Implementing an agitator with a soft starter controlled by a new HMI

LUNDS UNIVERSITET
Lunds Tekniska Högskola

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

Bachelor thesis:
Fredrik Johansson
Sara Malmström

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

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

Printed in Sweden
Lunds universitet
Lund 2017

Abstract

When an agitator is in commission for an extended period of time the impellers can become clogged. In order to clean the impeller, the agitator is runned backwards. This is one of the main functions for a product that CG Drives & Automation is developing for a customer in the agitator business. The purpose of this bachelor thesis is to develop and test this product for CG Drives & Automation.

The product is an automated smart agitator control system. The heart of the product is a B&R PLC that is connected to a soft starter, a panel that runs a HMI as well as various digital outputs for external alarms and actions. By using a soft starter, the mechanical stress in starts and stops will be reduced. The soft starter is built and manufactured by CG Drives & Automation. The HMI is presented through a touchscreen panel where an operator can get a good overview of the process and, if necessary, manually run cleaning sequences or acknowledge alarms. The components are built into a cabinet by Rittal which is installed on site near the agitator.

These functionalities, with minimized mechanical stress, improved safety features along with alarms, will lead to a decrease in maintenance as well as an increase in uptime. And long term, the advantages will lead to revenues for companies using this product.

The soft starter is connected to the PLC both through hard wires (IOs) and Modbus TCP/IP. The hard wiring is used for crucial commands that are not allowed to fail, such as start and stop. Modbus handles the communication of other data and functions, such as the current and power percentage that is displayed on the screen. The PLC code and HMI were made in B&Rs Automation Studio software. The languages used for the PLC code were ladder, function block, and structured text.

Despite the numerous amounts of technical difficulties that arose during the project, it was completed on time and according to the specification. However, the setup has not been tested in the correct environment because a suitable agitator was not accessible. A smaller electrical motor was used as a substitute for the real agitator.

Keywords: Automation, PLC, HMI, soft start, agitator

Sammanfattning

När en omrörare är i drift under en längre tid kan rotorbladen bli igensatta med smuts eller materialet som omröres. För att rengöra rotorbladen kan man köra omröraren baklänges. Detta är en av huvudfunktionerna i en produkt som CG Drives & Automation utvecklar för en kund inom omrörarbranschen. Att ta fram denna produkt är syftet med detta examensarbete.

Produkten är ett smart styrsystem för omrörare. Produkten styrs av en PLC från B&R som är kopplad till en mjukstartare, en panel med ett HMI samt utgångar för larm och andra funktioner. Genom att använda en mjukstartare minskar den mekaniska belastningen på omröraren kraftigt. Mjukstartaren byggs och säljs av CG Drives & Automation. HMI:et presenteras med en touchpanel där en operatör kan få en bra överblick över processen och också köra sekvenser manuellt och kvittera larm. Hela lösningen är inbyggd i ett skåp från Rittal och installeras nära omröraren.

Dessa funktioner, som ger minskad mekanisk påfrestning, högre säkerhet och bättre kontroll leder till minskad service, och längre drifttid för omröraren. Detta sparar pengar för företagen som använder sig av produkten.

Mjukstartaren är kopplad till PLC:n både via hårdtrådade kablar och via Modbus TCP/IP. De hårdtrådade kablarna används för kritiska funktioner som start och stopp medan Modbus hanterar datakommunikation för att visa variabler som motorström och effekt på panelen. PLC-koden och och HMI'et utvecklades i B&Rs Automation Studio programvara. Programspråken som användes för PLCn var ladder diagram, funktionsblock samt strukturerad text.

Trots många tekniska problem längs projektets gång så blev produkten klar i tid enligt den angivna specifikationen. Dock testades den aldrig ute på fält vid en riktig omrörare, utan en mindre motor fick användas som simulering.

Nyckelord: Automation, PLC, HMI, Mjukstartare, omrörare.

Foreword

We would like to extend our sincerest gratitude to the company, CG Drives & Automation, not only for providing us with a bachelor thesis, but, for all the help and kindness they've shown us throughout the project.

We further extend our gratitude to our supervisor at CG, Mikael Mårtensson, for being so kind and taking good care of us. We would also like to thank Anders Nilsson for his dedication towards this project and for the excellent help with the soft starter.

We also want to thank our supervisor and examiner at Lund's University, Henriette Weibull and Mats Lilja, for their dedication and help throughout this project.

Last but not least, the coffee machine at CG Drives & Automation, without you, this wouldn't have been possible.

Abbreviations

B&R	:	Bernecker & Rainer
CG	:	Crompton Greaves
CPU	:	Central processing unit
DOL	:	Direct On Line
EINT	:	Emotron Integer
FBD	:	Function Block Diagram
GND	:	Ground
HMI	:	Human Machine Interface
HTML	:	Hyper Text Markup Language
INT	:	Integer
LD	:	Ladder Diagram
MSB	:	Most Significant Bit
I/O’s	:	Inputs and Outputs
OPC	:	Open Platforms Communications
OPC UA	:	OPC Unified Architecture
PLC	:	Programmable Logic Controller
PPU	:	Portable Presentation Unit
PTC	:	Positive Temperature Coefficient [thermistor]
ST	:	Structured Text
TCP	:	Transmission Control Protocol
TSA	:	Torque Start generation A
UINT	:	Unsigned Integer
VAC	:	Voltage in Alternating Current
VDC	:	Voltage in Direct Current
XML	:	Extensible Markup Language

List of contents

1 Introduction

1.1 Background

The thesis work was done for CG Drives & Automation. CG Drives & Automation is a medium sized company located in Helsingborg, Sweden, and they mostly develop and manufacture soft starters and frequency converters for electrical motors. In addition to the commissioning and deployment of projects involving their own products, CG Drives & Automation are also involved in other automation projects. [1] One such project is the development, testing and installation of a soft starter, a PLC, and an HMI to control the electrical motor in an agitator for one of CG Drives & Automation's clients. The aim of this thesis work is to complete this project from start to finish.

As the scope of the project contains a wide array of different products and devices, the most important ones will be outlined under their own respective headlines.

1.2 Aims and objectives

The goal of this project is to have a fully functional product ready to be delivered to the customer in accordance to the agreed time frame of the 1st – 2nd of June.

1.3 Research Questions

The main research questions during this is project relates to the implementation and development of the product. Since this represents the bulk of the project it is naturally also where the focus of the research questions will be. In addition, there are also questions regarding the design and the documentation.

1. How do we control specific sequences for an Agitator with a PLC via a soft starter?
2. How is a complete HMI developed for the application?
3. How do we connect a soft starter, HMI, and an electric motor with a PLC?
4. How should the electrical cabinet be designed?
5. How should the project be documented so that further development and service will be easy to perform?

1.4 Limitations

The products and brands that CG Drives & Automation uses are selected by people at the products department, we will work with what they select. The electrical cabinet along with its components are built by professional cabinet

builders. However, we supplied the electrical schematics to the cabinet builders.

Furthermore, we did not have the sufficient clearances or certificates for installing three-phase equipment and would not partake in the installation of the product in a hands-on manner, although we would be able to assist with last minute development changes.

2 Technical background

The main technical components of this project are the agitator, the PLC, the HMI as well as the soft starter. Along with the functional description these will described in this chapter.

Since the technical aspects of the soft starter requires a technical description of general motor starts that is also included.

2.1 Functional Description

The functional description is supplied by the customer. They presented their specifications through a document with mostly general functions without technical details, leaving that up to us. They did not have any wishes when it came to specific component brands.

The customer's desired functions are the following:

- Automatic cleaning of the impellers by rotating backwards.
- Automatic filling of an external water trap designed to keep methane gas from rising from the tanks where the agitator is in drift.
- Smooth starts.
- Motor load monitoring.
- Alarm management:
    - Impeller clogging, motor power to high.
    - Water trap.
    - Motor temperature.

Figure 1 shows a mapping of how the project is implemented.

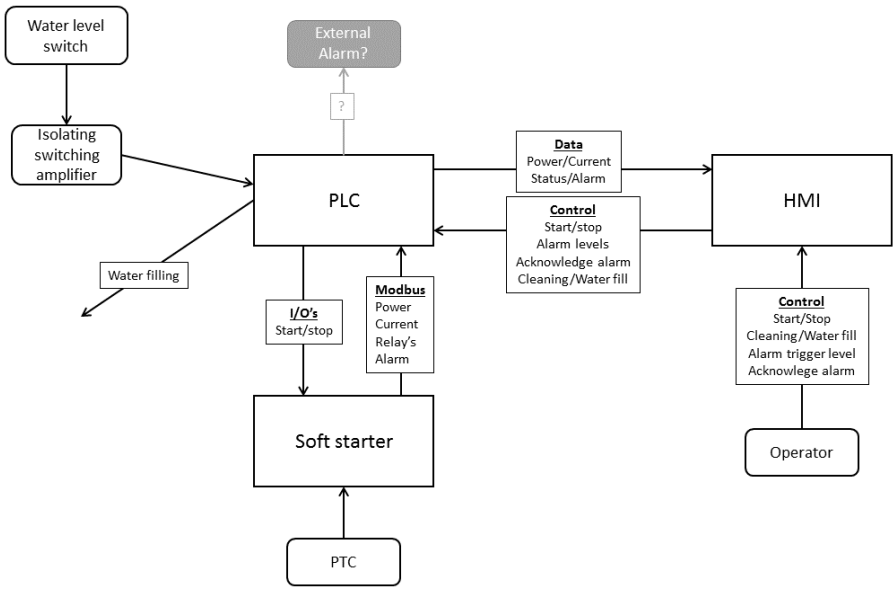


Figure 1. A map describing the system.

2.2 The Agitator

The agitators in this project are agitators used in sludge tanks and in the process industry. Present-day agitators only have a start and stop function. The benefits with the product will be less downtime and therefore cost saving. This is thanks to the following assets of the setup:

- The soft starter will minimize the stress on the motor during starts.
- The automatic cleaning sequence of the impellers will decrease the maintenance and therefore the downtime.
- By automatically filling up the water traps, problems with methane gas can be avoided.

Already installed on the agitators are a change gear mechanism, a digital water-level sensor, and a PTC thermistor. [2]

2.3 Motor starts

There are several different ways to start an electrical three-phase motor without a soft starter. The curves for these start-ups compared to those of a soft starter are shown in Figure 4.

2.3.1 Direct On Line starting (DOL)

As the name suggests, this method means that the motor is simply connected to the AC-power supply through a circuit breaker. In case of a larger motor, contactors and overload relays may also be used. When the circuit breaker is switched on, there will be a high current in the motor, about five to ten times the full motor load current. The rotor will begin to rotate with a very high torque and as the rotor accelerates up towards full speed the current will decrease. [3][4]

2.3.2 Star-delta motor start

The star-delta motor start uses reconfigured motor windings in order to drop the voltage on each motor winding during start up. When the motor starts, the windings are configured like a star, as seen in figure 2. This leads to each motor winding only receiving $\frac{1}{\sqrt{3}}$ of the voltage, causing a lower current and lower starting torque. When the motor reaches a higher speed, or after a set time of interval contactors switch over and the motor starts running on delta windings. This is a very common start up method for electrical three phase motors. [3]

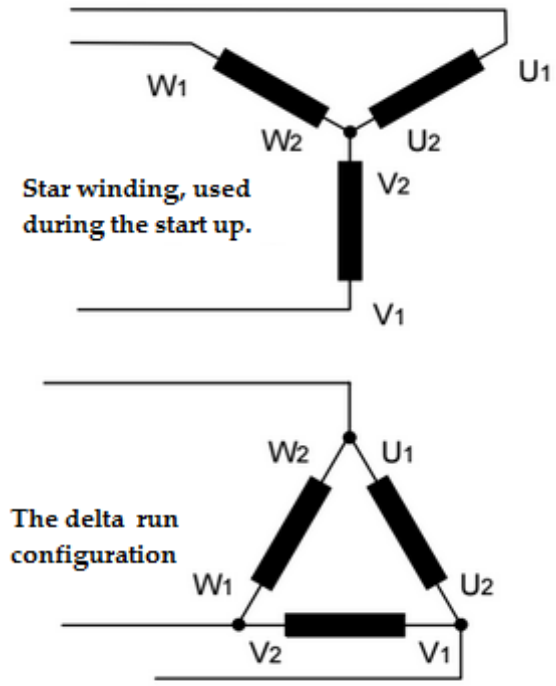


Figure 2. Star-Delta configuration

2.3.3 Autotransformer starting

The third way of starting the motor is with an autotransformer. An autotransformer is a transformer where the two windings are electrically coupled and the transformer can therefore be tapped to deliver an output that is any percent of the full voltage. A normal sequence is when a transformer is tapped for three different output percentages, from low to high. This decreases the voltage and torque but there are still jumps when the transformer changes from one tapping to the other. [3]

2.3.4 Soft starter

A soft starter is an electrical device used during starts of electrical motors to reduce the torque or the current. There are two main reasons to control the torque during start-up. The first reason is to reduce the mechanical stress on the motor and shaft. Reducing the mechanical stress will prolong the life of the motor and the cables into the motor. The second reason is due to the fact that the electrical distributor sometimes require big industries to use soft starters to reduce the current peaks on the distribution network.

