    Case Else
        statusVar_printers_statusPrinter12.IDLE = False
        statusVar_printers_statusPrinter12.BUSY = False
        statusVar_printers_statusPrinter12.FAULT = True
        workstationErr01 = True
End Select

' Status for 3D Printer 13
Select Case currStatCodePrinter13
    Case statCodeIdle
        statusVar_printers_statusPrinter13.IDLE = True
        statusVar_printers_statusPrinter13.BUSY = False
        statusVar_printers_statusPrinter13.FAULT = False
    Case statCodeBusy
        statusVar_printers_statusPrinter13.IDLE = False
        statusVar_printers_statusPrinter13.BUSY = True
        statusVar_printers_statusPrinter13.FAULT = False
    Case statCodeFault
        statusVar_printers_statusPrinter13.IDLE = False
        statusVar_printers_statusPrinter13.BUSY = False
        statusVar_printers_statusPrinter13.FAULT = True
        workstationErr01 = True
    Case Else
        statusVar_printers_statusPrinter13.IDLE = False
        statusVar_printers_statusPrinter13.BUSY = False
        statusVar_printers_statusPrinter13.FAULT = True
        workstationErr01 = True
End Select

' Status for 3D Printer 14
Select Case currStatCodePrinter14
    Case statCodeIdle
        statusVar_printers_statusPrinter14.IDLE = True
        statusVar_printers_statusPrinter14.BUSY = False
        statusVar_printers_statusPrinter14.FAULT = False
    Case statCodeBusy
        statusVar_printers_statusPrinter14.IDLE = False
        statusVar_printers_statusPrinter14.BUSY = True
        statusVar_printers_statusPrinter14.FAULT = False
    Case statCodeFault
        statusVar_printers_statusPrinter14.IDLE = False
        statusVar_printers_statusPrinter14.BUSY = False
        statusVar_printers_statusPrinter14.FAULT = True
        workstationErr01 = True
```



```
        Case Else
            statusVar_printers_statusPrinter14.IDLE = False
            statusVar_printers_statusPrinter14.BUSY = False
            statusVar_printers_statusPrinter14.FAULT = True
            workstationErr01 = True
    End Select

' Status for 3D Printer 15
Select Case currStatCodePrinter15
    Case statCodeIdle
        statusVar_printers_statusPrinter15.IDLE = True
        statusVar_printers_statusPrinter15.BUSY = False
        statusVar_printers_statusPrinter15.FAULT = False
    Case statCodeBusy
        statusVar_printers_statusPrinter15.IDLE = False
        statusVar_printers_statusPrinter15.BUSY = True
        statusVar_printers_statusPrinter15.FAULT = False
    Case statCodeFault
        statusVar_printers_statusPrinter15.IDLE = False
        statusVar_printers_statusPrinter15.BUSY = False
        statusVar_printers_statusPrinter15.FAULT = True
        workstationErr01 = True
    Case Else
        statusVar_printers_statusPrinter15.IDLE = False
        statusVar_printers_statusPrinter15.BUSY = False
        statusVar_printers_statusPrinter15.FAULT = True
        workstationErr01 = True
End Select

' Status for 3D Printer 16
Select Case currStatCodePrinter16
    Case statCodeIdle
        statusVar_printers_statusPrinter16.IDLE = True
        statusVar_printers_statusPrinter16.BUSY = False
        statusVar_printers_statusPrinter16.FAULT = False
    Case statCodeBusy
        statusVar_printers_statusPrinter16.IDLE = False
        statusVar_printers_statusPrinter16.BUSY = True
        statusVar_printers_statusPrinter16.FAULT = False
    Case statCodeFault
        statusVar_printers_statusPrinter16.IDLE = False
        statusVar_printers_statusPrinter16.BUSY = False
        statusVar_printers_statusPrinter16.FAULT = True
        workstationErr01 = True
    Case Else
        statusVar_printers_statusPrinter16.IDLE = False
        statusVar_printers_statusPrinter16.BUSY = False
        statusVar_printers_statusPrinter16.FAULT = True
        workstationErr01 = True
End Select

' Status for 3D Printer 17
Select Case currStatCodePrinter17
    Case statCodeIdle
```

```

        statusVar_printers_statusPrinter17.IDLE = True
        statusVar_printers_statusPrinter17.BUSY = False
        statusVar_printers_statusPrinter17.FAULT = False
    Case statCodeBusy
        statusVar_printers_statusPrinter17.IDLE = False
        statusVar_printers_statusPrinter17.BUSY = True
        statusVar_printers_statusPrinter17.FAULT = False
    Case statCodeFault
        statusVar_printers_statusPrinter17.IDLE = False
        statusVar_printers_statusPrinter17.BUSY = False
        statusVar_printers_statusPrinter17.FAULT = True
        workstationErr01 = True
    Case Else
        statusVar_printers_statusPrinter17.IDLE = False
        statusVar_printers_statusPrinter17.BUSY = False
        statusVar_printers_statusPrinter17.FAULT = True
        workstationErr01 = True
End Select

' Status for 3D Printer 18
Select Case currStatCodePrinter18
    Case statCodeIdle
        statusVar_printers_statusPrinter18.IDLE = True
        statusVar_printers_statusPrinter18.BUSY = False
        statusVar_printers_statusPrinter18.FAULT = False
    Case statCodeBusy
        statusVar_printers_statusPrinter18.IDLE = False
        statusVar_printers_statusPrinter18.BUSY = True
        statusVar_printers_statusPrinter18.FAULT = False
    Case statCodeFault
        statusVar_printers_statusPrinter18.IDLE = False
        statusVar_printers_statusPrinter18.BUSY = False
        statusVar_printers_statusPrinter18.FAULT = True
        workstationErr01 = True
    Case Else
        statusVar_printers_statusPrinter18.IDLE = False
        statusVar_printers_statusPrinter18.BUSY = False
        statusVar_printers_statusPrinter18.FAULT = True
        workstationErr01 = True
End Select

' Status for 3D Printer 19
Select Case currStatCodePrinter19
    Case statCodeIdle
        statusVar_printers_statusPrinter19.IDLE = True
        statusVar_printers_statusPrinter19.BUSY = False
        statusVar_printers_statusPrinter19.FAULT = False
    Case statCodeBusy
        statusVar_printers_statusPrinter19.IDLE = False
        statusVar_printers_statusPrinter19.BUSY = True
        statusVar_printers_statusPrinter19.FAULT = False
    Case statCodeFault
        statusVar_printers_statusPrinter19.IDLE = False
        statusVar_printers_statusPrinter19.BUSY = False

```

