

OPC UA Server

Design of a server-client environment with the OPC Unified Architecture.

Albert Labodia Farreny

Degree Thesis

Bachelor Thesis for Bachelor of Engineering

Degree Programme in Energy Technology

Vaasa, Finland, 2023

DEGREE THESIS

Author: Albert Labodia Farreny

Degree Programme: Electrical Engineering and Automation

Specialisation: Energy Technology

Supervisor(s): Joachim Böling and Philip Hollins

Title: OPC UA server: Design of a server-client environment with the OPC Unified Architecture.

Date: 18.05.2023

Number of pages: 38

Appendices: 5

Abstract

The emerging fourth industrial revolution has created the need to adapt existing technologies and devices towards absolute connectivity. This thesis aims to lay the foundations in Novia UAS for the OPC UA environment, an architecture that allows the connection between devices from different manufacturers.

The thesis framework consisted of one Phoenix Contact PLC acting as the server, programmed using Codesys software, and one Bachmann Electronics display, configured by the Atvise software. The PLC project was designed to efficiently test the different data types and alarms that can be handled by the OPC UA protocol. The UaExpert client was used to test all the server outcomes.

The connection between the server and the Bachmann display as a client was set up using the Atvise connection configurator. Although the connection was established, the method for synchronizing the server data variables with the client was not found. This was due to a lack of information and knowledge about both the device and the software. Consequently, the aim of the thesis was not fully achieved and the connection between the display and the server could not be established as desired.

Future research could be carried out to synchronize the server data with the display. Also, further development of the server certificates would improve the security of the overall environment. Additionally, the thesis could be a guide for projects that require OPC UA.

Language: English

Key Words: Open Platform Communications Unified Architecture, server, PLC, Codesys, Atvise

Table of Contents

1	Introduction.....	1
1.1	Aims and Objectives.....	2
1.2	Project purpose.....	2
1.3	Document structure.....	3
2	Practical framework.....	3
2.1	Open Platform Communications Unified Architecture	3
2.2	Equipment.....	5
2.2.1	Phoenix Contact AXC F 2152 PLC.....	6
2.2.2	Digital module	7
2.2.3	Analogue module.....	8
2.2.4	PWM module.....	8
2.2.5	Read and write devices.....	9
2.2.6	Operator terminal.....	11
2.3	Software.....	12
2.3.1	Codesys.....	12
2.3.2	UA Expert.....	13
2.3.3	Atvise	13
3	Methodology	14
3.1	Phoenix PLC configuration	14
3.1.1	Connection to the computer	14
3.1.2	PLC project.....	17
3.1.3	OPC UA Server	19
3.2	Bachmann OT1210WM installation.....	24
3.2.1	Firmware and licenses	25
3.2.2	Synchronizing the server data to the display	26

4	Results.....	28
4.1	Server results	28
4.2	Client results	31
5	Discussion	33
5.1	Aims and objectives evaluation	33
5.2	Project limitations.....	33
5.3	Consistencies and inconsistencies with the project	34
6	Conclusions.....	35
7	References.....	36
8	Appendices	I
8.1	Important configuration parameters.....	I
8.2	Components purchase information.....	II
8.2.1	OT1210WM purchase information	IV
8.3	OPC UA Built-in data types	VI
8.4	PLC Code	VIII
8.5	Connection between the PLC and the external box	IX

List of Figures

Figure 1: OPC UA specifications.....	4
Figure 2: List of OPC UA Nodes.....	4
Figure 3: OPC UA diagram overview	5
Figure 4: Diagram of the equipment used during the thesis	6
Figure 5: PLC mount with all the modules	6
Figure 6: Digital inputs and outputs in the PLC mount	7
Figure 7: Analogue inputs and outputs in the PLC mount	8
Figure 8: PWM outputs in the PLC mount.....	8
Figure 9: External module for controlling the inputs and outputs	9
Figure 10: Fluke 7 multimeter	10
Figure 11: Operator terminal.....	11
Figure 12: PLCnext window on Codesys	15
Figure 13: Device selection in Codesys.....	15
Figure 14: PLC and its module shown in the devices tree tab in Codesys	16
Figure 15: PLC configuration for always update the variables in Codesys.....	16
Figure 16: Analogue pin mapping.....	17
Figure 17: Functions blocks in Codesys for testing digital values	18
Figure 18: Different networks at Codesys for testing analogue data	18
Figure 19: PWM data assignment in Codesys using the analogue inputs.....	18
Figure 20: App configuration in Codesys.....	19
Figure 21: Symbol configuration from Codesys for the OPC UA server	20
Figure 22: Security Screen access button	20
Figure 23: Certificate configuration in Codesys	21
Figure 24: New certificate icon in Codesys.....	21
Figure 25: Add server popup window in UaExpert	22

Figure 26: Certificate validation popup window on UaExpert	23
Figure 27: Quarantined UaExpert certificated created after trusting it for the first time..	23
Figure 28: Network settings on the display configuration	24
Figure 29: License tab in display configuration	25
Figure 30: Location screen of the device to configure in Atvise connect configurator	26
Figure 31: Display device selection in Atvise connect configurator.....	27
Figure 32: Adding the OPC UA server in the Atvise connect project	27
Figure 33: Connections status in Atvise connect with the PLC connected	27
Figure 34: Photo from the external box connected to the working PLC.....	28
Figure 35: Location of the PLC variables in the UaExpert client	29
Figure 36: PLC variables data are shown in a) Codesys b) UaExpert client.....	30
Figure 37: Events viewer in the UaExpert client	30
Figure 38: OPC UA client option variable list in Atvise connect.....	31
Figure 39: Adding the OPC UA source in Atvise builder	31
Figure 40: OPC UA server variable browser in Atvise builder	32

1 Introduction

It is recognized through history that society has shifted through four different industrial revolutions. According to Deane (1965), the first started in Great Britain during the mid-18th century and lasted between 1820 and 1840. During this period, human and animal power transitioned to machinery power with the invention of steam motors (Mohajan, 2019). The key points of the second revolution, which took place from the 19th century into the early 20th century, were the discovery of new energy resources such as electricity, oil and gas, and the introduction of mass production by the assembly line. In the mid-20th century, the third industrial revolution began and is known for introducing nuclear energy and using electronic devices to automate production lines and processes (iED Team, 2019).

During the third industrial revolution, a key component of current production lines was introduced, the Programmable Logical Controller or PLC. This device is a computer specifically designed to operate and control industrial processes and machinery such as conveyor systems, manufacturing lines and many other different types of machines and processes (Romero & Theorin, 2012).

Nowadays, the fourth industrial revolution is emerging and adapting the technologies brought from the third and refining them to fulfil emerging needs. According to Schwab (2016), connectivity and artificial intelligence (AI), such as cyber-physical systems, the Internet of Things (IoT) and further similar technologies, are the industrial revolution's main aspects.