On a power electronic level, a soft starter brings down the current by chopping the sinus curve with thyristors. These thyristors are often referred to as SCR, which stands for silicon controlled rectifiers, originally a trade name from General Electric for a type of their thyristors. [3][4][5]

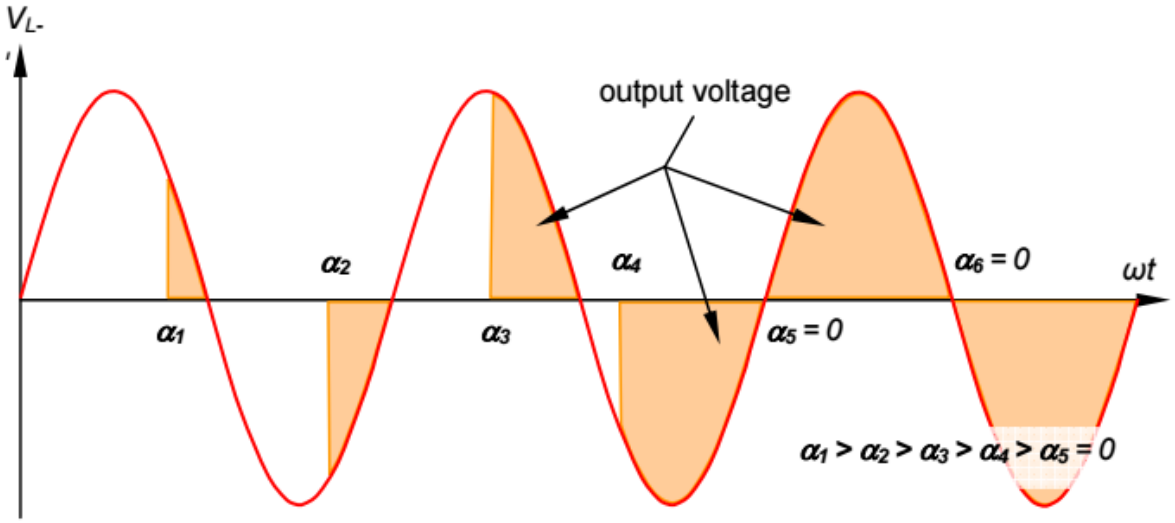


Figure 3. The chopped sinus curve with marked output voltage. [3]

As seen in figure 3, the firing angle α of the thyristors controls the output voltage and is gradually getting bigger as time progresses. The time it takes for the output voltage to reach its maximum is called ramp time. The ramp time needed can greatly vary depending on the load, motor, and for what type of application the soft starter is being used.

There are two main ways a soft starter can work in order to set the ramp time and the firing angle for the thyristors.

The first is called “open loop control”, where the output voltage, set by the firing of the thyristors, follows a time dependant template. As the name suggests, there is no feedback loop and the soft starter works without knowledge of the motor speed or current. Usually the starting voltage is set at somewhere around 10% to 20%, and then it ramps through the set template. [3]

The second way is called a “closed loop control” which is utilized by the soft starter used in this project. A controller or a control board is used to calculate a more optimal ramping with the use of a feedback loop. One way this can be achieved is by measuring the current in all three phases, and if the current increases too fast, decrease the firing angle of the thyristors, or simply hold the same angles until the speed of the motor becomes higher and the current drops. The current over time with the TSA soft starter compared to the previously explained starting methods for electrical three phase motors is shown in Figure 4. At the cost of time, the current, and subsequently the torque will severely decrease. [3][4]

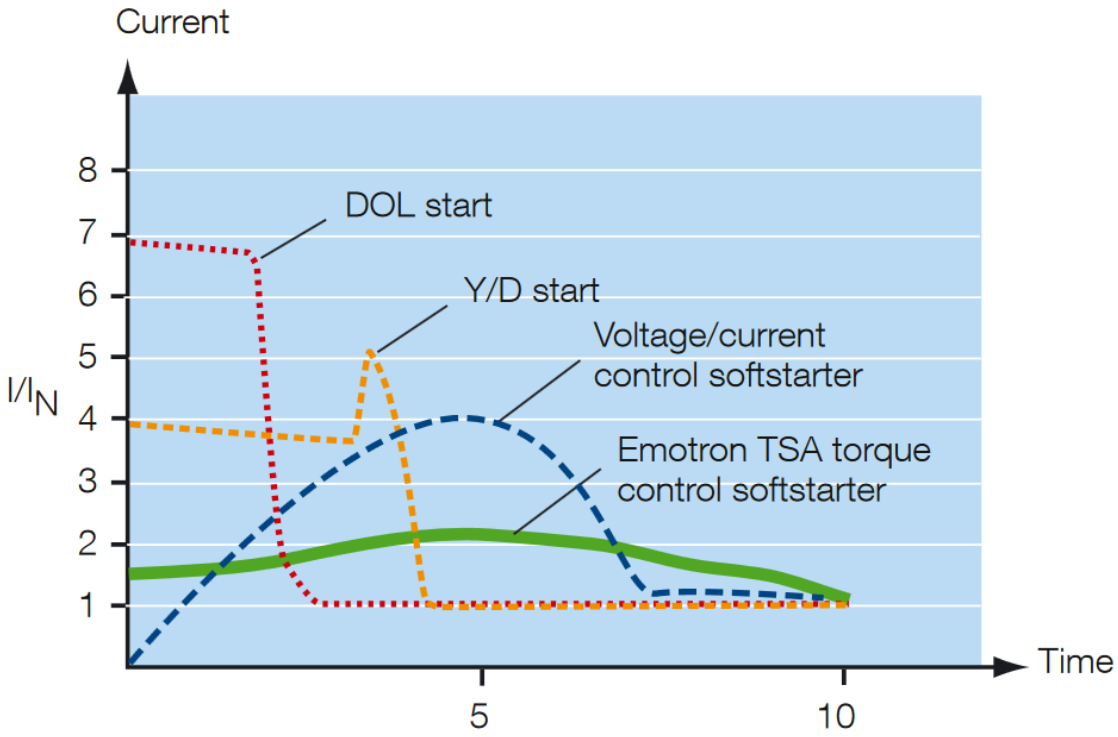


Figure 4. Current over time for the Emotron TSA torque control soft starter. [19]

The opposite of a soft start, a soft stop, is also a function available with a soft starter. If the soft starter is equipped with contactors, they will close and the current will go through the soft starter components. The sinus curve will once again be chopped by the thyristors, but this time from maximum to minimum, causing the motor to decelerate, since it cannot provide the necessary torque required for the load. As mentioned previously in this text, this can be done with both an open and closed loop, with the latter offering a higher degree of precision and more effective motor control. Something worth noting is that a soft stop is not a quicker way to stop a machine than letting it roll out and should therefore be used when the stopping time needs to be increased. An example of where a soft stop could be needed is when a gearbox would cause a lot of mechanical stress on certain components if the motor stops rotating too quickly. [5]

A soft starter can also be used as a brake for quicker stops. This is done as follows, one thyristor fires in the positive direction in one phase, and another thyristor fires in the negative direction in a second phase, both with a very narrow firing angle per cycle. As a result, a high DC current will flow through the motor and this leads to stationary torque field in the stator. This is called DC braking because of the DC current injected, and will cause the motor to break quickly. [3] [5]

The power losses when using a soft starter together with by-pass contactors are close to zero. By-pass contactors are used to lead the current through the

by-pass outside the soft starter windings instead of through it when the motor is at full speed. Another benefit is that since there will not be any power loss, there will not be any heat produced, and the soft starter can be locked in an electrical cabinet without having any cooling system. In addition to this, the lifetime of the soft starter will increase when it is only used during starts and stops instead of continuously. [3]

2.4 PLC

A PLC, Programmable Logic Controller, is a controller based on a microprocessor often used to control industrial processes. PLC uses programmable memory to remember instructions and implement functions. [6]

A PLC consists of the following [6]:

- Processor unit or CPU: This is the part of the PLC that carries out the control actions according to the programs.
- Memory unit: Where the roles and instructions are stored.
- I/O's: Often both analogue and digital inputs and outputs.
- Other communication interfaces: For example, fieldbus or communications between PLC's.
- Developer environment: Where the developer is programming the roles and instructions for the PLC.

The differences between a computer and a PLC are that the computer is optimized for calculations and display tasks whereas the PLC is optimized for control tasks. Other functions the PLC has are I/O's and an advantage of the PLC is that it is easily programmed with due to the languages are easy to understand. The I/O's are used to receive the external data on the inputs from sources such as sensors and for the program to know what to send, to which external devices. [6]

A PLC operates through four states [7]:

- Input scan
- Program scan
- Output scan
- Communications with programming terminals, internal diagnostics, etc...

To make it easier for developers and more cost effective for companies to switch between manufacturers, an International Electrotechnical Commission (IEC) standard was published, IEC 61131. The standard has 8 different parts as follow [6 direct quote]:

- Part 1: General definition of basic terminology and concepts.
- Part 2: Electronic and mechanical equipment requirements and verification tests for PLCs and associated equipment.
- Part 3: Programming languages. Five languages are defined: ladder diagram (LD), sequential function charts (SFC), function block diagram (FBD), structured text (ST), and instruction list (IL).
- Part 4: Guidance on selection, installation, and maintenance of PLCs.
- Part 5: Software facilities needed for communication with other devices based on the Manufacturing Messaging Specification (MMS).
- Part 6: Communications via fieldbus software facilities.
- Part 7: Fuzzy control programming.
- Part 8: Guidelines for the implementation of PLC programming languages defined in Part 3.

2.4.1 IEC 61131-3: Programming Languages

2.4.1.1 Ladder diagram (LD)

LD is the most traditional way to program a PLC. The basic principle is to use switching operations with logic (AND, OR, etc) and latches. The programming takes place between two vertical lines representing +VDC and GND. If the inputs are "closed", the circuit will be closed and the output can be set. It is also possible to use special blocks, so called function blocks, to perform operations, such as timers. Example of source code can be viewed in figure 5 and figure 6. [6] [8]

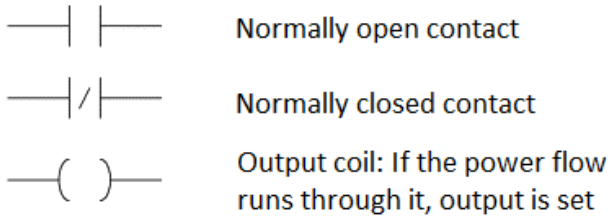


Figure 5. The most common signs in LD

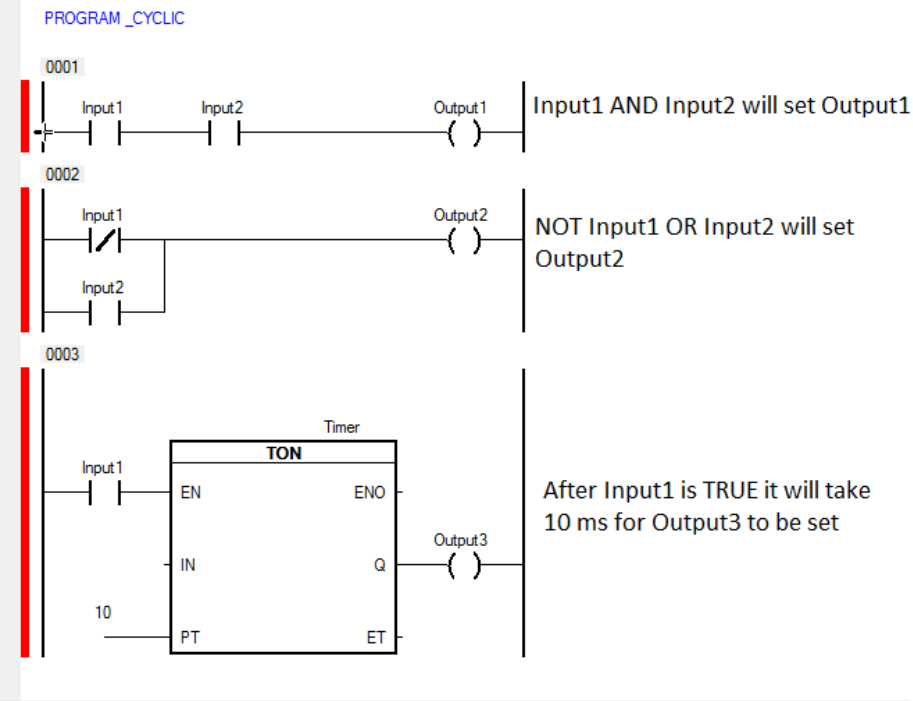


Figure 6. Example of LD source code.