```
        statusVar_printers_statusPrinter19.FAULT = True
        workstationErr01 = True
    Case Else
        statusVar_printers_statusPrinter19.IDLE = False
        statusVar_printers_statusPrinter19.BUSY = False
        statusVar_printers_statusPrinter19.FAULT = True
        workstationErr01 = True
    End Select
,
' Status for Laser Cutter
,
' Status for Laser Cutter 00
Select Case currStatCodeLaserCutter00
    Case statCodeIdle
        statusVar_laserCutter_statusLaserCutter00.IDLE = True
        statusVar_laserCutter_statusLaserCutter00.BUSY = False
        statusVar_laserCutter_statusLaserCutter00.FAULT = False
    Case statCodeBusy
        statusVar_laserCutter_statusLaserCutter00.IDLE = False
        statusVar_laserCutter_statusLaserCutter00.BUSY = True
        statusVar_laserCutter_statusLaserCutter00.FAULT = False
    Case statCodeFault
        statusVar_laserCutter_statusLaserCutter00.IDLE = False
        statusVar_laserCutter_statusLaserCutter00.BUSY = False
        statusVar_laserCutter_statusLaserCutter00.FAULT = True
        workstationErr02 = True
    Case Else
        statusVar_laserCutter_statusLaserCutter00.IDLE = False
        statusVar_laserCutter_statusLaserCutter00.BUSY = False
        statusVar_laserCutter_statusLaserCutter00.FAULT = True
        workstationErr02 = True
End Select
,
' Status for Mobile Robots
,
' Status for Mobile Robots 00
Select Case currStatCodeMobileRobot00
    Case statCodeIdle
        statusVar_mobileRobot_statusMobileRobot00.IDLE = True
        statusVar_mobileRobot_statusMobileRobot00.BUSY = False
        statusVar_mobileRobot_statusMobileRobot00.FAULT = False
    Case statCodeBusy
        statusVar_mobileRobot_statusMobileRobot00.IDLE = False
        statusVar_mobileRobot_statusMobileRobot00.BUSY = True
        statusVar_mobileRobot_statusMobileRobot00.FAULT = False
    Case statCodeFault
        statusVar_mobileRobot_statusMobileRobot00.IDLE = False
        statusVar_mobileRobot_statusMobileRobot00.BUSY = False
        statusVar_mobileRobot_statusMobileRobot00.FAULT = True
        workstationErr03 = True
    Case Else
        statusVar_mobileRobot_statusMobileRobot00.IDLE = False
```

```
        statusVar_mobileRobot_statusMobileRobot00.BUSY = False
        statusVar_mobileRobot_statusMobileRobot00.FAULT = True
        workstationErr03 = True
    End Select

' Status for Mobile Robots 01
Select Case currStatCodeMobileRobot01
    Case statCodeIdle
        statusVar_mobileRobot_statusMobileRobot01.IDLE = True
        statusVar_mobileRobot_statusMobileRobot01.BUSY = False
        statusVar_mobileRobot_statusMobileRobot01.FAULT = False
    Case statCodeBusy
        statusVar_mobileRobot_statusMobileRobot01.IDLE = False
        statusVar_mobileRobot_statusMobileRobot01.BUSY = True
        statusVar_mobileRobot_statusMobileRobot01.FAULT = False
    Case statCodeFault
        statusVar_mobileRobot_statusMobileRobot01.IDLE = False
        statusVar_mobileRobot_statusMobileRobot01.BUSY = False
        statusVar_mobileRobot_statusMobileRobot01.FAULT = True
        workstationErr03 = True
    Case Else
        statusVar_mobileRobot_statusMobileRobot01.IDLE = False
        statusVar_mobileRobot_statusMobileRobot01.BUSY = False
        statusVar_mobileRobot_statusMobileRobot01.FAULT = True
        workstationErr03 = True
End Select

' Status for Mobile Robots 02
Select Case currStatCodeMobileRobot02
    Case statCodeIdle
        statusVar_mobileRobot_statusMobileRobot02.IDLE = True
        statusVar_mobileRobot_statusMobileRobot02.BUSY = False
        statusVar_mobileRobot_statusMobileRobot02.FAULT = False
    Case statCodeBusy
        statusVar_mobileRobot_statusMobileRobot02.IDLE = False
        statusVar_mobileRobot_statusMobileRobot02.BUSY = True
        statusVar_mobileRobot_statusMobileRobot02.FAULT = False
    Case statCodeFault
        statusVar_mobileRobot_statusMobileRobot02.IDLE = False
        statusVar_mobileRobot_statusMobileRobot02.BUSY = False
        statusVar_mobileRobot_statusMobileRobot02.FAULT = True
        workstationErr03 = True
    Case Else
        statusVar_mobileRobot_statusMobileRobot02.IDLE = False
        statusVar_mobileRobot_statusMobileRobot02.BUSY = False
        statusVar_mobileRobot_statusMobileRobot02.FAULT = True
        workstationErr03 = True
End Select

'
' Status for Cobots
'
' Status for Mobile Robots Cobots 00
Select Case currStatCodeCobot00
```

```
Case statCodeIdle
  statusVar_mobileRobot_statusMobileRobotCobot00.IDLE = True
  statusVar_mobileRobot_statusMobileRobotCobot00.BUSY = False
  statusVar_mobileRobot_statusMobileRobotCobot00.FAULT = False
Case statCodeBusy
  statusVar_mobileRobot_statusMobileRobotCobot00.IDLE = False
  statusVar_mobileRobot_statusMobileRobotCobot00.BUSY = True
  statusVar_mobileRobot_statusMobileRobotCobot00.FAULT = False
Case statCodeFault
  statusVar_mobileRobot_statusMobileRobotCobot00.IDLE = False
  statusVar_mobileRobot_statusMobileRobotCobot00.BUSY = False
  statusVar_mobileRobot_statusMobileRobotCobot00.FAULT = True
  workstationErr03 = True
Case Else
  statusVar_mobileRobot_statusMobileRobotCobot00.IDLE = False
  statusVar_mobileRobot_statusMobileRobotCobot00.BUSY = False
  statusVar_mobileRobot_statusMobileRobotCobot00.FAULT = True
  workstationErr03 = True
End Select

' Status for Cobots 00
Select Case currStatCodeCobot01
  Case statCodeIdle
    statusVar_cobot_statusCobot00.IDLE = True
    statusVar_cobot_statusCobot00.BUSY = False
    statusVar_cobot_statusCobot00.FAULT = False
  Case statCodeBusy
    statusVar_cobot_statusCobot00.IDLE = False
    statusVar_cobot_statusCobot00.BUSY = True
    statusVar_cobot_statusCobot00.FAULT = False
  Case statCodeFault
    statusVar_cobot_statusCobot00.IDLE = False
    statusVar_cobot_statusCobot00.BUSY = False
    statusVar_cobot_statusCobot00.FAULT = True
    workstationErr04 = True
  Case Else
    statusVar_cobot_statusCobot00.IDLE = False
    statusVar_cobot_statusCobot00.BUSY = False
    statusVar_cobot_statusCobot00.FAULT = True
    workstationErr04 = True
End Select