A key enabler to adapt the PLCs to make them suitable for the fourth industrial revolution will be a huge increase in levels and rates of connectivity and compatibility. This is due to every device driver requiring software from its vendor. If protocol changes, it can result in communication errors that require adjustments that prolong the projects and create unexpected costs for companies (Pranowo et al., 2020). Adding to the current global silicon crisis which has led to a decrease in PLC production, companies must use different manufacturers, which as stated, increases the money and time put into the companies' projects (Chakraborty, 2022).

A solution to the problem is a standardized, open-source architecture capable of translating specific protocols (such as Profibus, Modbus, etc.) into a standardized interface for control

devices that can convert the generic protocol into the device-specific one and vice versa. This is accomplished with the Open Platform Communications or OPC standard, which is maintained by the OPC Foundation (OPC Foundation, 2017a). In 2008, the OPC Foundation released an architecture that integrates all the functionalities of all the individual specifications of classic OPC into the same framework and named Unified Architecture, also known as OPC UA (OPC Foundation, 2019).

1.1 Aims and Objectives

This thesis aims to engineer a system with one PLC acting as the server communicating with the OPC UA client in the operator terminal, thus laying the OPC UA foundations for further creation of laboratory tasks for students to acquire the protocol knowledge.

To achieve the aim of this thesis, the following objectives are defined:

- Define OPC UA server on PLC identifying the restrictions.
- Configure a data structure for the OPC UA server.
- Establish a secure connection between the server and the client.

1.2 Project purpose

Joachim Böling, senior lecturer in Automation Technology at NOVIA University of Applied Sciences, has commissioned the thesis. The interests come from the current automation situation, where everything is transitioning to the fourth industrial revolution, thus compatibility and connection between devices are mandatory. And as stated, a solution is adapting the existing devices to fulfil the new revolution needs. In this case, with the help of new software to connect different devices, such as PLCs and operator terminals (OT), from different manufacturers, by the modern OPC UA architecture. The University of Novia does not have any experience with architecture, and this thesis will provide some of the bases to further educate new students about this promising technology.

1.3 Document structure

This thesis starts by introducing the history of automation, followed by a brief overview of the context, objectives, and purpose of the project (Introduction). During Practical framework the theoretical aspects of the OPC UA environment will be introduced, followed by the explanation of the equipment used during the project.

The practical framework is detailed in Methodology with a step-by-step explanation of the procedures conducted to get the desired result. This section will be divided into two parts, the first regarding the PLC and server part, and the second and last focusing on the display acting as the client.

Lastly, 28 analyses the results of the project. Proceeding with the evaluation of the outcome, Discussion and Conclusions discuss the project and proposes future implementations.

2 Practical framework

The OPC UA environment requires different software and hardware devices. This section will give an overview of all the components used during the thesis by giving a brief explanation of both theoretical and technical aspects.

2.1 will discuss the basics of the OPC UA protocol by reviewing its main aspects and giving a detailed description of the most important protocol specifications for this project, while section 2.2 introduces all the hardware equipment used for the thesis, and 2.3 is an overview of the software used for operating the different devices.

2.1 Open Platform Communications Unified Architecture

According to the OPC Foundation (2022), in the middle of the 19th century, some automation vendors Rockwell Software, Opto 22 and others form a task force to develop a standard for data access. During the following years, they developed different specifications which provided separate definitions for alarms, data access and history. But in 2008, the development and creation of new technologies pushed the release of a

unification of all the different OPC Classic specifications into one, named OPC Unified Architecture, commonly known as OPC UA (OPC Foundation, 2019b).

The unified architecture consists of 14 specifications that can be found in Figure 1. According to Rinke (2022), the most important specifications to know for operational use are Data Access, Historical Access and Alarms and Conditions.

Data Access describes point-oriented data exchange, where each point is described by its value, the time stamp at when it was current and the quality of the value, whether it describes if the value is valid or if there have been some issues with the connection.

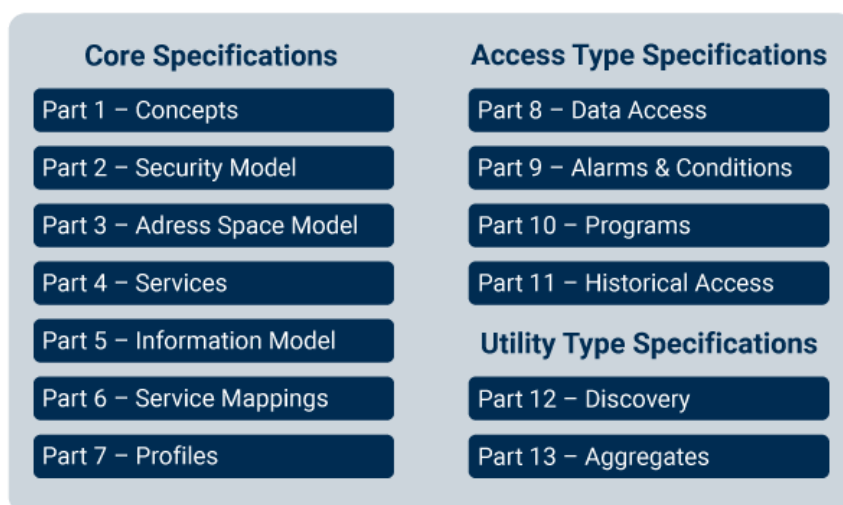


Figure 1: OPC UA specifications. (Source: www.opc-router.com)

Historical Access functions to store and show the previous values and their time stamp. Therefore, to store these historical values, the controller must provide an internal data memory (OPC Foundation, 2019a).

Another important specification that helps to understand how OPC UA works is the Address Space Model, which defines different types of objects which are called nodes (Profanter, 2019), and a list can be found in Figure 2.

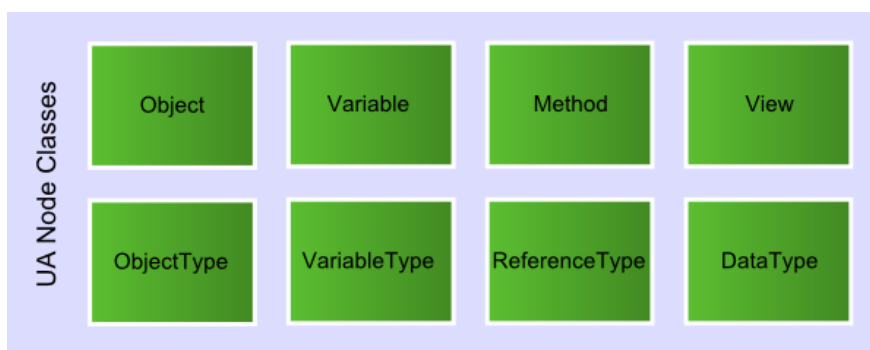


Figure 2: List of OPC UA Nodes. (Source: documentation.unified-automation.com)

To identify the different nodes, unique node identifications are used. These identification nodes, also known as nodes id, are only complete if both the identification and the namespace Uniform Resource Identifier, or namespace URI, are indicated (OPC Foundation, 2017b).

Essentially, the nodes are the structures that define the values in the OPC UA protocol. Therefore, an important part of the thesis is to properly define these nodes to correctly read them through the different clients.