2.4.1.2 Function block diagram (FBD)

FBD is a graphical language where signals and data run through blocks containing functions and instructions to process the signals and data to an output. Blocks can contain logics such as AND and OR, or it could be timers, mathematical, or Set/Reset blocks. It is possible to create new function blocks. For example, if a program is used multiple times, it can be easier to place the program inside of a block than rewriting the program every time it is needed. Example of source code can be found in figure 7. [6][8]

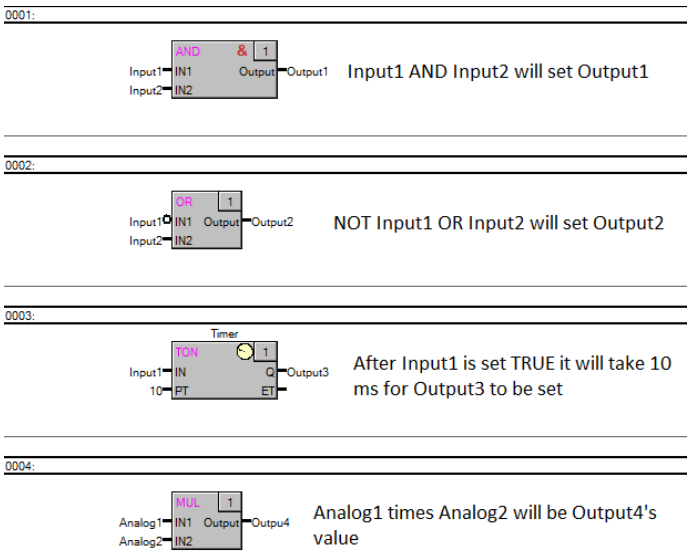


Figure 7. Example of FBD source code.

2.4.1.3 Structured Text (ST)

ST is a high-level programming language where the syntax is similar to that of PASCAL or C. Besides the normal syntax like IF, FOR, and WHEN, it is possible to use function blocks and logics. To use logics, one simply writes "output := input1 <logical operator> input2;". The syntax for function blocks is individual for each block; there should be examples in the PLC developer environment's help site. Example of source code can be found in figure 8. [6][8]

```
PROGRAM _CYCLIC
	Output1:= Input1 AND Input2;

	Output2:= NOT Input1 OR Input2;

	(*Timer delay output by 1 s and 200 ms*)
	(* call function block *)
	TON_01( IN:=input, PT:=T#1s200ms );
	(* assign results to variables *)
	output := TON_01.Q;
	elapsedTime := TON_01.ET;



END_PROGRAM
```

Figure 8. Example of ST source code

2.4.1.4 Sequential Function Charts (SFC)

If a process with well-defined sequences, for example a traffic-light or an elevator, should be programmed; SFC could be a good choice. SFC is based on different states where each state, between the states are transition conditions. To go from one state to the next state, the transition condition must be fulfilled. Example of source code can be found in figure 9. [6]

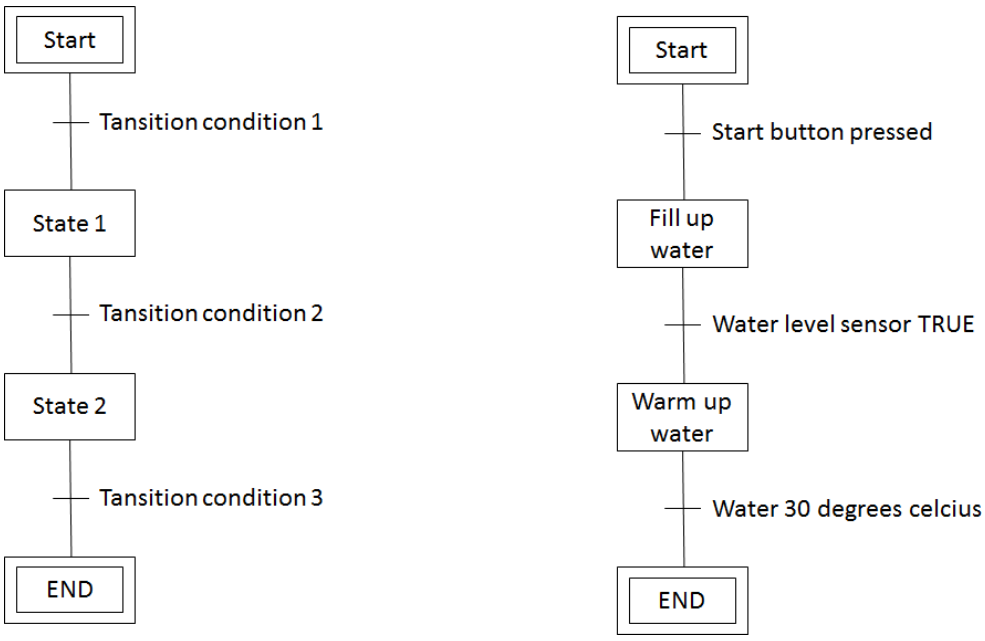


Figure 9. Example of SFC source code

2.4.1.5 Instruction lists (IL)

Instruction list is a low-level, assembler-like, language, mainly used in small and simple programs. Even if IL is in the IEC 61131-3 standard, some manufacturers do not support it. This is because many developers use the more powerful text-based language structured text. IL instructions translated into LD can be seen in table 1. Example of source code can be found in figure 10. [6]

Instruction list	Equal in Ladder diagram
LD	
LDN	
ST	

Table 1. IL equivalent in LD

```
PROGRAM _CYCLIC

        LD      Input1
        AND     Input2          Input1 AND Input2 will set Output1
        ST      Output1


        LDN     Input1
        OR      Input2          NOT Input1 OR Input2 will set Output2
        ST      Output2


END_PROGRAM
```

Figure 10. Example of IL source code

2.5 HMI

The Human Machine Interface is the interface in which an operator receives data and communicates with a machine, robot, or other process. The most obvious use for an HMI is to present data regarding the process, or system, on a screen for the operators to see. In addition to presenting data, its main purpose is often to take action and control the system in case of an error or divergence in the normal process flow. [9]

HMI's can exist in a number of different shapes and implementations. A common implementation is a stand-alone screen without a CPU connected to a PLC or other control unit. The screen is often controlled through a few buttons next to it, but touch screens are emerging and becoming less of an anomaly. Depending on the hardware available, the screen can range from a smaller black and white screen with simple text prompts to a bigger full HD screen capable of showing videos, 3d-renderings, and web pages. The HMI is usually found in close proximity to the process, for instance, it could be built into an electrical cabinet. But as technology progresses, solutions with hand-held HMIs wirelessly sending data and information are becoming more common. [9][10]

A developer can construct an HMI in a wide arrange of ways depending on the hardware used and for what process different programs and development tools will be used. A lot of newer HMIs, like the ones manufactured by ABB, Siemens, and B&R, use a graphical tool when developing the HMI. Using a graphical tool makes it easier to develop and gives a better overview over the HMI compared to solely coding in a text based language. [9]

2.6 Modbus TCP

Modbus is a communications protocol commonly used for communications between different devices in industries like automation and manufacturing as a whole. Modbus uses a master/slave configuration where, in our case, the PLC will be the master and the TSA will be in slave configuration with the address one. The serial speed is set at 9600 baud. The format is as follows; [11]

- 1 start bit
- 8 data bits
- 2 stop bits

There are various values that we want to read from the TSA using Modbus. A cyclic request to read the register 2820 from the TSA is sent from the PLC via Modbus. The register 2820 is a big holding register where 8 other registers from the TSA can be placed. So, the 8 most interesting registers from the TSA is put here and then received in the PLC.

2.7 EINT

EINT, Emotron integer, is an integer that is used only by Emotron products. Emotron made EINT so that very big, very small, positive, and negative numbers could be used. A big problem with EINT is that nothing but Emotron products understands it. For example, if it is 27.8 degrees Celsius in the soft starter, the PLC will see it as 63766. This is how it works:
63766 in UINT translates to 1111 1001 0001 0110 in binary, and it is in this form we can understand EINT. The most significant bit (bit 15) says that if it is an EINT or a regular INT. Bit 14 to 11 is the exponent e that goes from -8 to 7, bit 14 shows if the exponent is positive or negative. Bit 10 to 0 is the value from -1024 to 1023 where bit 10 shows if the value is negative (bit10 set to 1) or if it is positive (bit 10 set to 0). A illustrated explanation of EINT can be seen in figure 11. [12]

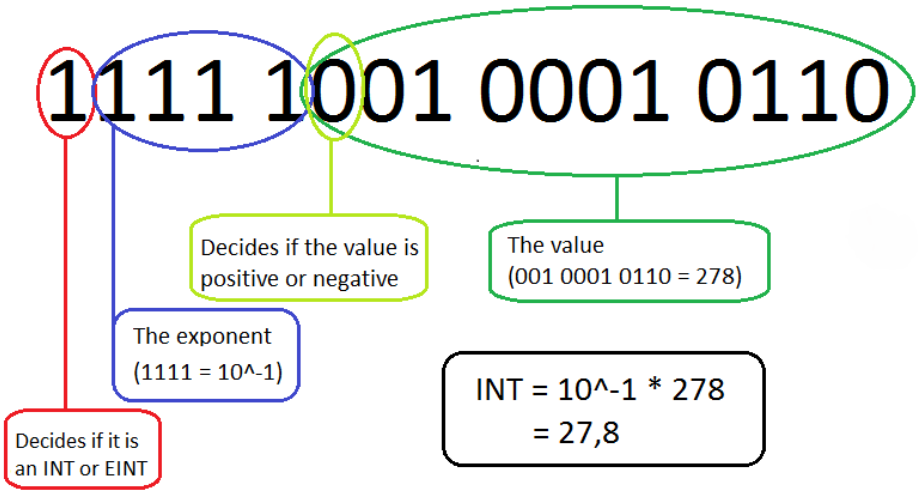


Figure 11. Explanation of EINT

3 Methodology

The initial plan was to go with a more modern approach and to work with smaller iterations over and over again. In practice, this means that instead of the conventional waterfall, with a big planning phase followed by another big design and development phase and lastly a final evaluation phase, we would use smaller iterations containing all these phases. However, this was only used a handful of times and most often so when bigger problems occurred. More often than not, the older waterfall model was used. Even if this was not something that was intended, it just occurred naturally. When one part or problem in the project was finished, it was natural to move on to the next one similar to the waterfall model. [13]

The normal procedure for dealing with a problem was the following:

- Test what is wrong a couple of times so it is not just a one-time bug
- Restart the system.
- Try some different approaches to solve the problem.
- Read through the help section of Automation Studio if the problem was software related.
- Google the error looking for information regarding how to solve the problem.
- Ask someone in the company or a teacher for help.
- Contact the support from the company responsible for the product where the problem exists.

This procedure proved effective and all the problems were eventually solved this way. Although contacting support centres was slow, and the response time for support tickets was often several days. In addition, discussing amongst ourselves proved effective when solving harder problems.

3.1 Source criticism

The three main sources used during this thesis are books, product information from manufacturers, and websites.

Most of the books used as sources were recommended by a lecturer as course materials throughout the education. We also used Lunds Universitys searching site for books, articles and journals. Other similar sites where Lunds Univerity students have access to the books, articles and journals, such as *Science Direct*, http://www.sciencedirect.com.ludwig.lub.lu.se, were used as it counts as

reliable. During training of Automations studio and mapp View we received several books written by B&R Automation, these books were also used as references.

Information about the components came directly from the manufacturers. Such information could come from the user manual, product brochure, or technical catalogue.

Only 3 websites were used as references during this thesis. Many websites do not tell who has written the information and it is hard to trust the information. The web sites used as references have contact information to the owner of the website. The information from the websites used as references was also checked with other sources.

4 Hardware

4.1 B&R's X20 CPU (PLC) and Power Panel T30

In this project, B&R Automation's PLC X20CP1382 and panel 6PPT30 are used. The PLC is a smaller PLC with 3 terminal blocks for I/Os. It can be seen in Figure 12. As can be seen in the aforementioned picture, it has somewhat sparse connections, but it is sufficient for this project. It contains 256 MB DDR3 in RAM memory as well as 2GB Flash memory acting as the program memory. The processor is a Vx86EX with 400MHz clock speed. The Ethernet connector will be used to connect to the panel while the Powerlink is reconfigured to Modbus in order to communicate with the soft starter. [14]

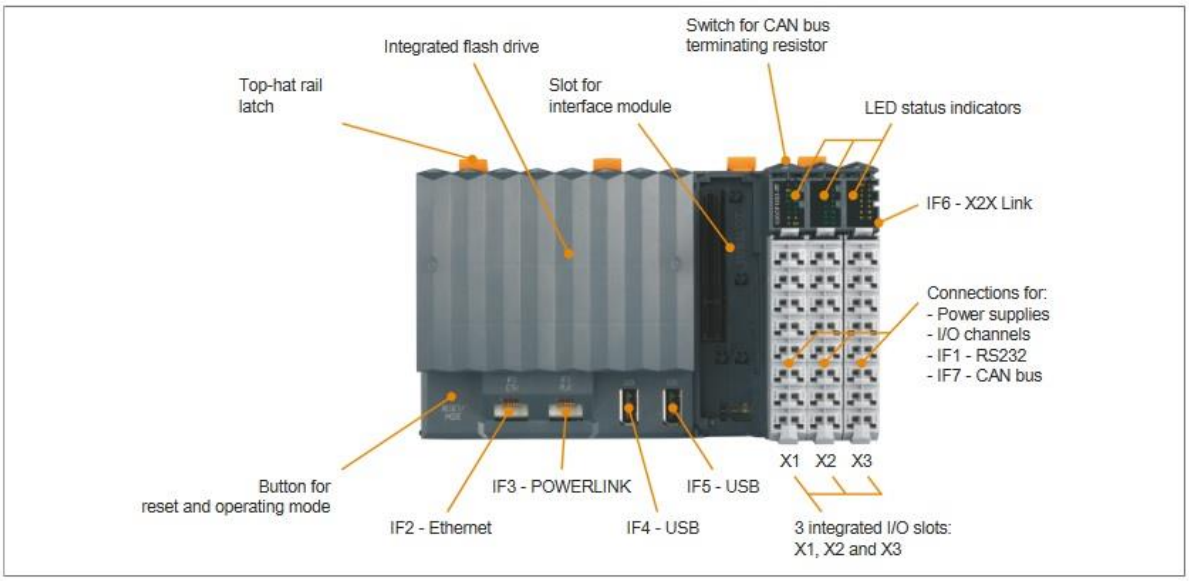


Figure 12. B&R's PLC 20CP1382 [14]