' Status for Cobots 01
Select Case currStatCodeCobot02
  Case statCodeIdle
    statusVar_cobot_statusCobot01.IDLE = True
    statusVar_cobot_statusCobot01.BUSY = False
    statusVar_cobot_statusCobot01.FAULT = False
  Case statCodeBusy
    statusVar_cobot_statusCobot01.IDLE = False
    statusVar_cobot_statusCobot01.BUSY = True
    statusVar_cobot_statusCobot01.FAULT = False
  Case statCodeFault
    statusVar_cobot_statusCobot01.IDLE = False
```

```
        statusVar_cobot_statusCobot01.BUSY = False
        statusVar_cobot_statusCobot01.FAULT = True
        workstationErr04 = True
    Case Else
        statusVar_cobot_statusCobot01.IDLE = False
        statusVar_cobot_statusCobot01.BUSY = False
        statusVar_cobot_statusCobot01.FAULT = True
        workstationErr04 = True
End Select

,
' Status for Quattro Robots
,
' Status for Quattro Robot 00
Select Case currStatCodeQuattro00
    Case statCodeIdle
        statusVar_quattro_statusQuattro00.IDLE = True
        statusVar_quattro_statusQuattro00.BUSY = False
        statusVar_quattro_statusQuattro00.FAULT = False
    Case statCodeBusy
        statusVar_quattro_statusQuattro00.IDLE = False
        statusVar_quattro_statusQuattro00.BUSY = True
        statusVar_quattro_statusQuattro00.FAULT = False
    Case statCodeFault
        statusVar_quattro_statusQuattro00.IDLE = False
        statusVar_quattro_statusQuattro00.BUSY = False
        statusVar_quattro_statusQuattro00.FAULT = True
        workstationErr05 = True
    Case Else
        statusVar_quattro_statusQuattro00.IDLE = False
        statusVar_quattro_statusQuattro00.BUSY = False
        statusVar_quattro_statusQuattro00.FAULT = True
        workstationErr05 = True
End Select

,
' Status for Viper Robots
,
' Status for Viper Robot 00
Select Case currStatCodeViper00
    Case statCodeIdle
        statusVar_viper_statusViper00.IDLE = True
        statusVar_viper_statusViper00.BUSY = False
        statusVar_viper_statusViper00.FAULT = False
    Case statCodeBusy
        statusVar_viper_statusViper00.IDLE = False
        statusVar_viper_statusViper00.BUSY = True
        statusVar_viper_statusViper00.FAULT = False
    Case statCodeFault
        statusVar_viper_statusViper00.IDLE = False
        statusVar_viper_statusViper00.BUSY = False
        statusVar_viper_statusViper00.FAULT = True
        workstationErr06 = True
    Case Else
```

```
        statusVar_viper_statusViper00.IDLE = False
        statusVar_viper_statusViper00.BUSY = False
        statusVar_viper_statusViper00.FAULT = True
        workstationErr06 = True
    End Select

' Status for Viper Robot 01
Select Case currStatCodeViper01
    Case statCodeIdle
        statusVar_viper_statusViper01.IDLE = True
        statusVar_viper_statusViper01.BUSY = False
        statusVar_viper_statusViper01.FAULT = False
    Case statCodeBusy
        statusVar_viper_statusViper01.IDLE = False
        statusVar_viper_statusViper01.BUSY = True
        statusVar_viper_statusViper01.FAULT = False
    Case statCodeFault
        statusVar_viper_statusViper01.IDLE = False
        statusVar_viper_statusViper01.BUSY = False
        statusVar_viper_statusViper01.FAULT = True
        workstationErr06 = True
    Case Else
        statusVar_viper_statusViper01.IDLE = False
        statusVar_viper_statusViper01.BUSY = False
        statusVar_viper_statusViper01.FAULT = True
        workstationErr06 = True
End Select

'
' Status for Master
'
' Status for the master
Select Case currStatCodeMaster00
    Case statCodeIdle
        statusVar_master_statusMaster00.IDLE = True
        statusVar_master_statusMaster00.BUSY = False
        statusVar_master_statusMaster00.FAULT = False
    Case statCodeBusy
        statusVar_master_statusMaster00.IDLE = False
        statusVar_master_statusMaster00.BUSY = True
        statusVar_master_statusMaster00.FAULT = False
    Case statCodeFault
        statusVar_master_statusMaster00.IDLE = False
        statusVar_master_statusMaster00.BUSY = False
        statusVar_master_statusMaster00.FAULT = True
        workstationErr07 = True
    Case Else
        statusVar_master_statusMaster00.IDLE = False
        statusVar_master_statusMaster00.BUSY = False
        statusVar_master_statusMaster00.FAULT = True
        workstationErr07 = True
End Select

'
' Set workstation error to appropriate tag
```

```
,
statUpdateStatus = "Set station error"
,
statusVar_workstation_printerRack00 = workstationErr00
statusVar_workstation_printerRack01 = workstationErr01
statusVar_workstation_laserCutterWorkstation = workstationErr02
statusVar_workstation_mobileRobotFleet = workstationErr03
statusVar_workstation_cobotWorkstation = workstationErr04
statusVar_workstation_quattroWorkstation = workstationErr05
statusVar_workstation_viperWorkstation = workstationErr06
statusVar_workstation_masterWorkstation = workstationErr07
,
statUpdateStatus = "Sleeping"
Thread.Sleep(sleepTime)
,
Loop Until stopVar = True
statUpdateStatus = "Stoped"
End Sub
</sCode>
<sessionName i:nil="true" />
<slowSamplingInterval>5000</slowSamplingInterval>
<stopCommandTimeout>2000</stopCommandTimeout>
<threadPriority>Highest</threadPriority>
<useAlwaysSecureConnections>>false</useAlwaysSecureConnections>
<writeTimeout>0</writeTimeout>
</ScriptDocument>
```


Appendix H

OctoPrint Communicator Script Source Code

File - C:\Devel\OctoPrintCommunicator\config.ini

```
1 # For configuring the OctoPrintCommunicator script.
2 # Customize file paths and tweak settings to suit the machine you're deploying
  it on.
3
4 [Paths]
5 ListOfPrinters = ListOfPrinters.csv
6 PrinterStatus = PrinterStatus.csv
7 PrinterCommands = PrinterCommands.csv
8 Log = Log.txt
9
10 [Settings]
11 # Connect all printers to Pis on script startup. It is recommended to use the
  connect-command for reconnections.
12 StartupAutoConnect = True
13 # If verbose is set to true, print status and info to console
14 Verbose = True
15 # HTTP Timeout threshold in seconds
16 HTTP_timeout = 2
17 # Time between program cycles in seconds
18 CycleTime = 4
```

File - C:\Devel\OctoPrintCommunicator_main__.py