2.2 Equipment

To create an OPC UA environment various hardware devices are required. The basic environment is composed of one server and at least one client, where an example can be seen in Figure 3.

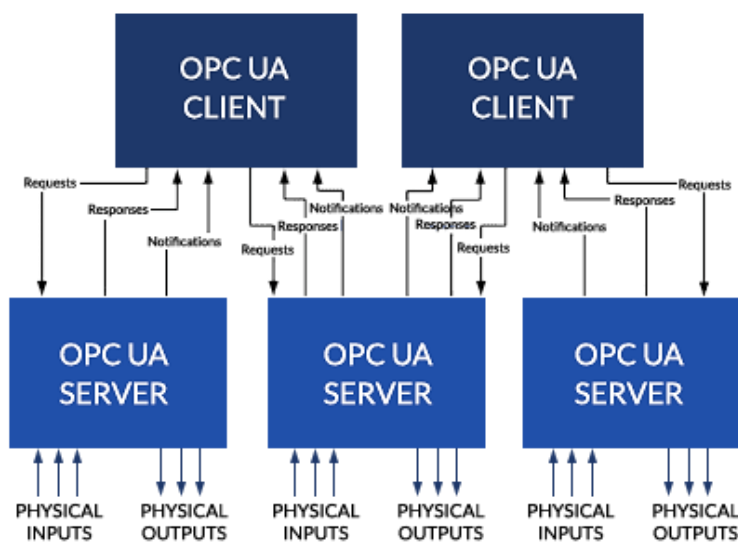


Figure 3: OPC UA diagram overview. (Source: techstep.co.nz)

In the final version, the PLC acted as the server while the operator terminal, or OT, acted as the client. However, during the testing phase, different devices were used to test the different functionalities. A diagram for the test environment for the thesis can be seen in Figure 4.

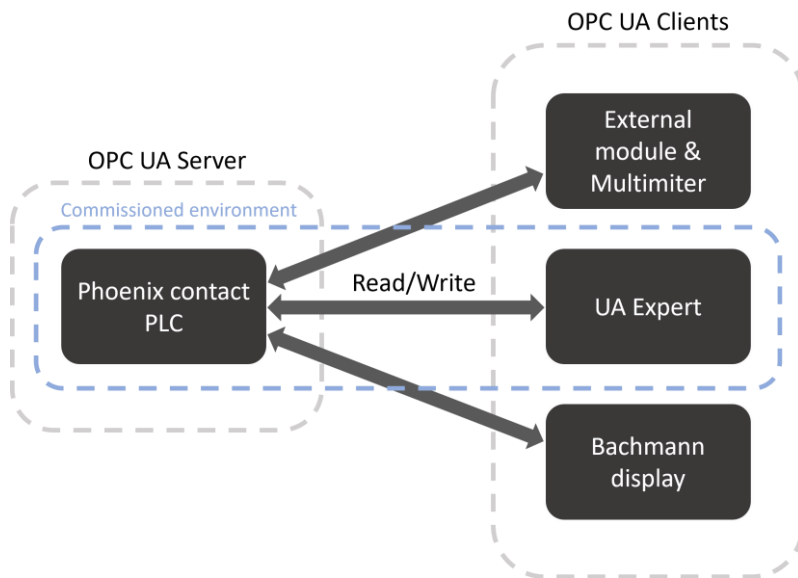


Figure 4: Diagram of the equipment used during the thesis. (Source: Author's own)

As can be seen in Figure 4 above, the PLC acts as the OPC UA server. In bigger installations like the ones that can be found in factories, the server could not be in the PLC, but in devices that only store the servers.

2.2.1 Phoenix Contact AXC F 2152 PLC

The PLC used for the thesis is the AXC F 2152 model from Phoenix Contact, which is designed for maximum performance, easy handling and harsh industrial environments (Phoenix Contact, 2013). The PLC is mounted in a platform designed by Novia for easier use of the extension modules in a student environment, as can be Figure 5.

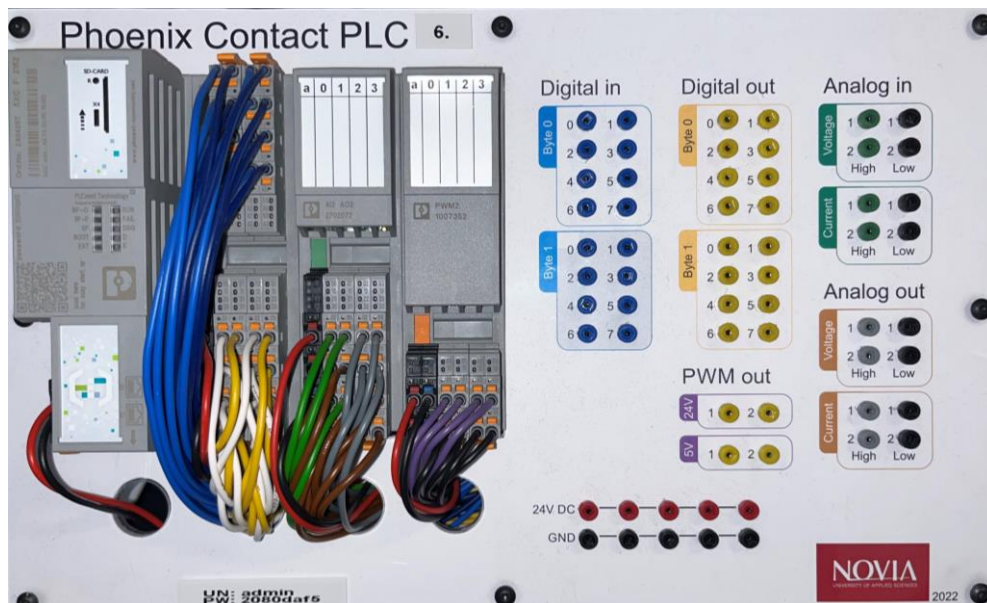


Figure 5: PLC mount with all the modules. (Source: Author's own)

The PLC setup has three different modules, connected with the local bus by a push-in connection with a speed of 100Mbps. The bus system used to interact with the PLC is two ethernet RJ45 interfaces, with a speed between 10 and 100 Mbps (Phoenix contact, 2013). It supports various protocols, such as OPC UA, which makes this PLC suitable for the thesis. Different hardware has been used to control the different inputs and visualize the outputs for easier readings.

2.2.2 Digital module

The digital module, named AXL DI16/1 DO16/1 2H 2702106, has sixteen different digital inputs and an equal number of digital outputs that read Boolean values from -3V DC to 30V DC, with a nominal input voltage U_{IN} of 24V DC and a range from -3V to 5V for the 0 or False value, and from 11V to 30V for the 1 or True value (Phoenix Contact, 2019a). The pins are easily accessible through the PLC mount, as shown in Figure 6. For more information and the datasheet, see 8.2.