The 6PPT30 panel is a 7-inch analogue resistive touch screen and can be seen in Figure 13. It has two Ethernet connectors as well as two USB 2.0 interfaces. It runs on an ARM Cortex A8 1GHz processor unit with 512 MB Flash memory and 256 MB DRAM memory. The PLC and the panel are CE and UL certified. One of the Ethernet connectors is used to communicate with the PLC, while the other is normally free but used when developing in order to communicate with a laptop. [15]

Figure 13. B&R's 6PPT30 Panel [16]

All B&R's products use the same development environment which is a big advantage. The development environment for B&R products is called Automation Studio. It is an integrated software package for motion, control, visualization, and safety control systems.
In Automation Studio, it is possible to both simulate your program and upload it to a hardware such as a PLC or a panel. [17]
The layout and functions of the panel are programmed with the help of mapp View. Mapp View is an integrated system in Automation Studio for HMI applications. It makes the implementation of the HMI easier by using drag and drop widgets for the design and functions. The HMI can be viewed at all times as a simulation in Google Chrome which helps the developer, even if the panel is not present. This is possible because mapp View is based on web technology. More information about mapp View can be found later in the mapp View section under PLC implementation. [18]

4.2 Emotron TSA

Emotron TSA is the most advanced of CG's soft starters. It will be referred to in this report simply as "TSA" or "the TSA". The version used in this project is the TSA52-042, rated for 22kW at 400V as well as 42 Amps in a normal start. [19] An unmounted TSA can be seen in figure 14.

Figure 14. An unmounted TSA 52-042. [1]

The TSA has a built-in load monitor where you can display the load either directly on the soft starters display or on an external display via analogue or Fieldbus communication.
The TSA also has built in bypass contactors. When the motor is at full speed, the current will not be flowing through the soft starter windings but through the bypass contactors. The advantage of this is that there will be no heat produced through energy losses in the soft starter and it can easily be locked in a closed environment.

The TSA also features a control board with some simple logical operators and a smaller CPU. This makes it possible to control the TSA through an interface called PPU (Portable Presentation Unit). It is a simple 2x16 character single colour display with 9 buttons. A range of settings can be set up and modified in this interface. The most important ones are motor data, relays, ramp times, alarms for various errors, as well as digital in and outputs. The PPU also makes its simple to remotely control the soft starter, for instance from a PLC

like in this project. The TSA with its control board uses closed loop control for the ramping, as discussed in the Technical Background at page 5. [20]

4.2.1 Soft starter set up

All of the parameters needed to be set in the soft starter can be viewed in Table 2. The soft starters PPU is used to set the parameters.

Software settings	Parameter in PPU	Value	Comment
Motor Data	220-227	See motor label	
IP-adress	2651	192.168.240.123	
Subnet mask	2652	255.255.0.0	
Relay's	550-552	Relay: 1-FWD; 2-BCK; 3-Trip	To set forward-, backward rotation and to trip for alarm
PTC alarm	2331	Hard Trip	To activate the PTC alarm
Run/stop control	2151	Remote	To be able to control start and stop externally (PLC)
Level/edge	21A	Edge	To start/stop by a pulse and not continues
Modbus	2820-2825	See table xx	See "Modbus TCP"

Table 2. Set ups for soft starter.

4.3 Electrical cabinet

The electrical cabinet used is a Rittal IP66. The IP-code is a classification of a product's degree of protection against dust, solid foreign bodies, and water. The first 6 in IP66 stands for “dust proof”, which means that the cabinet doesn't let in any dust and is completely protected from contact with foreign bodies. The second 6 in IP66 stands for “protection from powerful water jets”, which means that the cabinet can be exposed to water from any direction with a powerful water jet without letting water in. [21] However, in the current application for the client’s agitator, a hole was made in the cabinet to make it possible to mount the panel, which compromises the cabinet’s protection. If correctly mounted, the hole in the cabinet led to a downgrading of the IP-code of the cabinet from IP66 to IP65. [15] This means that the cabinet can only resist a moderate water jet instead of a powerful water jet. In this case, where

our product will be used, the cabinet will not be exposed to powerful water jets and therefore this downgrading will not affect the product.
A picture of the cabinet can be viewed in figure 15 and the electrical schematic can be seen in figure 16 and appendix 2.

4.3.1 List of cabinet content

- B&R X20CP1382 CPU (PLC)
- B&R PPT30 7'' Power Panel
- Emotron TSA Soft starter
- Chinfa DRAN30-24A VAC/VDC power supply
- EATON DILM65 contactors
- EATON automatic fuse
- Cylec main switch
- Omron relay's 24V
- Ethernet cables
- Wires
- Connection blocks

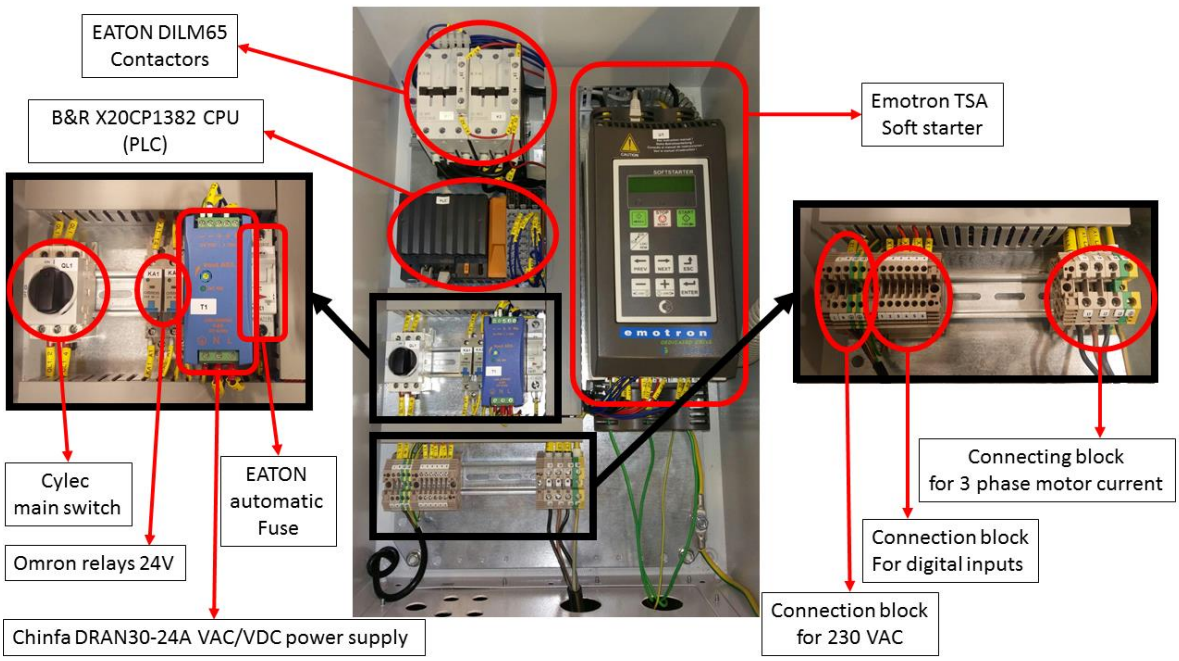


Figure 15. Picture of the components in the electrical cabinet.

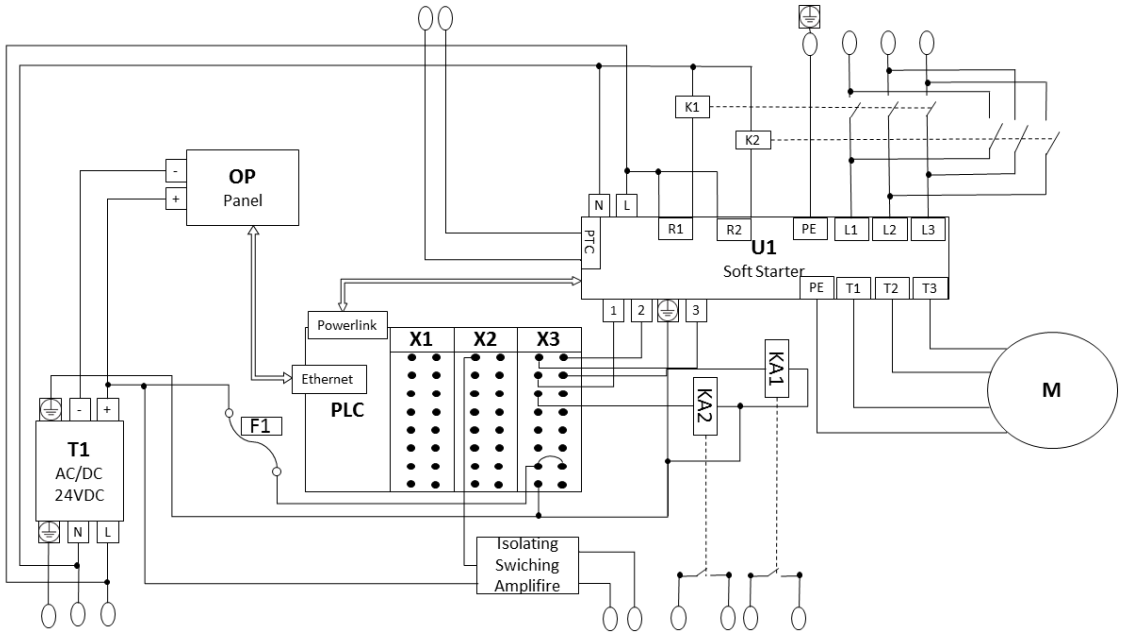


Figure 16. Electrical schematic of the cabinet, appendix 2.

4.3.2 Contactors

To be able to rotate the motor backwards, the soft starter needs external contactors. The TSA has built-in relay to control the conductors so there is no need for specific PLC code to control the contactors. [12] From a safety perspective, this is a big advantage as the soft starter never allows the relays to be closed at the same time and short-circuiting will be prevented.

5 PLC Implementation

5.1 Connectivity

One of the bigger problems early on was that it was not possible to get the PLC online and, consequently, to transfer anything to the PLC. After trying different approaches, it became clear that there were two separate issues; the runtime and IP-addresses. After changing the IP-address on the PLC to match the laptop's IP-address, the PLC was able to go online. Once online, a secondary issue that appeared was that every time a different laptop was used, the IP-address had to be changed in the PLC and the panel. The solution was to change the IP-address on the Ethernet output on all the laptops to be the same. However, this leads to problems when connecting to other networks and the IP-address must be set to automatic again to avoid problems. The runtime version [22], which is the library that the compiler in the PLC uses, also had to be updated on all computers and transferred accordingly to the PLC. After that, the PLC could connect to both the panel and the computers, and the internal communication worked.

5.2 Setup of the Modbus communication

To set up the Modbus communication, the power link output was configured to be an Ethernet output open for Modbus TCP. To be able to read and write to the soft starter the function ModbusTcp_any were used. In the ModbusTcp_any, blocks are used to read and/or write to different parameters by simply setting the first Modbus address and how many parameters to read and/or write to/from. The ModbusTcp_any will then sort the values to different inputs/outputs which could be bound to the right variables.

Inputs to PLC by Modbus	Modbus Address	Logic Modbus Address	Comment
Current	42821	2820	EINT
Shaft power-watt	42822	2821	EINT
Shaft power - %	42823	2822	UINT
Relay status	42825	2824	Relay 1: forward; Relay 2: Backwards; Relay 3: Alarm; UINT
Alarm code	42826	2825	PTC alarm code = 2

Table 3. Modbus addresses from soft starter.

5.3 Programs

5.3.1 Introduction

Three languages have been used in this project, they are: FBD, LD and ST. The main language is FBD, but in some cases LD or ST have been more suitable options.
Throughout the project, seven different programs were used. These are; RelayStatus.fbd, EINT.st, PowerCurrent.fdb, StartStop.fbd, Cleaning.fdb, Alarm.ld and water.fdb.
In RelayStatus.fdb, the bits from the parameter “Relay Status” in the soft starter get separated. The parameter, “Relay Status”, is to see if the motor is going forward, backward, or if there are any alarms. Relay 1 (bit 2 from the Modbus) is forward rotation, Relay 2 (bit 1 from the Modbus) is backwards rotation, and Relay 3 (bit 0 from the Modbus) is activated if any soft starter alarms occur. All the roles for the water trap are implemented in Water.fbd. It is a basic program where the output, filling up the water, gets TRUE if the water trap sensor is TRUE or if the operator presses the fill-up button. It is also possible to run water filling sequences in timed intervalls. The program stops filling up water in two cases; when the operator presses the stop button or a certain time, chosen by the operator, has passed. A timestamp is also set to be able to see the last time the tank was filled with water.
The StartStop.fbd program collects all the data from the other programs showing when to start and stop the agitator and sends out the pulse to start or stop the agitator.
The program PowerCurrent.fbd is used to convert power and current from the soft starter from EINT to INT. The Function block of use, EINT_TO_INT, is a reference to the program EINT.st.
The three extended programs, cleaning, alarm and EINT, will be explained further down.