```

1 from octoprintcommunication import OctoPrintClient
2 from pathlib import Path
3 from time import sleep
4 import configparser
5 import logging
6 import pandas
7 import json
8 import csv
9 import sys
10
11 '''
12 In this module, a list of IP addresses and API keys for Octoprint-connected 3D
13 printers are read, then used to create
14 instances of OctoPrintCommunicator clients. The main program then handles
15 communication between printer OPC clients
16 and various equipment.
17 '''
18 # First we need to set up the necessary variables to make the script run.
19 # Settings and file paths are read from config.ini
20 config = configparser.ConfigParser()
21 config.read('config.ini')
22
23 opcs = list() # For storing
24               all initialized OctoPrintClient objects
25 path_ListOfPrinters = Path(config['Paths']['ListOfPrinters']) # Path to the
26                       list of printer IPs / API keys
27 path_PrinterStatus = Path(config['Paths']['PrinterStatus']) # Path to where
28                       to export the printer status
29 path_PrinterCommands = Path(config['Paths']['PrinterCommands']) # Path to
30                       printer commands from the IPC
31 path_Log = Path(config['Paths']['Log']) # Where to
32                       write the error log
33 verbose = config['Settings'].getboolean('Verbose') # Toggle
34           whether to print all responses to console
35 timeoutThreshold = int(config['Settings']['HTTP_timeout']) # HTTP timeout
36                 threshold in seconds
37 cycleTime = int(config['Settings']['CycleTime']) #
38 startupAutoConnect = config['Settings'].getboolean('StartupAutoConnect') #
39                 Autoconnect to printers when starting script
40
41 # Set up Logger
42 logging.basicConfig(filename=path_Log, level=logging.INFO, format='%(asctime)s
43                       %(levelname)s %(name)s %(message)s')
44 logger = logging.getLogger(__name__)
45
46 def inferCsvFormat(path_csv):
47     '''
48     Find the delimiter/separation string used in a CSV, then set the data/
49     column header position.
50     Returns: delimiter sign (string), which row number to read headers from (
51             int)

```

File - C:\Devel\OctoPrintCommunicator_main__.py

```

41     '''
42     with open(path_csv, 'r') as csvfile:
43
44         # Infer delimiter data automatically based on first line
45         dialect = csv.Sniffer().sniff(csvfile.readline(), [';', ','])
46         csvfile.seek(0) # Return to first line of file for next read operation.
47         delimiter = dialect.delimiter
48         header = 0
49
50         csvData = csv.reader(csvfile)
51         csvDataList = list(csvData)
52         if verbose:
53             print("Opening " + str(path_csv) + " with delimiter=" + delimiter)
54
55         # If the first line contains Excel separator data, use that instead,
56         # then use next line as header
57         if csvDataList[0][0] == "sep=":
58             delimiter = ";"
59             header = 1
60         elif csvDataList[0][0] == "sep=":
61             delimiter = ","
62             header = 1
63
64         return (delimiter, header)
65
66 def importPrinterList():
67     '''
68     Import Octopi / printer IP addresses and API keys from the Local
69     ListOfPrinters.csv
70     Usernames and passwords for the Octoprint user on each Pi is included as
71     well.
72     Lists are read by parsing columns. The first row should contain the
73     following fields:
74     "ipAddress", "apiKey", "username", "password". The delimiter sign is
75     automatically inferred (, or ;).
76     All valid rows are used to create OctoPrintClient objects, which are then
77     stored in a list.
78     '''
79     global ipList
80     global apiList
81     global usernameList
82     global passwordList
83     try:
84         delimiter, header = inferCsvFormat(path_ListOfPrinters)
85
86         # Read csv to Pandas dataframe using first row as headers
87         dataframe = pandas.read_csv(path_ListOfPrinters, sep=delimiter, header=
88         header)
89
90         ipList = list(dataframe.ipAddress)
91         apiList = list(dataframe.apiKey)
92         usernameList = list(dataframe.username)
93         passwordList = list(dataframe.password)

```

File - C:\Devel\OctoPrintCommunicator_main__.py

```

87     rackIDlist = list(dataframe.rackID)
88     xPosList = list(dataframe.xPos)
89     yPosList = list(dataframe.yPos)
90
91     # Create an OPC instance for every element in the List Of Printers
92     for i in range(len(ipList)):
93         opcs.append(OctoPrintClient(ipList[i], apiList[i], usernameList[i]
94 ], passwordList[i],
95                                rackIDlist[i], xPosList[i], yPosList[i]
96 ],
97                                path_log=path_Log, timeout=
98 timeoutThreshold, verbose=True))
99
100 except Exception as e:
101     logger.error(e)
102     if verbose:
103         print("ListOfPrinters.csv may be missing or of invalid format")
104
105 def connectToPrinters():
106     '''
107     Autoconnect the Pis to their respective printers (over USB)
108     '''
109     for i, opc in enumerate(opcs):
110         if verbose:
111             print("Attempting to connect to " + opc.ipAddress)
112         if not opc.isPrinterConnected():
113             response = opc.connectToPrinter()
114             if verbose:
115                 print("HTTP " + str(response))
116         else:
117             if verbose:
118                 print("Already connected.")
119
120 def updatePrinterStatus():
121     '''
122     Read status from all available printers. Write each printer's status as a
123     row in a CSV file.
124     Also, for testing purposes, write each printer's status to a separate txt
125     file.
126     '''
127     try:
128         # Row header structure:
129         # [IP]; [connected]; [printing]; [ready]; [operational]; [pausing]; [
130 paused]; [finished]; [nozzle temp]; [bed temp]; [print job]; [rack ID]; [X pos
131 ]; [Y pos]
132         opcStatusFields = ("IP;Connected;Printing;Ready;Operational;Pausing;
133 Paused;Finished;NozzleTemp;BedTemp;PrintJob;RackID;Xpos;Ypos\n")
134
135     # Set up status CSV & txt
136     statusCsv = open(path_PrinterStatus, 'w+') # Clear file before
137     writing

```

File - C:\Devel\OctoPrintCommunicator_main_.py