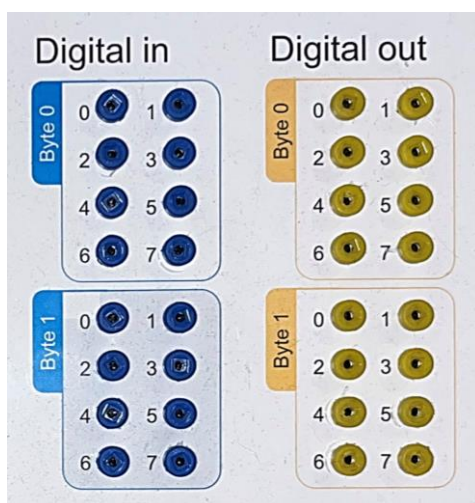


Figure 6: Digital inputs and outputs in the PLC mount. (Source: Author's own)

To easily write and read the values, the external module provided by Novia UAS detailed in 2.2.5 is used. The values can also be accessed via the different software, both the one used to program the PLC and the OPC UA client.

2.2.3 Analogue module

This module is identified by the name AXL F AI2 AO2 1H module. As its name suggests, is composed of two analogue inputs and two analogue outputs that read and write analogue data with a range from -10V to 10V (Phoenix Contact, 2019b). As with the other modules, the PLC mount offers easy access to the pins, as shown in Figure 7.

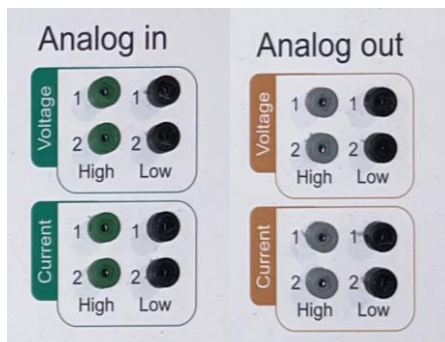


Figure 7: Analogue inputs and outputs in the PLC mount. (Source: Author's own)

The use of the potentiometer in the external module, explained in 2.2.5, serves for writing the analogue values to the PLC, while the multimeter Fluke 175 is used to read the analogue outputs. As with the digital module, the values can also be accessed via software, which depending on the situation could be preferable. For more information about the module, see the datasheet in 8.2.

2.2.4 PWM module

The third module used in this configuration is called AXL F PWM2 1H, which consists of two different pulse width modulation or PWM outputs, each of one with two different nominal outputs voltages, one of 24 V DC and the other of 5 V DC (Phoenix Contact, 2019c). The pins are accessible through the PLC mount, as Figure 8 shows.

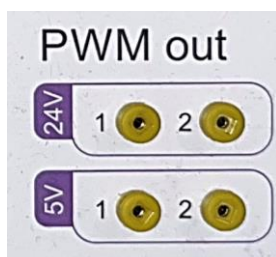


Figure 8: PWM outputs in the PLC mount. (Source: Author's own)

The output of the module is controlled by four words, two for each output. As can be seen in Table 1, the least 10 important bits in the words 0 and 2 are used to define the pulse

duty factor of the PWM signal, and words 1 and 3 define the frequency of the signal. For more information, see 8.2.

Table 1: Out process data for the PWM module. (Source: datasheets.shortec.com)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word 0	Not used						Channel 1, pulse duty factor									
Word 1	Channel 1 frequency															
Word 2	Not used						Channel 2, pulse duty factor									
Word 3	Channel 2 frequency															

To read the signal from the module, the root mean square or RMS value is read via the multimeter. This is not an accurate value since it calculates the square root of the arithmetic mean of the squares of the values from the PWM signal, but the multimeter aims to get an easy reading to give an idea if the program is working as expected, and the exact value can be accessed from the different software.

2.2.5 Read and write devices

For writing and reading the different data and values from the PLC it can be done both via software and hardware. While the software does not require extra equipment, there are some situations that are less favourable since it can take more time. On the other hand, the different hardware gives instant reading and writing values. Thus, a multimeter and an external module with switches and potentiometers are used.

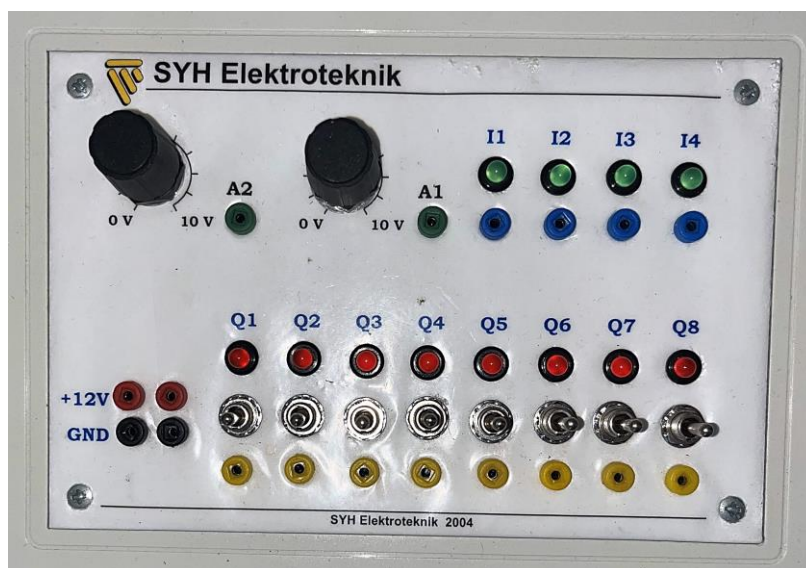


Figure 9: External module for controlling the inputs and outputs (Source: Author's own)

The external module box is composed of two different potentiometers, twelve LEDs and eight switches, where the arrangement of the different devices can be seen in Figure 9. The switches and LEDs are used for writing and reading digital data accordingly, while the two potentiometers are used for writing analogue data. When reading and writing the analogue data, both High and Low pins must be used to get accurate values.

The problem with using the external module with the PLC power supply is that it supplies 24V DC, thus the potentiometer writes values from 0V DC to 24V DC, but the analogue module only accepts voltages ranging from -10V DC to 10V DC. When voltages higher than the accepted range are written, the error LED in the analogue module turns red, meaning an error in the local bus has occurred. Also, the error LED on the processor turns yellow, which is a warning in the device's local bus (more information in the datasheet found in 8.2). But the module protection shields the high voltages, so when the voltage drops to the working range the LEDs turn off.

The multimeter used in the thesis is named Fluke 175, which is shown in Figure 10. This multimeter offers various measurements that can be selected with the rotatory switch, but the only measurement needed is the direct current voltage or DC.



Figure 10: Fluke 7 multimeter. (Source: Author's own)

Therefore, during the testing phase, the encoder was in the third position starting from the off position, and the cables were connected to the red and black inputs that can be seen in the bottom right of Figure 10.

2.2.6 Operator terminal

According to Knapp (2011, pp. 89–110), a Human Machine Interface (HMI) is a software and hardware device an operator uses to interact with the PLC. The devices can range from a physical control panel, such as the external module with switches and potentiometers, to operator terminal displays.