5.3.2 Cleaning

The cleaning of the agitator blades can be done in three ways; manually, automatically and/or time-intervals. The blades are cleaned when the motor is rotating in the other direction so dirt and clogging falls of the blades.

When a cleaning sequence is initiated, the program first checks if the motor is running forward or standing still. If it is standing still, it will begin to rotate backwards by triggering the digital output from the PLC that leads to the soft starter’s digital input which is coded to start the backwards rotation with the help of the contactors. On the other hand, if the motor is running when a cleaning sequence is initiated, the motor first stops, and when the motor power equals 0W, the backward rotation can begin with a soft start. The cleaning sequence lasts for a determined number of minutes which can be set through

the HMI. After the timer for the cleaning time is triggered, the motor once again stops, waits until the power is 0W and then begins with a forward rotation with a soft start.

When a manual cleaning sequence is triggered, an operator simply pushes a button in the HMI, starting the sequence. The automatic trigger is constantly checking if the torque, or power, that the soft starter is working at does not exceed the allowed number. This allowed number is set through the HMI. If the power percent exceeds or is equal to the allowed number, the cleaning sequence described above is initiated. The third option for running a cleaning sequence is by time-intervals. When time-intervals is active a cleaning sequence will run every 24 hours for example, starting when an operator activates time-intervals and sets the time. Automatic cleaning and time-intervals can be used together. There is also a filter time that can be manually set. This means that the power percent has to exceed the allowed amount for a certain time, the filter time, in order for the cleaning sequence to trigger. This is to stop sudden, very short, power spikes or interference from triggering the cleaning sequences.

The cleaning sequence, the help triggers, and programs around it were all written in functions blocks. It would, however, have made a lot of sense to write in SFC, since it is strictly sequences being run. But since FBD is easy to overview and common that was used instead.

5.3.3 Alarm

Alarm-handling is done through an extension in Automation Studio called mpAlarmX. Like mapp View, it is an extension based on mapp technology. It uses a function block called mpAlarmXCore as its base block. Thereafter, actions concerning the alarms, are performed with different blocks. For instance, if an alarm is triggered, the function block called mpAlarmXSet is enabled and the alarm is displayed using the mpAlarmXListUI that is bound to a widget. Each one of these mpAlarmX blocks are referenced to the central mpAlarmXCore block.

To be able to display the alarms when navigating away from the alarm page, caching must also be turned on in advanced options in a visualization file on the PLC.

5.3.4 EINT to INT

One of the harder problems encountered was the conversion of the received EINT from the Modbus communication with the TSA to a normal INT. Since there was no converter readily available, one had to be constructed as a part of the project.

The final conversion solution works as follows:

Initially, there were failed attempts at creating the conversion with function blocks. But since the conversion was such a number complex task, it was easier to do in structured text. The code for the converter can be seen in Appendix 3. Firstly, the MSB is checked to see if the number to be converted is an EINT at all. If not, the value can be used as a normal INT. Since EINT uses two-complement, the eleventh bit is checked to see if the number is positive or negative. Thereafter, the entire number is right shifted 11 steps to remove everything but the MSB and the following four numbers which represent the exponent. These numbers are placed in the variable “extracted” and checked with the corresponding values for the different exponents, which range from 10^-8 up to 10^7. The last step is to simply calculate the INT value of the number by masking out the last 11 bits with the logical operation AND, and then multiplying it with the correct exponent. If the number is in the negative loop, checked at the beginning, it is multiplied with minus one. The final INT number in its correct form is then stored in the variable “Float”.

5.4 HMI

5.4.1 Introduction

The goal of the HMI is for the operator to be able to see the status of the agitator, start a cleaning sequence, see and acknowledge the alarms, as well as control the filling of water traps.

The status of the agitator includes current, as well as shaft power, in percent and watts. In addition, it also includes the status of the motor, if it is running, and in which direction. The status of the motor is displayed with easy-to-understand symbols to make the perception as efficient as possible. The other data is presented in clear text inside big boxes with the corresponding unit next to it, with the exception of the power percentage, which is shown with a bar. All the input and navigation is made available through big, clear, grey buttons that work through OPC-UA and event-bindings. The concept of event-bindings will be explained later.

The HMI was designed through an extension in Automation Studio that is called mapp View.

5.4.2 mapp View

mapp View is a mapp component built upon the mapp technique which means that it is a modular extension that fits into and works with all of B&R’s products. [17] [18]

The page content in mapp View is largely done with drag and drop and pre-made widgets. These pages are then defined in an overhead layout along with all the different areas of the page. The pages and the layout are then referenced in a Visualization file which is what we download and run on the PLC. [18]

Every page, along with its overhead layout structure, is then converted to a file with XML structure, similar to web pages. It is therefore possible, and sometimes necessary, to manually write and edit the code in XML. For instance, when creating a button, it can be done with a widget using drag and drop. But what happens when you click the button, the so-called event, has to be defined in the XML code. [18]

The layout for the pages is constructed in such a way that only the information in the middle page changes when the operator navigates to a new page. The menu, the logo, and the alarm box are constant and always loaded. [18]

5.4.3 OPC-UA

OPC Unified Architecture is the communications protocol used for most of the data binding. All of the variables in the entire project can be enabled or disabled as an OPC-UA variable. If enabled, they can then be transmitted through the PLC, which acts as an OPC-UA server. In this implementation, there is only one client on the network other than the PLC capable of OPC-UA communication, the panel. For instance, the current in the soft starter is measured by sending a Modbus command to the TSA reading the register which holds the data containing the current. The answer is then stored in a variable which is converted from EINT to INT. Thereafter, the value is in the correct form, and since it is being enabled in the OPC-UA server, any widget, in this case, the numeric output widget, can access and display the correct value. [18]

It is also possible to write to a variable through OPC-UA, however, only one value can be changed at a time and written to a variable. This means that only simple widgets, such as a toggle button and the set numeric value widget, can be used with OPC-UA. In theory, OPC-UA could be used to change multiple values with one click, but this is not supported by Automation Studios widgets. [18]

5.4.4 Events and bindings

To deal with more complex data binding, Automation Studio uses what they call Events. This can either use OPC-UA or use Action based events. OPC-UA events modify previously explained OPC-UA variables. Action based events modify and use function blocks. An event is coded in XML in a special event binding file and contains a source, page reference ID, widget reference

ID, as well as an action triggering the event (for instance a click). Subsequently, one event can trigger several actions. Every action has a target as well as a method. [18]

Every normal button, that is not a toggle button, uses events to do what they are designed for. Since it is possible to define multiple actions with different targets and methods within one event, it is possible to make advanced changes and commands with one click on the panel. However, coding an event takes a lot of time and the syntax and text prompt does not account for errors or missing commas. [18]

6 Result

The result is a finished product. It is a product that is capable of soft starting an agitator, running automatic cleaning cycles on it, as well as handling various alarms that might occur. Amongst the alarms handled, there is a PTC alarm from the agitator, a water trap alarm to stop methane gas from rising from the biomasses that are being agitated and alarms from the soft starter. All this, along with relevant data concerning the process, is presented on an easy to read and manoeuvre HMI. Furthermore, an operator can manually start the cleaning cycles or filling of water traps, as well as control the agitator.

However, it has not been installed at the end customer site yet, and has thus only been tried on a smaller electrical motor.

6.1 Final HMI design

The final design for the HMI can be viewed in figure 17, figure 18, figure 19 and figure 20.

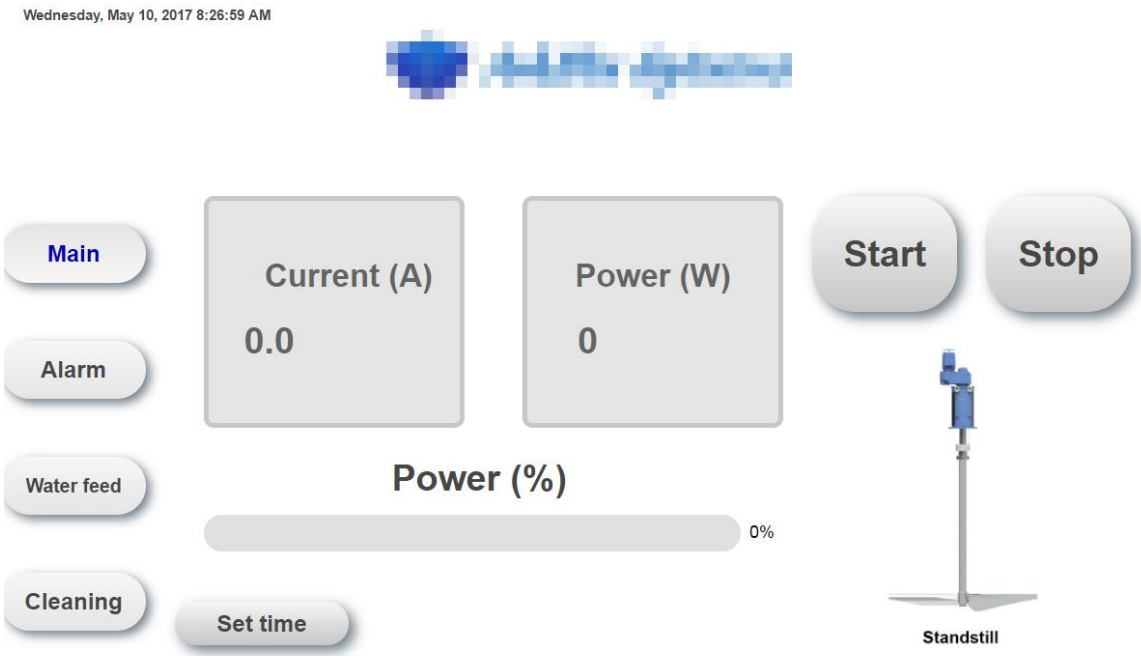


Figure 17. The main page of the HMI.

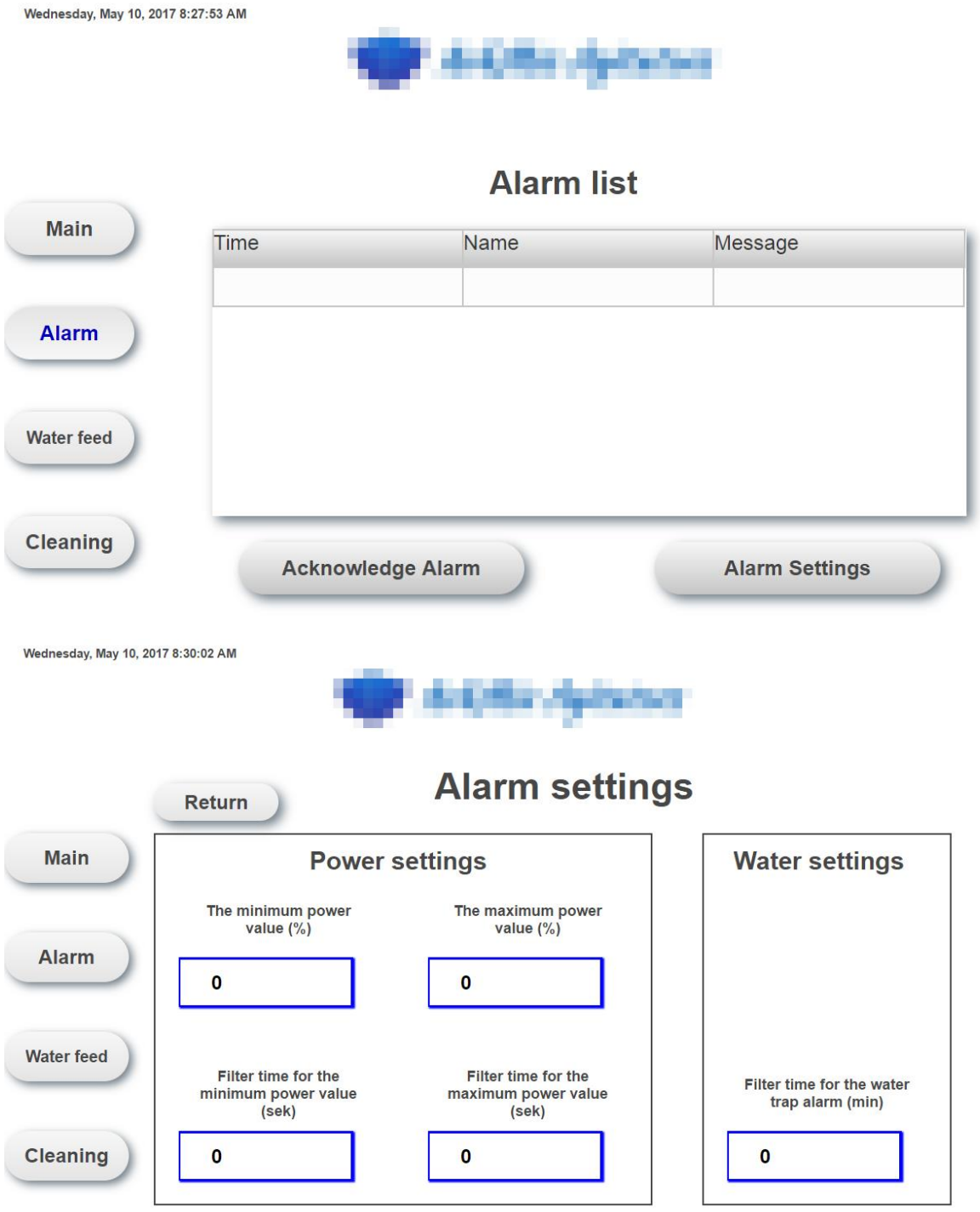


Figure 18. The alarm page and alarm settings page of the HMI.