```

131     statusCsv.write(opcStatusFields)           # Add headers
132
133     # Create status string for each connected printer
134     for i, opc in enumerate(opcs):
135         printerIsConnected = opc.isPrinterConnected()
136         if printerIsConnected:
137             opcStatus = opc.getPrinterStatus()
138             opcSJ = json.loads(opcStatus)
139             opcCurrentPrintJob = json.loads(opcs[0].getCurrentPrintJob())
140
141             # The "finished"-status is a local variable in the client
142             # object.
143             # It is only set when "finishing" is true, at the end of each
144             # print job.
145             # It is reset by the printer command [printerIP , currentPrint
146             # , retrieved]
147             if "true" in str(opcSJ['state']['flags']['finishing']):
148                 opc.printFinished = "true"
149
150             opcStatusString = (
151                 str(opc.ipAddress
152                 + ';' +
153                 str(printerIsConnected
154                 + ';' +
155                 str(opcSJ['state']['flags']['printing'
156                 + ';' +
157                 str(opcSJ['state']['flags']['ready'
158                 + ';' +
159                 str(opcSJ['state']['flags']['operational'
160                 + ';' +
161                 str(opcSJ['state']['flags']['pausing'
162                 + ';' +
163                 str(opcSJ['state']['flags']['paused'
164                 + ';' +
165                 str(opc.printFinished
166                 + ';' +
167                 str(opcSJ['temperature']['bed']['actual'
168                 + ';' +
169                 str(opcSJ['temperature']['tool0']['actual'
170                 + ';' +
171                 str(opcCurrentPrintJob['job']['file']['
172                 name']) + ';' +
173                 str(opc.rackID
174                 + ';' +
175                 str(opc.xPos
176                 + ';' +
177                 str(opc.yPos)
178                 )
179             )
180         else:
181             opcStatus = "Not connected to Pi!"
182             opcStatusString = (
183                 str(opc.ipAddress) + ";" +
184                 str(printerIsConnected) + ";" +

```

File - C:\Devel\OctoPrintCommunicator_main_.py

```

168         ';' +
169         ';' +
170         ';' +
171         ';' +
172         ';' +
173         ';' +
174         ';' +
175         ';' +
176         ';' +
177         ';' +
178         ';' +
179         ';'
180     )
181
182     # Append status string to CSV & txt
183     statusCsv.write(opcStatusString + "\n")
184     # Print responses if the verbose debugging variable is set to true
185     if verbose:
186         print(opc.ipAddress + " printer status: " + opcStatusString)
187
188     # Close CSV to avoid access issues
189     statusCsv.close()
190
191     except Exception as e:
192         logger.error(e)
193         if verbose:
194             print(e)
195
196     def getCommandList():
197         '''
198         Read and sort CSV containing commands for the printer. The CSV is intended
199         to be written by an IPC.
200         Returns the commands as lists of IP addresses, commands and arguments.
201         '''
202         delimiter, header = inferCsvFormat(path_PrinterCommands)
203
204         # Read csv to Pandas dataframe using the first relevant row as headers
205         dataframe = pandas.read_csv(path_PrinterCommands, sep=delimiter, header=
header)
206         ipList = list(dataframe.IP_Address)
207         commandList = list(dataframe.Command)
208         argumentList = list(dataframe.Argument)
209
210         # Create a 2D list of command data.
211         outputList = [ipList, commandList, argumentList]
212
213         open(path_PrinterCommands, 'w').close() # Clear file after parsing it
214
215         return outputList
216
217
218

```

File - C:\Devel\OctoPrintCommunicator_main__.py

```

219 '''
220 MAIN SCRIPT STARTS HERE
221
222 From the PrinterCommands.csv, each column (IP, command, argument) is parsed
and stored in a list.
223 For every cycle, the program checks if there is an IP address matching that of
a registered OctoPrint client.
224 If there is, the command is read and carried out, using arguments if
applicable.
225 '''
226 if __name__ == "__main__":
227     # Upon calling the script, printers are connected to Pis, then ran until
the script / shell is closed.
228     importPrinterList() # Must be run first. Otherwise there won't be any
OPCs to work with.
229
230     # Connection time varies between hardware and network configurations.
231     # Printers will usually take around 10 seconds to establish connection to
OctoPrint.
232     if startupAutoConnect:
233         connectToPrinters()
234         sleep(10)
235
236     while True:
237         updatePrinterStatus()
238         commandList = getCommandList()
239
240         # Parse and run commands
241         for i, opc in enumerate(opcs):
242             ipAddress = commandList[0][i]
243             command = commandList[1][i]
244             argument = commandList[2][i]
245
246             # # # If printer is connected, run commands # # #
247             if opc.isPrinterConnected:
248                 if ipAddress == opc.ipAddress:
249
250                     if command.lower() == "print":
251                         selectedFile = argument
252                         # Check if the string actually points to the files
directory
253                         if "api/files" in str(selectedFile):
254                             opc.selectPrintJob(selectedFile)
255                             opc.startPrintJob()
256                             print(ipAddress + ": attempting to print: " +
selectedFile)
257
258                         # For external applications,
259                         if command == "printRetrieved":
260                             opc.printFinished = "false"
261
262             #Administrative commands (shutdown, settings, connections, debug)
263

```


File - C:\Devel\OctoPrintCommunicator_main_.py

```
264         # Terminate script remotely
265         if "shutdown" or "exit" in ipAddress.lower():
266             msg = "Script shut down by external command"
267             logger.info(msg)
268             if verbose:
269                 print(msg)
270             sys.exit()
271
272         # (Re)connect Pi to printer
273         if command.lower == "connect":
274             if ipAddress == opc.ipAddress:
275                 opc.connectToPrinter()
276             if argument.lower == "all":
277                 connectToPrinters()
278
279         sleep(cycleTime) # Slow down cycle time to reduce congestion, as tasks
                        # are not really time sensitive.
280
```

File - C:\Devel\OctoPrintCommunicator\ListOfPrinters.csv

```
1 ipAddress,apiKey,username,password,rackID,xPos,yPos,comment
2 IPADDRESS1,APIKEY1,USERNAME1,PASSWORD1,printer rack ID,X-position from bottom
  left,Y-position (height) from bottom,comment space
3 IPADDRESS2,APIKEY2,USERNAME2,PASSWORD2,printer rack ID,X-position from bottom
  left,Y-position (height) from bottom,comment space
```

File - C:\Devel\OctoPrintCommunicator\octoprintcommunication.py

```

1 import ipaddress
2 import requests
3 import logging
4 import json
5
6
7 from requests.exceptions import ConnectionError
8 from requests.adapters import HTTPAdapter
9
10 '''
11 This class contains the necessary commands to extract information from
12 Raspberry Pis running Octoprint,
13 as well as commands to start print jobs. Communication between the script and
14 Pis are done through parsing of
15 JSON objects. Methods primarily return JSON-formatted strings.
16 '''
17
18 class OctoPrintClient:
19     def __init__(self, ipAddress, apiKey, username, password,
20                 rackID=1, xPos=1, yPos=1, path_log='Log.txt', timeout=2,
21                 verbose=False):
22         '''
23         Initialize a "client". Each client handles one connection to one
24         printer.
25         A logger object is initialized to write error logs as well.
26         '''
27         self.ipAddress = ipAddress      # Raspberry Pi IP Address
28         self.apiKey = apiKey            # Octoprint API Key
29         self.username = username        # Octoprint Username
30         self.password = password       # Octoprint User Password
31         self.rackID = rackID           # Printer rack number for this printer
32         self.xPos = xPos               # X-position of printer in rack
33         self.yPos = yPos               # Y-position of printer in rack
34         self.timeout = timeout         # HTTP timeout threshold (seconds)
35         self.printFinished = "false"   # Status to be used by external
36         applications
37         self.verbose = verbose        # Toggle whether to print responses to
38         console
39
40         logging.basicConfig(filename=path_log, level=logging.ERROR,
41                             format='%(asctime)s %(levelname)s %(name)s %(
42 message)s')
43         self.logger = logging.getLogger(__name__)
44
45     def get(self, url, headers=None):
46         '''
47         Performs a HTTP get using the Requests Library.
48         Handles some common exceptions.
49         Returns a Requests response object.
50         '''
51         try:

```

File - C:\Devel\OctoPrintCommunicator\octoprintcommunication.py