During the development of the OPC UA environment, the different testing was easier with physical controls from the external module. That is because only the connection between PLC and the physical controls is needed, which is the fastest method. But when PLCs are used in the industry, there can be large amounts of inputs and outputs that would make physical devices hard to control, hence dangerous. Therefore, the usage of the operator terminal.

For the thesis, the display available at Novia UAS was the OT1210WM from the OT1200 series from Bachmann Industries, shown in Figure 11.



Figure 11: Operator terminal. (Source: <https://www.bachmann.info>)

According to Bachmann Electronic GmbH (2020), the display specifications make it ideal for small and medium web visualizations such as Atvise, hence the choice of the Atvise software. Web visualizations, instead of relying on the program installed on the device, connect to the internet to access and visualize the data from the web (Rohrer & Swing, 1997). So, for the thesis, the Supervisory Control and Data Acquisition or SCADA will not be installed directly on the display but accessed through its browser.

The display has the 3.4 Atvise version preinstalled, which is used to connect the OT through an OPC UA server to the computer for designing the SCADA. The interface of the display is composed of two ethernet RJ45 connectors that read and write the data from the PLC as also being used to configure it. It also has two USB 2.0 connectors that can be used to connect an external keyboard and mouse to easily control it without the use of the touchscreen display. This display model integrates a CFast interface with a lock, that is used for inserting SD cards, but it was not used during the project. Lastly, the OT is supplied by a 24V DC power supply.

2.3 Software

The use of different manufacturers for the devices required using different software programs to engineer the OPC UA environment, thus for this project three different software programs were used: Codesys for the PLC, Atvise for the OT and UA Expert acting as a client for testing the OPC UA server.

2.3.1 Codesys

Normally, the PLC manufacturers provide their software which is designed specifically for their devices. But for this project, the technical thesis supervisor, Joachim Böling, suggested using Codesys instead of PLCNext, which is the software intended by the manufacturer. The reasons behind this are that Codesys is free to use, hence no license fee for end users, and that the software is used in other Novia courses, so it will be easier for students to use it since they will be already familiarized.

At the writing of the thesis, Codesys is available from <https://store.codesys.com/en/>, where only registration is needed to install it. But for more advanced configurations, different add-ons can be bought from the same webpage. This can range from less than 100€ to more than 2000€.

The version used for the thesis is the 3.5 SP18 patch 4. For controlling the Phoenix Contact PLC, the PLC Next add-on had to be installed.

The software incorporates the IEC 61131-3, which specifies the syntax and semantics for programming languages for programmable controllers (PCs). The IEC standard incorporates

two graphical languages, Ladder Diagram (LD) and Function Block Diagram (FBD), and two textual languages, Instruction List (IL) and Structured Text (ST) (IEC, 2013). It also incorporates tools to create and manage the OPC UA protocol, as other options and objects to program PLCs.

2.3.2 UA Expert

There are different OPC UA clients available that can be easily downloaded from the internet, but UA Expert was chosen for its open-source code, which makes it free to use and consequently its popularity. Therefore, it is easier to find information about it (cacamille3, 2022). And according to Codesys group (2022), it can be easily connected to the Codesys OPC UA server.

UA Expert is a fully featured client that supports OPC UA features like data access, calling of UA Methods, Alarms and Conditions, etc. The software is free to download and use without any restrictions, only registration is mandatory (Unified Automation GmbH, n.d.). The software is available from <https://www.unified-automation.com/downloads/opc-ua-clients.html>. The version used for the thesis was 1.6.3.

2.3.3 Atvise

Atvise is a company launched in 2005 that offers leading visualization and center solutions based on OPC UA and pure web technologies (Atvise, 2015). The Atvise SCADA bundle provides different software to engineer the HMI and connect it to the PLC. In this bundle, used for the thesis, Atvise Connect, Atvise Builder and Atvise Monitor can be installed. The different installations are available from: <https://atvise.vesterbusiness.com/en/download>. Atvise Connect provides connectivity from the display to the different industrial controllers and ensures data acquisition. Therefore, the OPC UA server data structure has to be configured via this program for sending it to the OT, and at the same time, writing the values changed from the display (Atvise, 2023b).

Atvise Builder is the software engineered for creating HMI solutions. To configure the SCADA system for the OT, first, a connection to the OPC UA Atvise server in the display must be ensured (Atvise, 2023a).

The software administrating the Atvise bundle licenses is Atvise Monitor. These licenses are linked to the devices, so to properly design the system, the display must be connected to the computer. But the workaround is to use the trial license, which only requires an email and lasts one month. This program also starts a local OPC UA Atvise server to design SCADA that can only be accessed locally (Atvise, 2023c).

3 Methodology

In this chapter the configuration of the environment is detailed. First, the PLC part is explained by dividing it into the setup, the project on the Codesys software and the OPC UA server commission. The second part of this section describes the installation and the implementation of the display.

3.1 Phoenix PLC configuration

In this sub-section, the configuration of the PLC, which acts as the OPC UA server, will be explained. First, in section 3.1.1 **Error! Reference source not found.** the preparation of the device and the establishment of the connection between the PLC and the computer are detailed. Secondly, in 3.1.2 the data structure created for the project is explained. Lastly in section 3.1.3, the configuration of the OPC UA server is described.

3.1.1 Connection to the computer

To establish a functional connection between the PLC and the computer, the IPv4 address in the computer was changed from automatic to manual. Any 192.168.1.XX address should work, but 192.168.1.99 was set to avoid using the same address as other devices. Other important configurations can be found in Appendix 8.1.

To set up the PLC using Codesys, a new standard project was created with the PLC's next SL device selected. The programming language does not affect the script since it could be changed afterwards, but the Function Block Diagram (FBD) was selected since was the one the author was most used to. To avoid problems with the PWM module, the newest CODESYS runtime version must be installed in the PLC. To do so, the 'Update PLCnext' option under the tools tab (Figure 12), takes to the PLCnext window. There, the devices

were scanned, and Phoenix PLC was selected. After, the install button was clicked on the runtime package section. If it is the first connecting the PLC to the computer, a login to the PLC is required. The username and password can be found in Appendix 8.1.

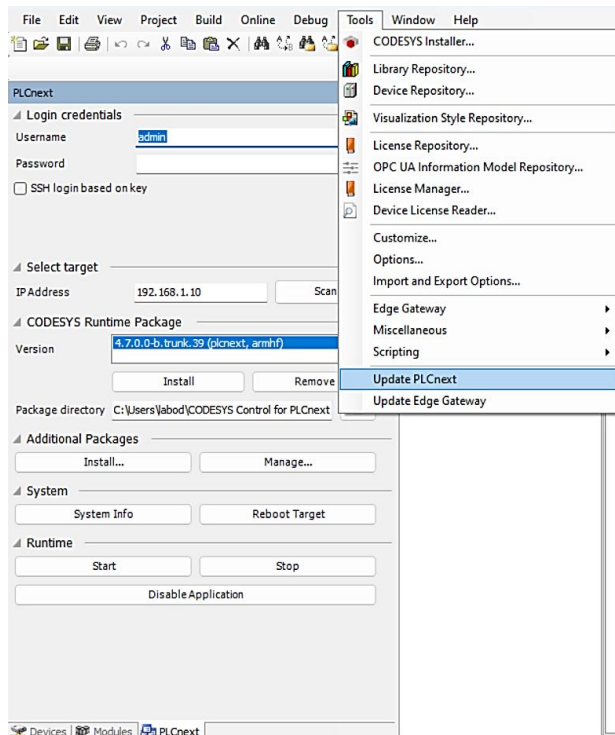


Figure 12: PLCnext window on Codesys. (Source: Author's own)

After installing the latest PLC version, to establish the connection a new Gateway was added. The IP address must be the same as the IPv4 address, in this case, 192.168.1.99. After setting the gateway, clicking on the 'Scan Network' button opens a popup window, where the PLC should appear as axcf2152 [000A]. Figure 13 shows the popup window for selecting the device, with the Phoenix PLC connected to the computer.