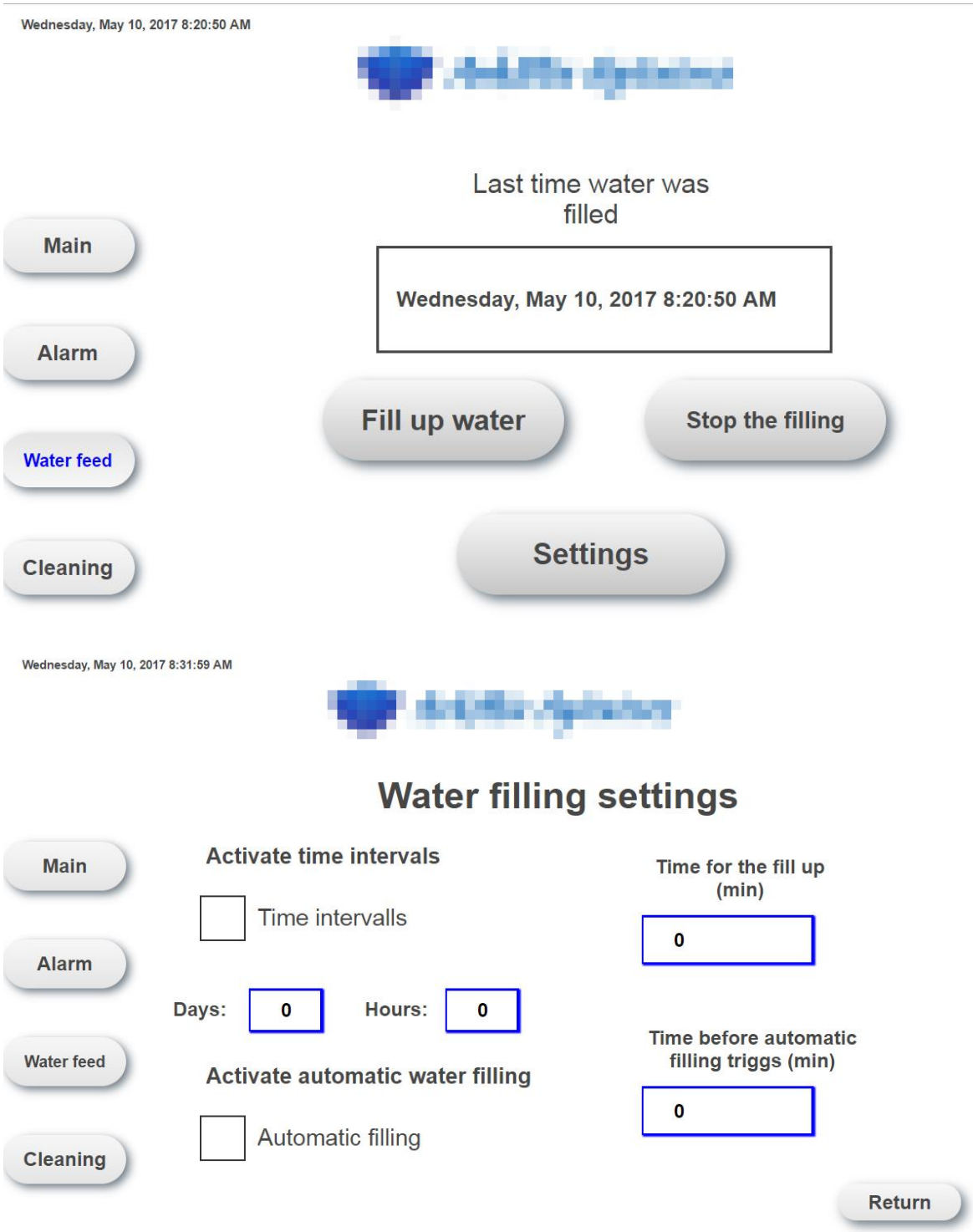


Figure 19. The pages controlling the water filling and water filling settings.

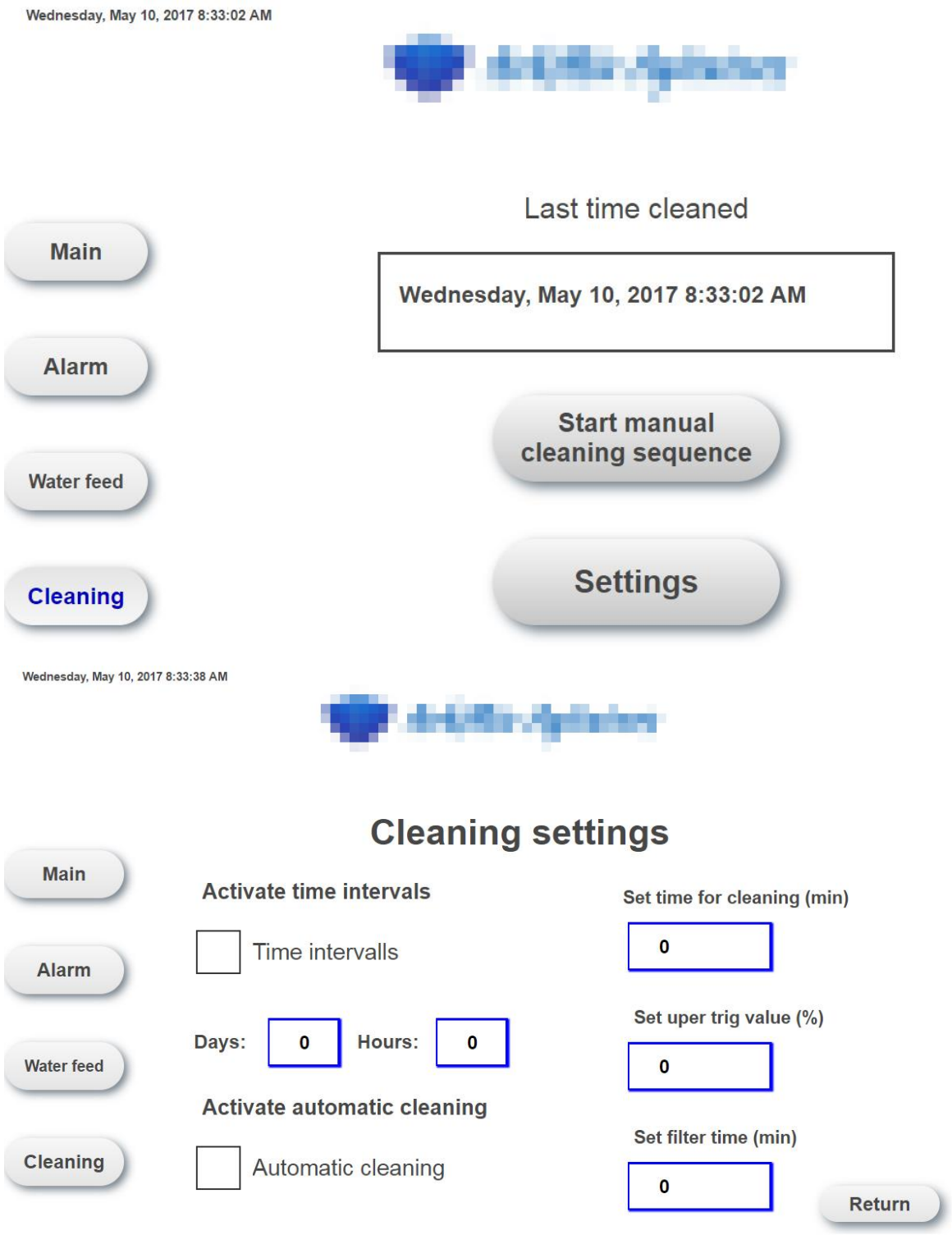


Figure 20. The pages controlling the cleaning sequence and cleaning settings.

6.2 Electrical cabinet design

The electrical cabinet built by the cabinet builders can be seen in figure 21 and figure 22.

Figure 21. Closed cabinet from the outside.

Figure 22. Open cabinet with all the components.

7 Discussion

The project had some problems which either slowed the progress or caused us to seek external help. One of the recurring problems was the lack of communication with the end-customer. Ideally, we would prefer an ongoing discussion with the customer in regard to not only specifics about the specification for the product, but also about questions such as how the design should be implemented. Since there was little to no communication, some of the implementation choices that weren't really clear in the initial specification provided had to be improvised, and a lot of the time we found ourselves having to decide what we thought would work best for the customer.

The project came with a number of technical difficulties. The early connectivity issues are mentioned above, under the PLC section, but even once those were solved we couldn't set any of the digital in our outputs. After consulting with the support centre, it turns out that one of the outputs in the corner wasn’t an output; it was +24VDC and it had to be yoked over with a cable to another connector next to it in order to electrify the I/Os to be able to use them. This information was in a paragraph inside a thousand-long page help centre, but was supposed to be common knowledge for the experienced PLC-programmer. This helped us realise that when we are making our manual, we shouldn't take something that we consider simple for granted, as we never know what kind of background the operators might have.

Furthermore, there are some really important settings that were hard to find, and Automation Studio turned out to be a very advanced and impressive tool, but not so user-friendly. For instance, when we wanted to set up a cache to store alarms that hadn't yet been acknowledged, we had to do so in the advanced settings of a minor file on the PLC controlling the visualisation part of the programs. Since anyone that uses an alarm system wants to cache their alarms, if the operator happens to switch away from the alarm page the alarms should still be visible. Therefore, we found it strange that is was so well hidden.

7.1 The development process

Working with B&Rs Automation Studio has, over all, worked very well. It is a very new and modern development environment which is always welcome. It boasts very well-designed features for Modbus communication and a well-designed overview for managing big programs, which proved to work very well. But it also had its drawbacks. Being as new as it was, a lot of features and widgets weren't implemented yet and had to be coded manually. Alarm handling is a very common feature in most automation projects, despite this, the widgets for alarm handling weren't ready for “at least a year” according to

the people responsible for the software training. Due to this issue, they had to be implemented manually which proved to be a very time consuming and at times hard task.

7.2 Soft Starter instead of a frequency converter

One of the main functions in this product is the ability for the agitator to run backwards, effectively cleaning itself of dirt attached to the impeller blades. The easiest way to rotate an electrical motor backwards is by using a frequency converter. A frequency converter or a drive, as it is often referred to, would also be able to perform soft starts and soft stops by adjusting the frequency to fitting values. At first thought, it might seem unnecessary to use a soft starter with external contacts in order to run the agitator backwards. But there are plenty of advantages when using a soft starter with external contactors instead of a frequency converter.

The two most important advantages would be price and heat. The price of a 30kW frequency converter is roughly 3 times higher than the prices of a soft starter of the same size. [24] The price of the contactors in the range of 30kW is very small in comparison to a frequency converter. However, if you need a bigger soft starter, for example 250 kW, the price difference becomes much smaller and the main reason for this is that the contactor gets expensive.

Equally important is the heat from power losses. A modern frequency converter has an efficiency of about 97- 98% while running [23]. In 30kW two percent losses means 600W, which is roughly the same power that an external electrical radiators outputs. That leads to a lot more time and money having to being spent on solutions for the cooling of the cabinet and getting the correct airflow through it with the use of fans and air outlets. In contrast, the soft starters run on 100% efficiency while the bypass is connected, since no current passes through the soft starter.

In conclusion, it is cost-effective and perhaps even easier to use a soft starter with internal bypass and external contactors in this case, since the agitator runs on a constant speed during its normal uptime. However, if it is going to have a lot of starts and stops at a higher power level than 30kW it might be worth to investigate in using a frequency converter. Or if for some reason the agitator needs to run at different speeds, we are left with no other choice than a frequency converter.

7.3 Further developments

There are things that can be improved on this product. For instance, the HMI could be designed to be better-looking. There are also some minor bugs with the alarm system where the first alarm triggered since a restart will sometimes

lack a timestamp. The cause for this bug is unknown, but we worked around it by having the end-user do a couple of test alarms when the product is commissioned or has lost power. Since we simply set up an existing alarm, handling extension through Automation Studio, it is very hard to go into detail about the alarm management system and find the reason for this bug. However, given more time this is something we really would like to have solved.

8 Conclusions

The project's main objective was to deliver a finished product on time, fulfilling the specification delivered by the customer: A smart agitator control system with soft starts, capable of monitoring alarms, starting external water trap filling as well as running cleaning cycles. Despite some difficulties, this objective was achieved. The product works, although we never got to test it on the real process it was intended for but only on a smaller test motor.