```
47         return requests.get(url, headers=headers, timeout=self.timeout)
48     except ConnectionError as e:
49         errorStr = self.ipAddress + " HTTP get: No connection to Pi"
50         self.logger.error(errorStr)
51         self.logger.error(e)
52
53
54     def post(self, url, headers=None, data=None, json=None):
55         """
56         Performs a HTTP post using the Requests Library.
57         Handles some common exceptions.
58         Returns a Requests response object.
59         """
60         try:
61             return requests.post(url, headers=headers, data=data, json=json,
62 timeout=self.timeout)
63         except ConnectionError as e:
64             errorStr = self.ipAddress + " HTTP post: No connection to Pi"
65             self.logger.error(errorStr)
66             self.logger.error(e)
67
68     def printDebugInfo(self):
69         """
70         Print relevant info about this object for debugging purposes
71         """
72         print("IP: " + self.ipAddress + ", ")
73         print("API key: " + self.apiKey + ", ")
74         print("Username: " + self.username + ", ")
75         print("Password: " + self.password + ", ")
76
77
78     def login(self):
79         """
80         Log into Octoprint on the specified IP address
81         Returns response as string.
82         """
83         url = "http://" + self.ipAddress + "/api/login"
84         json = {"user": self.username, "pass": self.password}
85         r = self.post(url, json=json)
86         if r is not None:
87             return r.text
88         else:
89             errorStr = str(self.ipAddress) + " login response: No connection to
90 Pi"
91             self.logger.error(errorStr)
92             if self.verbose:
93                 print(errorStr)
94
95     def logout(self):
96         """
97         Log out from Octoprint on the specified IP address.
```

File - C:\Devel\OctoPrintCommunicator\octoprintcommunication.py

```

98     You probably do not need to use this for this program.
99     Returns response as string.
100     '''
101     url = "http://" + self.ipAddress + "/api/logout"
102     r = self.post(url)
103     if r is not None:
104         return r.text
105     else:
106         errorStr = str(self.ipAddress) + " logout response: No connection
to Pi"
107         self.logger.error(errorStr)
108         if self.verbose:
109             print(errorStr)
110
111
112     def connectToPrinter(self):
113         '''
114         Connect to the 3D printer over USB. Default values are used.
115         Returns response code as integer.
116         Note that establishing connection may take several seconds.
117         '''
118         url = "http://" + self.ipAddress + "/api/connection"
119         headers = {"Content-Type": "application/json", "X-API-Key": self.
apiKey}
120         json = {"command": "connect"}
121         r = self.post(url, headers=headers, json=json)
122         if r is not None:
123             return r.status_code
124         else:
125             errorStr = str(self.ipAddress) + " connectToPrinter response: No
connection to Pi"
126             self.logger.error(errorStr)
127             if self.verbose:
128                 print(errorStr)
129
130     def disconnectFromPrinter(self):
131         '''
132         Disconnect from the 3D printer if connected.
133         Returns response as string.
134         '''
135         url = "http://" + self.ipAddress + "/api/connection"
136         headers = {"Content-Type": "application/json", "X-API-Key": self.
apiKey}
137         json = {"command": "disconnect"}
138         r = self.post(url, headers=headers, json=json)
139         if r is not None:
140             return r.status_code
141         else:
142             errorStr = str(self.ipAddress) + " disconnectFromPrinter response
: No connection to Pi"
143             self.logger.error(errorStr)
144             if self.verbose:
145                 print(errorStr)

```

File - C:\Devel\OctoPrintCommunicator\octoprintcommunication.py

```

146
147
148     def isPrinterConnected(self):
149         '''
150         Check if the specified Pi is connected to the Printer.
151         Returns: True if connected, False if not, None if Pi cannot be reached
152         '''
153         url = "http://" + self.ipAddress + "/api/connection"
154         headers = {"X-Api-Key": self.apiKey}
155
156         r = self.get(url, headers=headers)
157         if r is not None:
158             rJson = json.loads(r.text)
159
160             # Check for responses
161             printerCurrentState = rJson["current"]["state"]
162             if printerCurrentState == "Operational" or printerCurrentState ==
163                 "Printing":
164                 return True
165             else:
166                 return False
167         else:
168             errorStr = str(self.ipAddress) + " isPrinterConnected response: No
169             connection to Pi"
170             self.logger.error(errorStr)
171             if self.verbose:
172                 print(errorStr)
173
174     def getPrinterStatus(self):
175         '''
176         Request the current status of the connected 3D printer.
177         Returns the response (JSON object) as a string.
178         '''
179         url = "http://" + self.ipAddress + "/api/printer"
180         headers = {"X-Api-Key": self.apiKey}
181         if self.verbose:
182             print("Accessing " + url + " using API Key " + self.apiKey)
183
184         r = self.get(url, headers=headers)
185         if r is not None:
186             rJson = json.loads(r.text)
187
188             if r.text == "Printer is not operational":
189                 errorStr = "Printer " + self.ipAddress + " is not operational"
190                 self.logger.error(errorStr)
191                 if self.verbose:
192                     print(errorStr)
193             else:
194                 return r.text
195         else:
196             errorStr = str(self.ipAddress) + " getPrinterStatus response: No

```

File - C:\Devel\OctoPrintCommunicator\octoprintcommunication.py