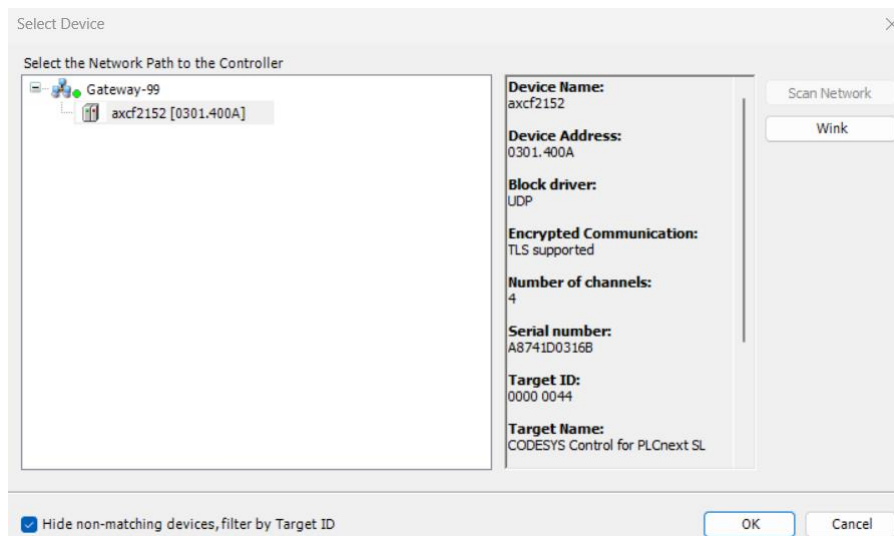


Figure 13: Device selection in Codesys. (Source: Author's own)

If the connection has been established, a new branch for the PLC should appear in the devices tab. Right-clicking into the PLC and then selecting the 'Add device...' option should open a new window with all the possible modules. As explained in section 2.2, the set-up consists of three different modules, which are added to the Codesys project. The easiest way to identify each module in the program is by its number code, printed on its case. The section corresponding to the PLC in the devices tab should be the same as in Figure 14.

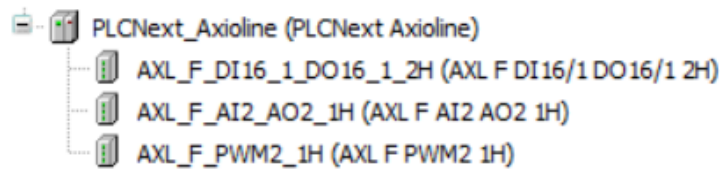


Figure 14: PLC and its module are shown in the devices tree tab in Codesys. (Source: Author's own)

For easier and faster testing, the updating of variables is set to always update. Therefore, the changes in the values can be seen instantly in Codesys while running the PLC. This setting is found under the device tab, in the PLC Settings. In this window, the 'Always update variables' dropdown menu is set to enable. Figure 15 shows the configuration used during the thesis.

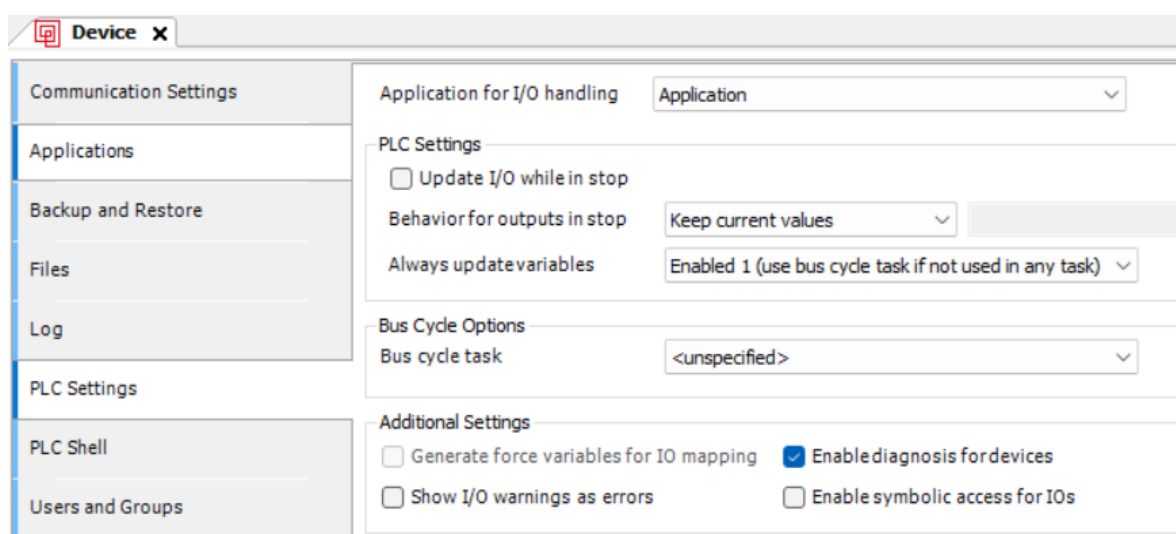
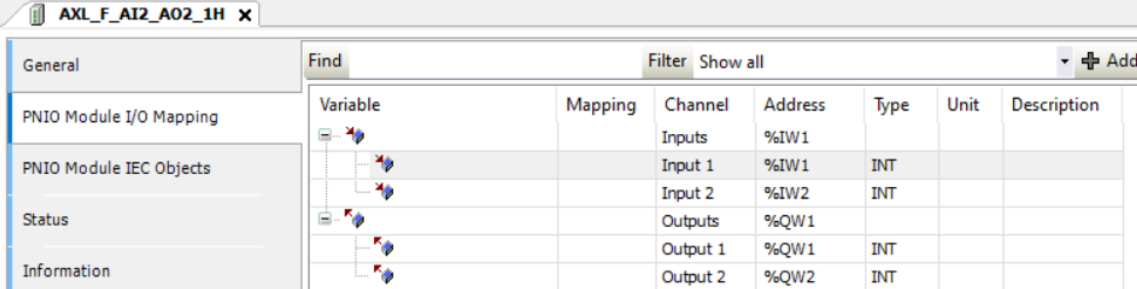


Figure 15: PLC configuration for always update the variables in Codesys. (Source: Author's own)

The box for updating I/O while in stop could also be selected, so the variables would be always updated, but it would also require a download every login. Thus, during the thesis, it was not selected.