In order to control specific sequences for an agitator with a PLC programs are written, mostly in function blocks but also structured text. In the programs, various conditions are checked, and if they are fulfilled the specific sequence will trigger and run for the set time. The program is further explained under section 5.3.2.

The HMI and the PLC programs was developed from scratch in Automation Studio. mapp View has made the HMI development easy even for inexperienced developers. The most extensive problem during the HMI development was the lack of feedback from the end customer. All the PLC and HMI implementation are described in chapter 5.

The project's main components are an Emotron soft starter, a B&R PLC, as well as a B&R Panel. These components communicate through Modbus, ethernet and hard wires to I/O’s. The electrical motor is connected to one end of the soft starter and the power supply to the other end. The connections can be viewed in appendix 2.

The cabinet was built by professional cabinet builders but we supplied them with the electrical drawings. The drawing was designed with help of user manuals and the cabinet builders choose the brands and size of the components. The electrical schematics can be viewed in appendix 2.

There are future improvements that could be done, given more time and interest from the market, but nothing very major. Due to extensive documentation and commented source code this should be easy to perform for another developer. In addition, a user manual, appendix 1, was also written for operators and service personal to understand and use the product in a correct way.

We learned a lot from this project, not only about PLC programming and the technical aspects, but also about the work flow at companies as well as some of the obstacles that are common in a work environment and how to overcome them, in our case mainly the lack of contact with the end customer.

8.1 Ethical aspects

There are also some ethical aspects in this thesis. Since the soft starter and cleaning process contributes to power savings, less downtime and a longer lifetime of the motor, this product contributes to a better environment. In order to protect the integrity of the customers their identities are not revealed. Most of the source code is also protected due to confidentiality.

9 References

[1] CG drives and Automation. [ONLINE] Available at: http://www.emotron.se.[Accessed 26 April 2017]

[2] Arkmix. [ONLINE] Available at: http://sv.arkmix.com/nyheter/ [Accessed 27 April 2017]

[3] Williams, B.W. (2006) Principal Elements of Power Electronics: Devices, Drivers, Applications, and Passive Components. Barry W Williams.

[4] Rockwell automations; When to use a Soft Starter or an AC Variable Frequency Drive [ONLINE] Available at: http://literature.rockwellautomation.com/idc/groups/literature/documents/wp/150-wp007_-en-p.pdf [Accessed 27 April 2017]

[5] Jaikrishna V., Alex L.T., Dash S.S., Gachhayat S.K. (2015) Fault Tolerant Soft Starter Control for Induction Motors. In: Kamalakannan C., Suresh L., Dash S., Panigrahi B. (eds) Power Electronics and Renewable Energy Systems. Lecture Notes in Electrical Engineering, vol 326. Springer, New Delhi

[6] Bolton, W. (2009). *Programmable logic controllers*. 5th ed. Oxford: Newnes

[7] Amci. [ONLINE] Available at: https://www.amci.com/industrial-automation-resources/plc-automation-tutorials/what-plc/[Accessed 24 April 2017]

[8] Sharma, K.L.S.(2011). Overview of industrial process automation. Elsevier.

[9] Katzel, J. (2004). *HMIs in robotics*. Estados Unidos: Cahners Publishing, Vol 51.

[10] Hansson, M. (2014). *HMIs embrace the future, deliver performance*. Plant Engineering, Vol. 68

[11] The Modbus Organization. [ONLINE] Available at: http://www.modbus.org/ [Accessed 2 March 2017].

[12] CG drives and Automation. [ONLINE] Available at: http://www.emotron.com/globalassets/downloads/products/softstarters/emotron-tsa/emotron-tsa-instruction-manual/emotron_tsa_softstarter_manual_01-5980-01r2.en.pdf [Accessed 25 June 2017].

[13] Balaji, S., & Murugaiyan, M. S. (2012). *Waterfall vs. V-Model vs. Agile: A comparative study on SDLC*. International Journal of Information Technology and Business Management, Vol.2 No.1

[14] B&R Automation [ONLINE] Available at: http://www.br-automation.com/downloads_br_productcatalogue/BRP4440000000000000493282/X20CP13xx-RT-ENG_V1.16.pdf [Accessed 26 April 2017].

[15] B&R Automation [ONLINE] Available at: https://www.br-automation.com/en-au/products/visualization-and-operation/power-panel-t-c-series/power-panel-t-series/power-panel-t30/6ppt300702-20b/#techdata [Accessed 26 April 2017].
[16] B&R Automation [ONLINE] Available at: https://www.br-automation.com/en-au/products/visualization-and-operation/power-panel-t-c-series/ [Accessed 26 April 2017].
[17] B&R. (2016). *Working with Automation Studio*. 2st ed. B&R Automation.
[18] B&R. (2016). *Working with mapp View*. 1st ed. B&R Automation.
[19] CG drives and Automation. [ONLINE] Available at: http://www.emotron.se/globalassets/downloads/products/softstarters/emotron-tsa/emotron-tsa---product-brochure/emotron_tsa_engelska-low.en.pdf .[Accessed 26 April 2017].
[20] CG drives and Automation. [ONLINE] Available at: http://www.emotron.com/globalassets/downloads/products/softstarters/emotron-tsa/emotron-softstarters-technical-catalogue/emotron_softstarter_technical-catalogue_01-5552-01r2.en.pdf [Accessed 25 June 2017].
[21] Mitutoyo UK. [ONLINE] Available at: https://www.mitutoyo.co.uk/media/pdf/Small-Tools/IP-Codes.pdf [Accessed 5 April 2017]
[22] B&R. (2016). *Automation Runtime*. 2st ed. B&R Automation
[23] CG drives and Automation. [ONLINE] Available at: http://www.emotron.se/globalassets/downloads/products/ac-drives/emotron-fdu/emotron-fdu--vfx-2.0-technical-catalogue/emotron_fdu-vfx2-0_technical_catalogue_01-4948-01.en.pdf [Accessed 26 April 2017].
[24] Interview with Mikael Mårtensson, head of the project department at CG Drives & Automation [13 April 2017]

Appendix 1, User manual

List of contents

1. The main page

At the main page, figure 23, you can control 3 things:

1. Start the agitator.
2. Stop the agitator.
3. Set the time for the panel and PLC.

You can also monitor the power and current as well as see in what direction the agitator is rotating.

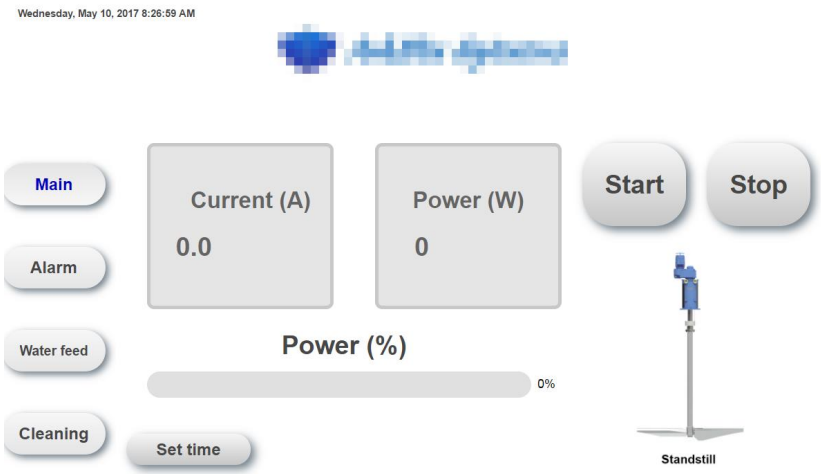


Figure 23. Main page

2. Cleaning

At the cleaning page, you can start a manual cleaning sequence as well as see when the agitator was last cleaned. To start a manual cleaning sequence just set the time for how long you want to clean and the press “start cleaning sequence”. If you want to stop the cleaning sequence just go to the main page and press “stop”.

At the settings page, figure 24, you can choose If you want system to do the cleaning for you, simply set automatic cleaning, trig level and filter time and system will clean when needed. Filter time is the time it takes from the trig

value is reached until the cleaning starts, this is to avoid cleaning because of spikes in the monitor.

It is also possible to clean with time-intervals, simply enable the option and set the time, is the time is set to 24 hours, a cleaning sequence will be run after 24 hours, and then again every 24 hours. **The time it takes for the cleaning sequence to run is counted into the time-intervals. This is important to note when running longer cleaning sequences.**

2.1 FAQ:

Does the cleaning not work? Check if there is a set cleaning time, it does not work with the standard 0 minutes.

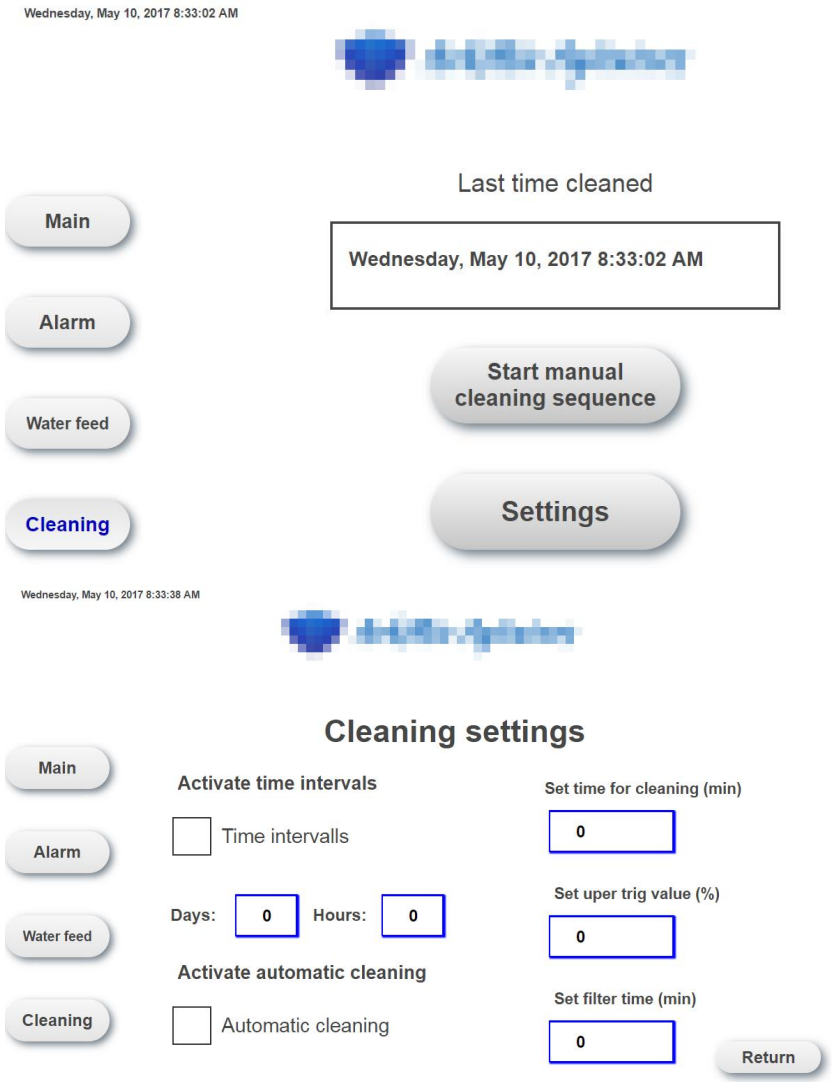


Figure 24. Cleaning page and cleaning setting page.

3. Fill up water

At the water page, figure 25, you can firstly set the settings for the water filling. To start the water filling simply press fill up water button and if you for some reason want to stop it you simply press stop. In the settings page the automatic filling can be turned on or off. This means that when the water trap alarm triggers, the water filling will begin for however long is set. A filter time can also be set, meaning that the water trap alarm has to be triggered for a set amount of time before the automatic water filling begins. This is to avoid water filling due to sudden short spikes or interference.

Time-interval water filling is also an option, simply enable it and set the time. If 24 hours is set, the water will be filled in 24 for hours for however long time is set in the "time for the fill up" box, and then again in the next 24 hours and so on.