```

196 connection to Pi
197     self.logger.error(errorStr)
198     if self.verbose:
199         print(errorStr)
200
201
202     def getCurrentPrintJob(self):
203         '''
204         Check to see what is currently printing, if anything at all.
205         Return JSON string containing job info.
206         '''
207         url = "http://" + self.ipAddress + "/api/job"
208         headers = {"X-Api-Key": self.apiKey}
209         r = self.get(url, headers=headers)
210         if r is not None:
211             return r.text
212         else:
213             errorStr = str(self.ipAddress) + " getCurrentPrintJob response: No
connection to Pi"
214             self.logger.error(errorStr)
215             if self.verbose:
216                 print(errorStr)
217
218
219     def selectPrintJob(self, gcodePath):
220         '''
221         Select gcode stored on the Pi.
222         Argument: path to the gcode
223         Returns the response as JSON string
224         '''
225         url = "http://" + self.ipAddress + gcodePath
226         headers = {"Content-Type": "application/json", "X-Api-Key": self.
apiKey}
227         json = {"command": "select"}
228         r = self.post(url, headers=headers, json=json)
229         if r is not None:
230             return r.text
231
232
233     def startPrintJob(self):
234         '''
235         Tell the specified printer to start printing the selected G-code
236         G-code should be uploaded to the Manulab-folder in octoprint in
advance.
237         Arguments: Octopi IP address, Octopi API key, path to g-code file on
the pi
238         Returns: response as JSON string
239         '''
240         url = "http://" + self.ipAddress + "/api/job"
241         headers = {"Content-Type": "application/json", "X-Api-Key": self.
apiKey}
242         json = {"command": "start"}
243         r = self.post(url, headers=headers, json=json)

```

File - C:\Devel\OctoPrintCommunicator\octoprintcommunication.py

```
244     if r is not None:
245         return r.text
246     else:
247         if self.isPrinterConnected():
248             errorStr = str(self.ipAddress) + " startPrintJob response:
Could not start print job. Printer might be busy"
249             self.logger.error(errorStr)
250         else:
251             errorStr = str(self.ipAddress) + " startPrintJob response:
Could not start print job. No connection to Pi"
252             self.logger.error(errorStr)
253         if self.verbose:
254             print(errorStr)
```


File - C:\Devel\OctoPrintCommunicator\PrinterCommands.csv

```
1 IP_Address,Command,Argument
2 IP_ADDRESS1,print,/api/files/local/example.gcode
3 IP_ADDRESS2,retrievedPrint,
4
```

File - C:\Devel\OctoPrintCommunicator\requirements.txt

```
1 certifi==2019.11.28
2 chardet==3.0.4
3 idna==2.9
4 numpy==1.18.1
5 pandas==1.0.1
6 python-dateutil==2.8.1
7 pytz==2019.3
8 requests==2.23.0
9 six==1.14.0
10 urllib3==1.25.8
11
```

File - C:\Devel\OctoPrintCommunicator\README.md

```
1 # OPC: OctoPrint Communicator
2
3 A Python 3 script for establishing basic communication between Octoprint-
  connected 3D printers and various equipment.
4
5 Communication between the script and printers is carried out over HTTP using
  the Requests library with the Octoprint REST API. To and from the IPC, CSV-
  files are used.
6
7 For those simply looking for a Python-based Octoprint client, there are
  probably better alternatives out there.
8
9 ### Main idea
10 This script is meant to run natively on an Omron NY-series IPC connected to
  industrial automation equipment, so as to save us the hardships of handling all
  communications using IEC 61131-3-compliant languages.
11 When initialized, a CSV containing IPs and API keys for each Octoprint instance
  are parsed and used to create client objects for each printer.
12 These clients then handle reading and writing data to the printers.
13
14 ### How to use it
15 First of all, edit ``config.ini`` and ``ListOfPrinters.csv`` to suit your
  needs. Set IP addresses, Octoprint API keys, where to write status messages,
  logs, verbosity, HTTP timeout thresholds and more. The script is invoked from a
  command shell on the controller. This can be done e.g. in SCADA software or
  Windows' own Command Prompt:
16
17 ``python /path/to/script/__main__.py``
18
19 Based on the entries in *ListOfPrinters.csv*, a list of Octoprint Client
  objects are initialised and used to represent each printer.
20
21 Periodically, the printers' status are written to a CSV, and another one -
  containing commands from the IPC - are read and parsed by the script. This file
  is written by the IPCs internal controller and cleared by the script after it
  has parsed the commands.
22
23 *Copyright © 2020 Fredrik Siem Taklo. MIT License.*
24
```

Appendix I

AddTextToDXF Program Source Code

APPENDIX I. ADDTEXTTODXF PROGRAM SOURCE CODE

File - C:\Users\Fredrik\Desktop\AddTextToDXF-master\AddTextToDXF.py

```
1 '''
2   ### Information for use of AddTextToDXF ###
3
4   @author Hans-Christian Ringstad
5
6   This script was made to be used in the Bachelor Thesis Manulab spring 2020, it
7   will put a Logo image and a text on a
8   dxfile template. Files used in this program needs to be put in the same
9   folder as the program. For the source code
10  Python 3.8.1 was used.
11
12  ### Files needed to run the program ###
13  info.csv: Contains information for the program to add text and Logo image. This
14  csv-file uses ";" as separator and the
15  program will only read the info in the last row.
16  This file will need to contain these fields;
17  # Name of field # Datatype # Comment
18  finalFileName : String : Name of the dxf file that will be created
19  templateName : String : Name of the template to be used as background
20  logoFileName : String : Name of the Logo PNG-file
21  xInsertPointLogo: float : x-coordinate of the insert point of logo, the
22  insertion point will be at the bottom right
23  yInsertPointLogo: float : y-coordinate of the insert point of logo, the
24  insertion point will be at the bottom right
25  xPixelSize : int : x length in pixels
26  yPixelSize : int : y length in pixels
27  xSizeInmm : float : x length in millimeter
28  ySizeInmm : float : y length in millimeter
29  rotationLogo : float : Rotation of Logo
30  textToInsert : String : Text to insert on the dxf file
31  textStyle : String : Text style
32  xInsertPointText: float : x-coordinate of the insert point of text, the
33  insertion point will be at the top center
34  yInsertPointText: float : y-coordinate of the insert point of text, the
35  insertion point will be at the top center
36  rotationText : float : Rotation of text
37  alignmentText : int : Aligment for the text*
38  textWidth : float : Width of text fields, does not cut individual
39  words
40  textHeight : float : Height of text
41  # # #
42  *See doc for more info: https://ezdxf.mozman.at/docs/dxfentities/mtext.html#
43  ezdxf.entities.MText.dxf.attachment_point
44
45  templateName.dxf: The file containing the dxf template to be used by the
46  program, the name of the file is to be decided
47  in info.csv
48
49  LogoFileName.png: The file containing the the png of the Logo in use to be used
50  by the program, the name of the file is
51  to be decided in info.csv
52
53  ### Files created by the program ###
```

File - C:\Users\Fredrik\Desktop\AddTextToDXF-master\AddTextToDXF.py