3.1.2 PLC project

This project is centred on the development and installation of the OPC UA server; therefore, the PLC project and code are focused on testing all the different data types that can be sent through the OPC UA server in the Phoenix PLC. Since there is no hardware such as sensors or motors in use, the variables in the code were named after their type, and with the I/O pin number. The address of the inputs and outputs pins for the different modules can be found in the PNIO Module I/O mapping tab, which is in the module option in the devices tree tab. An example of the analogue pin mapping can be found in Figure 16.



Variable	Mapping	Channel	Address	Type	Unit	Description
		Inputs	%IW1			
		Input 1	%IW1	INT		
		Input 2	%IW2	INT		
		Outputs	%QW1			
		Output 1	%QW1	INT		
		Output 2	%QW2	INT		

Figure 16: Analogue pin mapping. (Source: Author's own)

Since the main purpose of the PLC was to test the different aspects of the OPC UA environment, the code was as simplified as possible. Thus, only basic Boolean operators and variable conversions boxes were used. To test the digital data, also known as Boolean, which ranges from 0 to 1 or true and false, the digital module was used. This module is capable of reading 16 different bits from its input, and it can write 16 bits with its output pins.

To implement the Boolean variables with logic, an AND Boolean operator block with enable and enable (EN/ENO) output was used. The EN part represents enable, and controls whether the function box is activated or not, while the ENO is used to control if the function has been executed successfully (Schneider electric, 2019). To use the different function blocks, they can be dragged from the toolbox located on the right side of the program.

Therefore, this function block was used with the EN input connected to the negated ALERT bit alarm, which if activated, created an alarm, and disabled the function. The two normal inputs were assigned to the digital input bits 0.1 and 0.2. When both bits are true, the function output sets to 1 and the digital output 0.1. Figure 17 shows the code in function blocks in Codesys. The full code can be found in PLC Code.

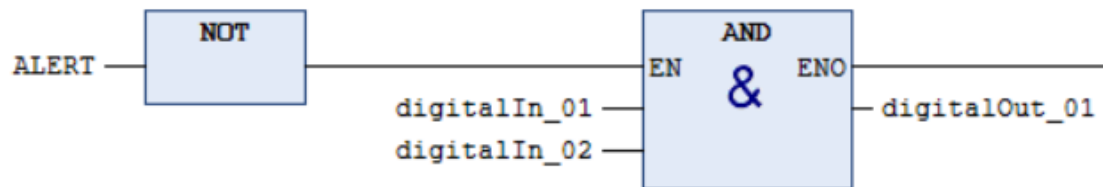


Figure 17: Functions blocks in Codesys for testing digital values. (Source: Author's own)

The analogue module transforms the I/O values into integers data (INT). Both inputs were used and named analogueIn_1, for the first input and analogueIn_2 for the second one. The value of the first input was assigned to the first analogue output, named analogueOut_1. Both analogue inputs were used in different function blocks provided by the program that convert the integer values into other data types (Figure 18). This was done to test that other data types were sent and received correctly to the OPC UA server.

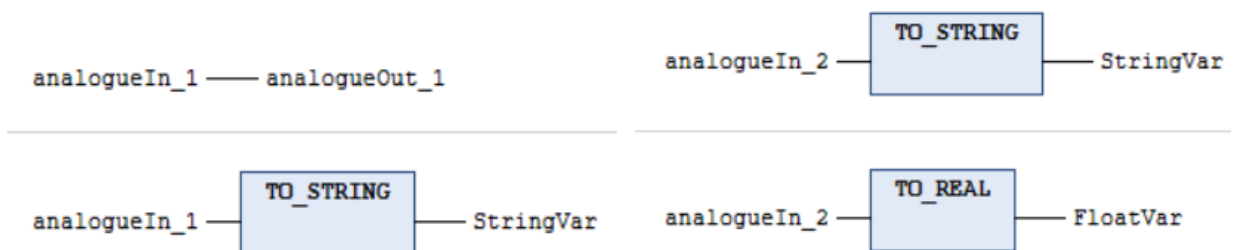


Figure 18: Different networks at Codesys for testing analogue data. (Source: Author's own)

For testing the PWM module, two different bytes are assigned. The first one, named PWMPulseDuty, controls the pulse duty factor, and the second byte named PWMOutFreq controls the frequency of the signal. Using conversion function blocks, the analogue input one was assigned to control the pulse duty factor, and the analogue input two to the frequency (Figure 19).

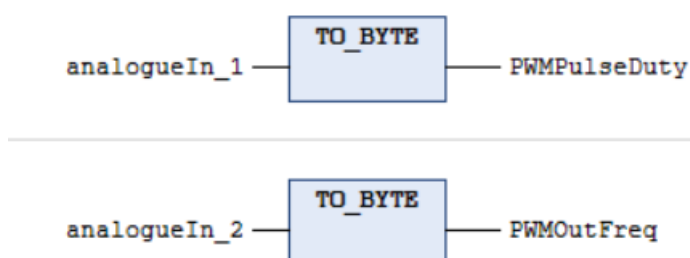
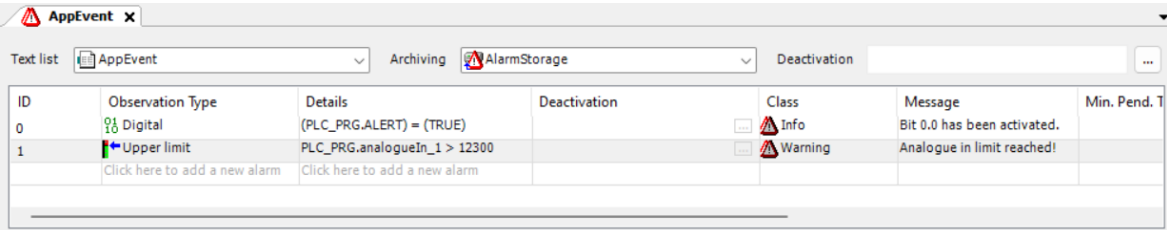


Figure 19: PWM data assignment in Codesys using the analogue inputs. (Source: Author's own)

For testing the alarm system, two alarms were created. The first one is a digital alarm that activates when the bit with address %IX0.0 is set to 1. In the code, this bit is named ALERT. In a real-world scenario, this could be for example a sensor that detected some malfunction. The other was an upper limit alarm, that activated when the analogue input

one reached a certain value. This could simulate a temperature sensor that reached a higher temperature than desired.

To set the alarms, an alarm configurator must be added. This is done by right-clicking the Application object in the devices tab. In the opened menu, clicking the 'Alarm Configuration' should add the object. When added, three alarm classes are added automatically: Error, Info and Warning. Since the main purpose of the Codesys project is to test the OPC UA server, no more classes were added. To add and edit the alarms, an alarm group was added by a right click on the Alarm Configuration object in the devices tree tab, and then selecting the 'Alarm group' in the 'Add object...' option.