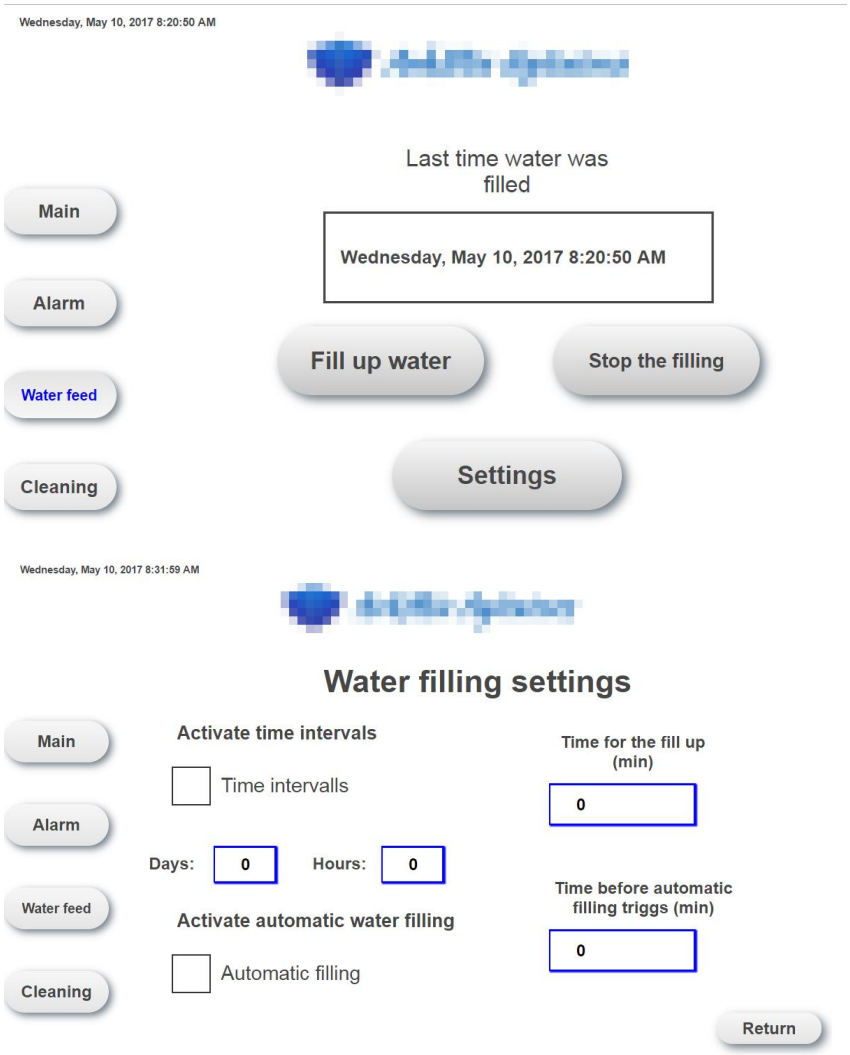


Figure 25. Water feed page and water filling settings page.

4. Alarms

The alarms are managed at the alarm page, figure 26, here the time for the alarm, the name of the alarm and sometimes additional information is shown in a table. To acknowledge an alarm simply select it and press the "Acknowledge Alarm". Before commissioning try setting some test alarms manually through a PC to check that the alarm handler works as intended.

Through the settings, the values for the power alarms and the water trap alarm can be set. Again, with a filter, if the filter is not wanted, simply set the time to zero.

4.1 FAQ:

Are you trying to acknowledge an alarm that isn't disappearing? Then there is probably still an active alarm signal, correct the reason for the alarm signal and then acknowledge it again.

Is the soft starter alarm triggered without you knowing why? Open the cabinet and check the error code in the PPU, then check the page 130 in the TSA manual to see what the number means and how to fix it.

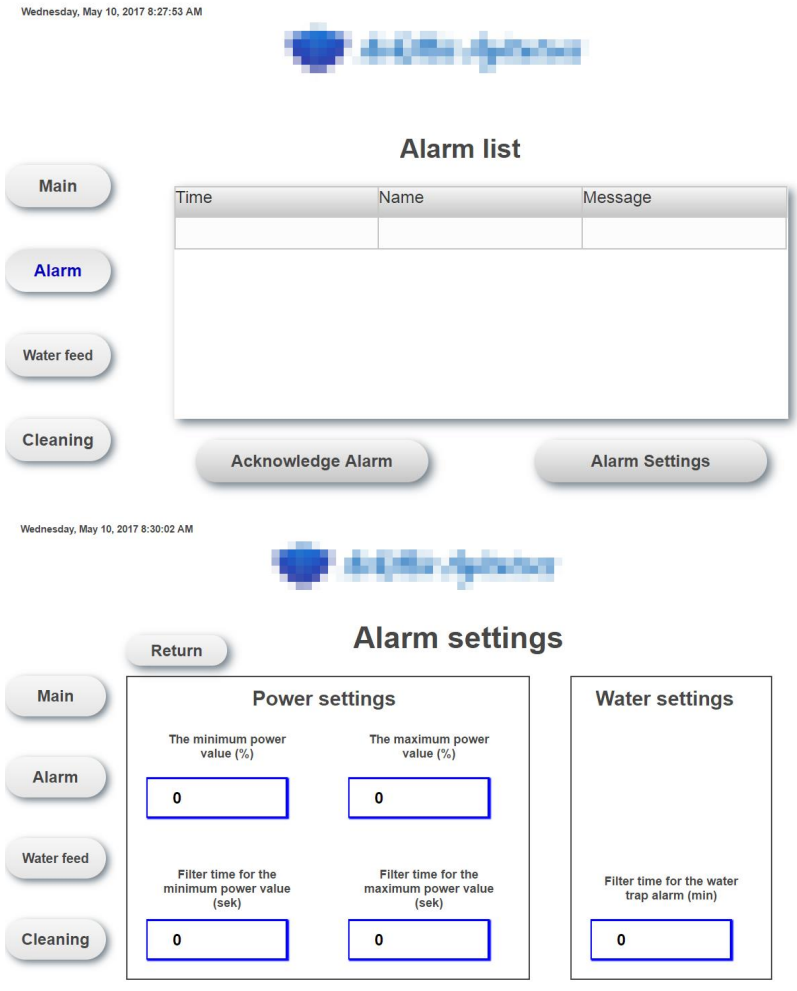
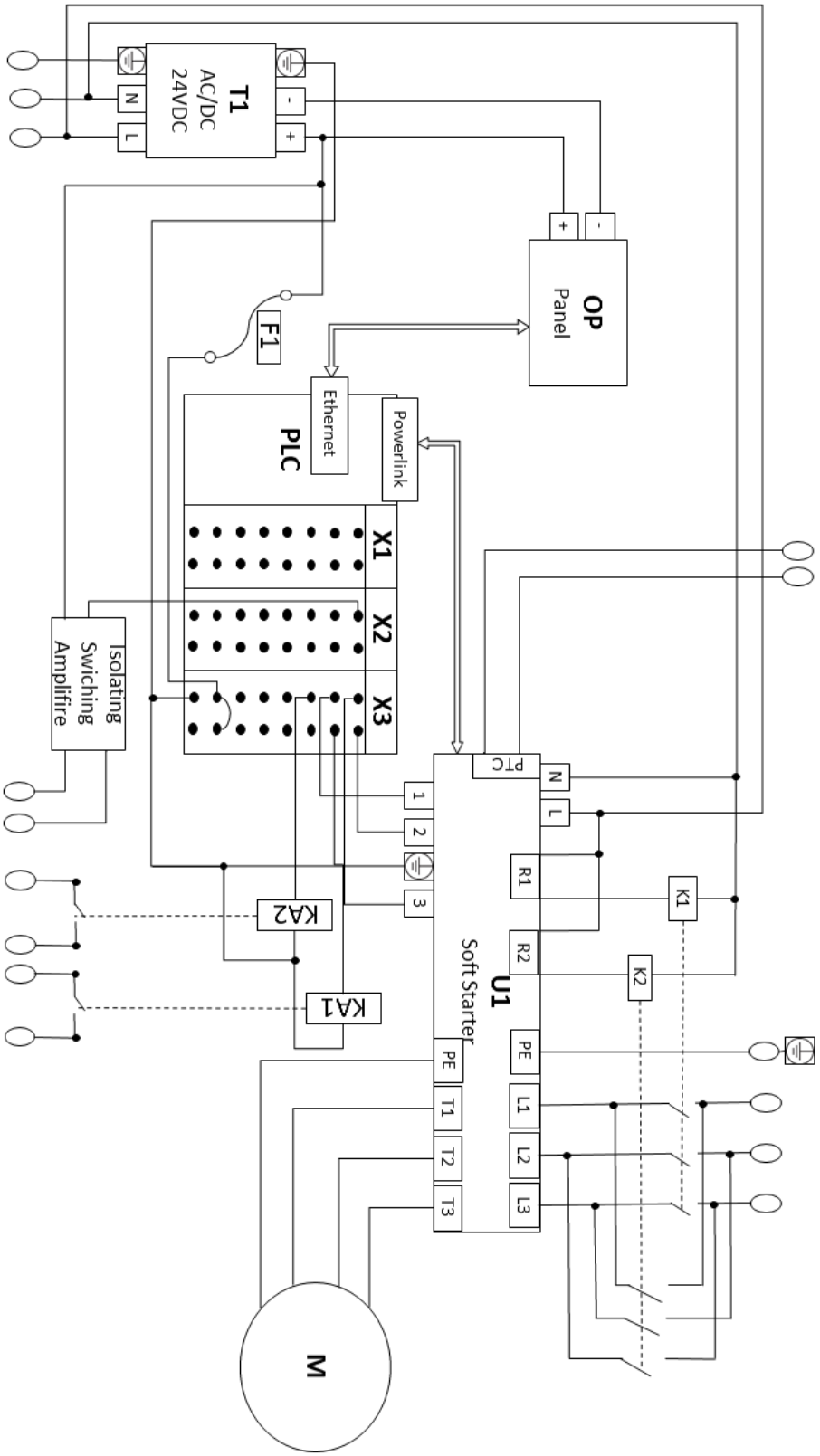


Figure 26. Alarm page and alarm setting page.

Appendix 2, Electrical schematic



Appendix 3, EINT source code

```
PROGRAM _CYCLIC

        IF(input4> 32767) THEN     //Checks if MSB is 1
                Extracted:= SHR(input4,11); //Right shift 11 steps
                IF ((input4 AND 1024) = 1024) THEN
        // extract the exponent and checks which exponent for all negative Values
                        inv:= NOT input4;
                        val:= inv + 1;

                        IF (Extracted= 31) THEN                 // 10^-1
                                Float := 0.1 * (val AND 2047);

                        ELSIF (Extracted=30) THEN               // 10^-2
                                Float := 0.01 * (val AND 2047);

                        ELSIF (Extracted=29) THEN               // 10^-3
                                Float := 0.001 * (val AND 2047);

                        ELSIF (Extracted=28) THEN               // 10^-4
                                Float := 0.0001 * (val AND 2047);

                        ELSIF (Extracted=27) THEN               // 10^-5
                                Float := 0.00001 * (val AND 2047);

                        ELSIF (Extracted=26) THEN               // 10^-6
                                Float := 0.000001 * (val AND 2047);

                        ELSIF (Extracted=25) THEN               // 10^-7
                                Float := 0.0000001 * (val AND 2047);

                        ELSIF (Extracted=24) THEN               // 10^-8
                                Float := 0.00000001*(val AND 2047);

                        ELSIF (Extracted=17) THEN               // 10^1
                                Float := 10 * (val AND 2047);

                        ELSIF (Extracted=18) THEN               // 10^2
                                Float := 100 * (val AND 2047);

                        ELSIF (Extracted=19) THEN               // 10^3
                                Float := 1000 * (val AND 2047);

                        ELSIF (Extracted=20) THEN               // 10^4
                                Float := 10000 * (val AND 2047);
```

```
                ELSIF (Extracted=21) THEN               // 10^5
                        Float := 100000 * (val AND 2047);

                ELSIF (Extracted=22) THEN               // 10^6
                        Float := 1000000 * (val AND 2047);

                ELSIF (Extracted=23) THEN               // 10^7
                        Float := 10000000 * (val AND 2047);
                ELSE
                        Float := (val AND 2047);
                END_IF

 Float := Float * -1;           // makes the value negative


ELSE                            // The exponet for all positiv values
val:= input4;

                IF (Extracted= 31) THEN                 // 10^-1
                        Float := 0.1 * (val AND 2047);

                ELSIF (Extracted=30) THEN               // 10^-2
                        Float := 0.01 * (val AND 2047);

                ELSIF (Extracted=29) THEN               // 10^-3
                        Float := 0.001 * (val AND 2047);

                ELSIF (Extracted=28) THEN               // 10^-4
                        Float := 0.0001 * (val AND 2047);

                ELSIF (Extracted=27) THEN               // 10^-5
                        Float := 0.00001 * (val AND 2047);

                ELSIF (Extracted=26) THEN               // 10^-6
                        Float := 0.000001 * (val AND 2047);

                ELSIF (Extracted=25) THEN               // 10^-7
                        Float := 0.0000001 * (val AND 2047);

                ELSIF (Extracted=24) THEN               // 10^-8
                        Float := 0.00000001*(val AND 2047);

                ELSIF (Extracted=17) THEN               // 10^1
                        Float := 10 * (val AND 2047);

                ELSIF (Extracted=18) THEN               // 10^2
                        Float := 100 * (val AND 2047);
```

```
                                    ELSIF (Extracted=19) THEN                   // 10^3
                                                Float := 1000 * (val AND 2047);

                                    ELSIF (Extracted=20) THEN                   // 10^4
                                                Float := 10000 * (val AND 2047);

                                    ELSIF (Extracted=21) THEN                   // 10^5
                                                Float := 100000 * (val AND 2047);

                                    ELSIF (Extracted=22) THEN                   // 10^6
                                                Float := 1000000 * (val AND 2047);

                                    ELSIF (Extracted=23) THEN                   // 10^7
                                                Float := 10000000 * (val AND 2047);
                                    ELSE
                                                Float := (val AND 2047);
                                    END_IF

                        END_IF
            ELSE
            Float := UINT_TO_INT(input4); // if it is not a EINT
            END_IF
END_PROGRAM
```