```
43 status.csv: Contains information on status of the program. This csv-file uses  
    ";" as separator and the program will  
44    write the status at the bottom row read the info in the last row.  
45    This file will contain these fields;  
46    # Name of field # Datatype # Comment  
47    working      : boolean : True when active, false if inactive  
48    done         : boolean : True when the dxf file was created succesfully  
49    error        : boolean : True when an error has ocurred  
50    #           #           #  
51  
52 finalFileName.dxf: The final dxf file created by the specifications in info.csv  
   ,the name of the file is to be decided in  
53    info.csv  
54  
55 errLog.txt: All expected errors will be printed to this file with an error  
   message of what has caused it.  
56 '''  
57 import datetime  
58 import sys  
59 import ezdxf  
60 import os  
61 import csv  
62 import time  
63  
64 '''  
65 Error log  
66 '''  
67  
68  
69 def errLog(errType, errMsg, stopProg, updateStatus):  
70     with open("errLog.txt", "a") as text_file:  
71         __ = text_file.write(str(datetime.datetime.now()) + " | " + str(  
errType) + ": " + errMsg + "\n")  
72     if updateStatus:  
73         __ = updateStatus(_statusFileName, False, False, True)  
74     if stopProg:  
75         sys.exit()  
76     return True  
77  
78  
79 '''  
80 Write to status.csv  
81 '''  
82  
83  
84 def updateStatus(statusFileName, working, done, error):  
85     returnVal = False  
86     attempts = 0  
87     maxAttempts = 50  
88     while (not returnVal) & (attempts < maxAttempts):  
89         try:  
90             with open(statusFileName, 'w', newline='') as csvfile:  
91                 sep = ';'}
```

File - C:\Users\Fredrik\Desktop\AddTextToDXF-master\AddTextToDXF.py

```

92         statusWriter = csv.writer(csvfile, delimiter=sep, quotechar=
' ', quoting=csv.QUOTE_MINIMAL)
93         statusWriter.writerow(['sep=' + sep])
94         statusWriter.writerow(['working', 'done', 'error'])
95         statusWriter.writerow([working, done, error])
96         returnVal = True
97     except PermissionError:
98         errType = str(PermissionError)
99         returnVal = False
100        attempts = attempts + 1
101        __ = errLog(errType, " has occured at " + statusFileName + '.
Attempt ' + str(attempts), False, False)
102        time.sleep(_permWaitTime)
103    if not returnVal:
104        __ = errLog(errType, " has occured at " + statusFileName + ". Force-
stops program after " + str(attempts)
105                + " attempts.", True, False)
106    return returnVal
107
108
109 '''
110 Removes file from system
111 '''
112
113
114 def removeFile(fileName):
115     returnVal = False
116     attempts = 0
117     maxAttempts = 50
118     while (not returnVal) & (attempts < maxAttempts):
119         try:
120             returnVal = True
121             os.remove(fileName)
122         except FileNotFoundError:
123             returnVal = True
124             errType = str(FileNotFoundError)
125             __ = errLog(errType, " File " + fileName + " was not found when
trying to delete it.", False, False)
126             pass
127         except PermissionError:
128             returnVal = False
129             errType = str(PermissionError)
130             __ = errLog(errType,
131                 " File " + fileName + " is in use when trying to
delete it. Close all programs using it", True,
132                 True)
133         if not returnVal:
134             __ = errLog(errType, " has occured at " + _statusFileName + ". Force-
stops program after " + str(attempts)
135                     + " attempts.", True, False)
136     return returnVal
137
138

```

File - C:\Users\Fredrik\Desktop\AddTextToDXF-master\AddTextToDXF.py

```
139 '''
140 Get a dxf file
141 '''
142
143
144 def getDXF(dxfileFileName):
145     try:
146         temp = ezdxf.readfile(dxfileFileName)
147     except FileNotFoundError:
148         __ = errLog(str(FileNotFoundError), " has occurred. " + dxfileFileName +
149 " were not found", True, True)
150     except PermissionError:
151         __ = errLog(str(PermissionError), " has occurred. Please close all
152 program using " + dxfileFileName
153 + " before continuing ", True, True)
154     return temp
155
156
157 Setup
158 '''
159 _permWaitTime = 0.1
160 _errType = "No Error"
161 _statusFileName = 'status.csv'
162 _infoFileName = "info.csv"
163
164 Update status
165 '''
166 status = updateStatus(_statusFileName, True, False, False)
167
168 Read info.csv
169 '''
170 with open('info.csv') as csvfile:
171     _infoReader = csv.reader(csvfile, delimiter=' ', quotechar='|')
172     for row in _infoReader:
173         _info = row
174         _info = ', '.join(_info)
175         _info = _info.replace(', ', '')
176         _info = _info.split(";")
177
178 Deletes the dxf file before saving the new file, also ignores exception
179 FileNotFoundError if file does not exist
180 '''
181 _dxfFileName = str(_info[0]) # finalFileName
182 removeFile(_dxfFileName)
183
184 Get template from file and create the modelspace to add design
185 '''
186 _dxfTemplateFileName = str(_info[1]) # templateName
187 _doc = getDXF(_dxfTemplateFileName)
188 _msp = _doc.modelspace()
189
190 Add NTNU Manulab logo
```


File - C:\Users\Fredrik\Desktop\AddTextToDXF-master\AddTextToDXF.py

```
189 '''
190 _logoFileName = str(_info[2]) # LogoFileName
191 _xInsert = float(_info[3]) # xInsertPointLogo
192 _yInsert = float(_info[4]) # yInsertPointLogo
193 _xPixel = int(float(_info[5])) # xPixelSize
194 _yPixel = int(float(_info[6])) # yPixelSize
195 _xSize = float(_info[7]) # xSizeInmm
196 _ySize = float(_info[8]) # ySizeInmm
197 _rotLogo = float(_info[9]) # rotationLogo
198
199 _my_image_def = _doc.add_image_def(filename=_logoFileName, size_in_pixel=(
    _xPixel, _yPixel))
200 _image = _msp.add_image(image_def=_my_image_def, insert=(_xInsert, _yInsert),
    size_in_units=(_xSize, _ySize),
201                    rotation=_rotLogo)
202 '''
203 Add name text from input
204 '''
205 _textFileName = str(_info[10]) # textToInsert
206 _textStyle = str(_info[11]) # textStyle
207
208 _xLoc = float(_info[12]) # xInsertPointText
209 _yLoc = float(_info[13]) # yInsertPointText
210 _rotation = float(_info[14]) # rotationText
211 _alignment = int(float(_info[15])) # alignmentText
212 _textWidth = float(_info[16]) # textWidth
213 _textHeight = float(_info[17]) # textHeight
214 _strNameLen = int(float(_info[18])) # strNameLen
215
216 _strName = _textFileName.replace('\n', ' ')
217 _strName = _strName[:_strNameLen]
218
219 _mtext = _msp.add_mtext(_strName, dxattrs={'style': _textStyle}).
    set_location((_xLoc, _yLoc), _rotation, _alignment)
220 _mtext.dxf.width = _textWidth
221 _mtext.dxf.char_height = _textHeight
222 '''
223 Save the new file
224 '''
225 _doc.saveas(_dxfFileName)
226 status = updateStatus(_statusFileName, False, True, False)
```