ID	Observation Type	Details	Deactivation	Class	Message	Min. Pend. T
0	Digital	(PLC_PRG.ALERT) = (TRUE)		Info	Bit 0.0 has been activated.	
1	Upper limit	PLC_PRG.analogueIn_1 > 12300		Warning	Analogue in limit reached!	

Figure 20: App configuration in Codesys. (Source: Author's own)

In the alarm group configuration, the mentioned alarms were configured. Figure 20 shows the result of the configuration. The arbitrary value for the upper limit alarm was set to 12300. The digital alarm was set to the Info class, while the analogue was set to the Warning. The messages were set to facilitate the testing phase, which is discussed in section 4.

3.1.3 OPC UA Server

In order to configure the OPC UA server in the PLC, the variables must have been created in the program. When created as explained in the section above, the first step is to insert a Symbol Configuration object by right-clicking 'Application' in the devices tree tab. Then, the Symbol Configuration can be found in the 'Add Object' option. While in the Add Symbol Configuration, the Support OPC UA features option must be checked.

Once is created, the Symbol Configuration can be opened. If it is the first time that the window is opened, no variables will be shown. By clicking the 'Build' button, all the variables set in the program will show in a tree structure, as can be seen in Figure 21.

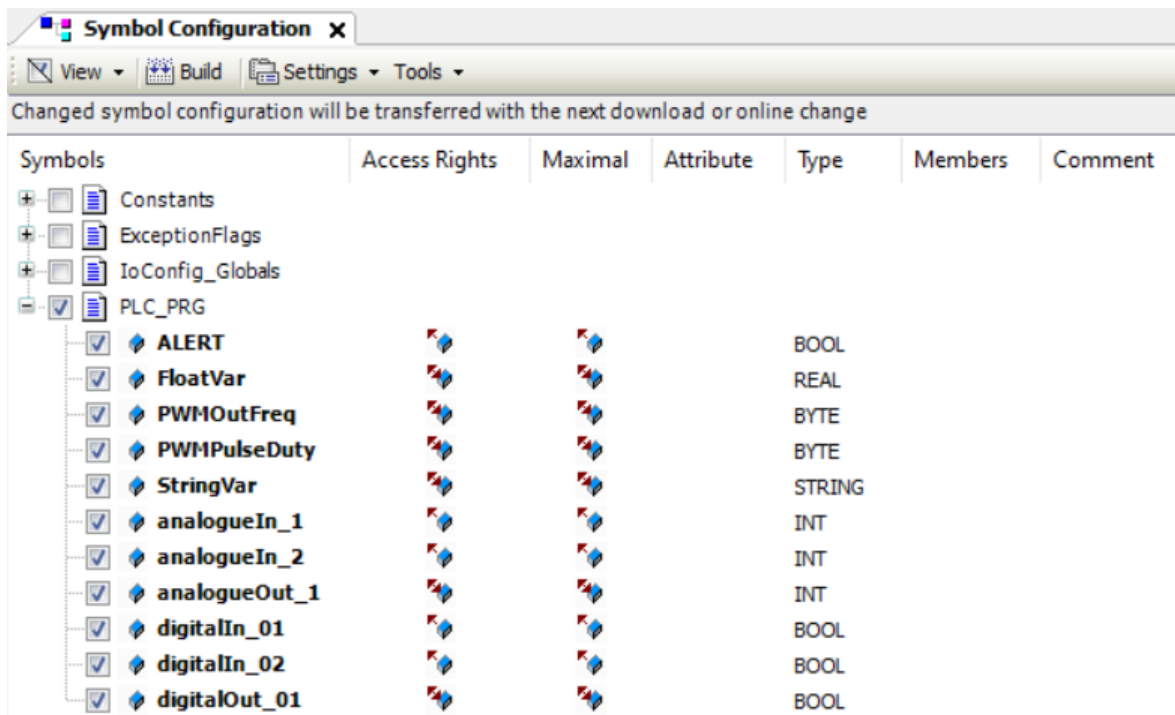


Figure 21: Symbol configuration from Codesys for the OPC UA server. (Source: Author's own)

The variables desired in the server have to be checked, and their access specified. In the next download or online change, the checked variables will be transferred to the PLC server. These first steps can be done without an established connection to the PLC, but the next steps will require it.

The next step requires the configuration of the certificates for the server. This configuration is done by the Security Tab, which can be accessed by double-clicking the security button, found in the bottom right corner of the program, and having the symbol found in Figure 22. It can also be accessed in the Device tab by in the security setting option in the device dropdown menu.



Figure 22: Security Screen access button. (Source: Author's own)

In the security window, the Devices tab is selected. Then, in the Information list, selecting Device will show all the services that require a certificate to use. Clicking the OPC UA in the list will select the service. Figure 23 shows the necessary certificates to create the OPC UA server in the PLC.

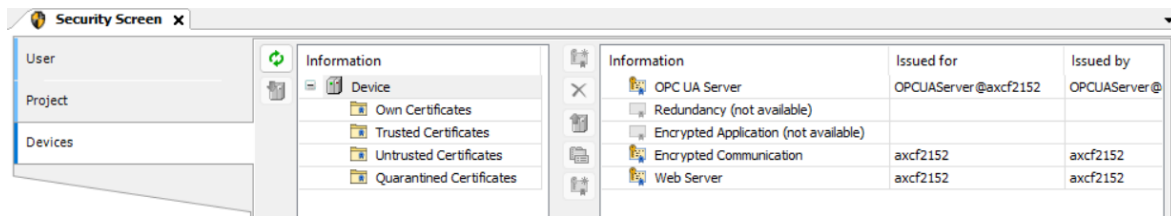


Figure 23: Certificate configuration in Codesys. (Source: Author's own)

If the certificate is expired, it can be either regenerated or deleted after creating a new one. If not created or just deleted, the icon shown in Figure 24 will create a new one. In the dialogue box that will pop up, the date for the certificate is set to 2000 days. This is due to the fact that all the certificates are created from the date 02/02/2020. So, if after creating the certificate it is expired, a new one must be created with a longer date.



Figure 24: New certificate icon in Codesys. (Source: content.helpme-codesys.com)

Once the certificate is up to date, to allow anonymous login, the checkbox on the 'Change Runtime Security Policy' has to be checked, which is found in the device tab under the devices dropdown menu. After that, the OPC UA server should be set up correctly.

To test that all the variables can be accessed by a client, UaExpert is used. To establish a secure connection, the UaExpert certificate must be properly installed on the server in the PLC controller. First, the PLC server is added by clicking on the plus sign icon found in the toolbar on the upper left side of the software, which opens the popup window shown in Figure 25. There, double-clicking on the blue text under 'Custom Discovery' opens a new window, where the OPC UA server IP is configured. During the project, the IP for the OPC UA server was set to 192.168.1.10. The OPC UA port is 4840, so the IP address 192.168.1.10:4840 should work the same.

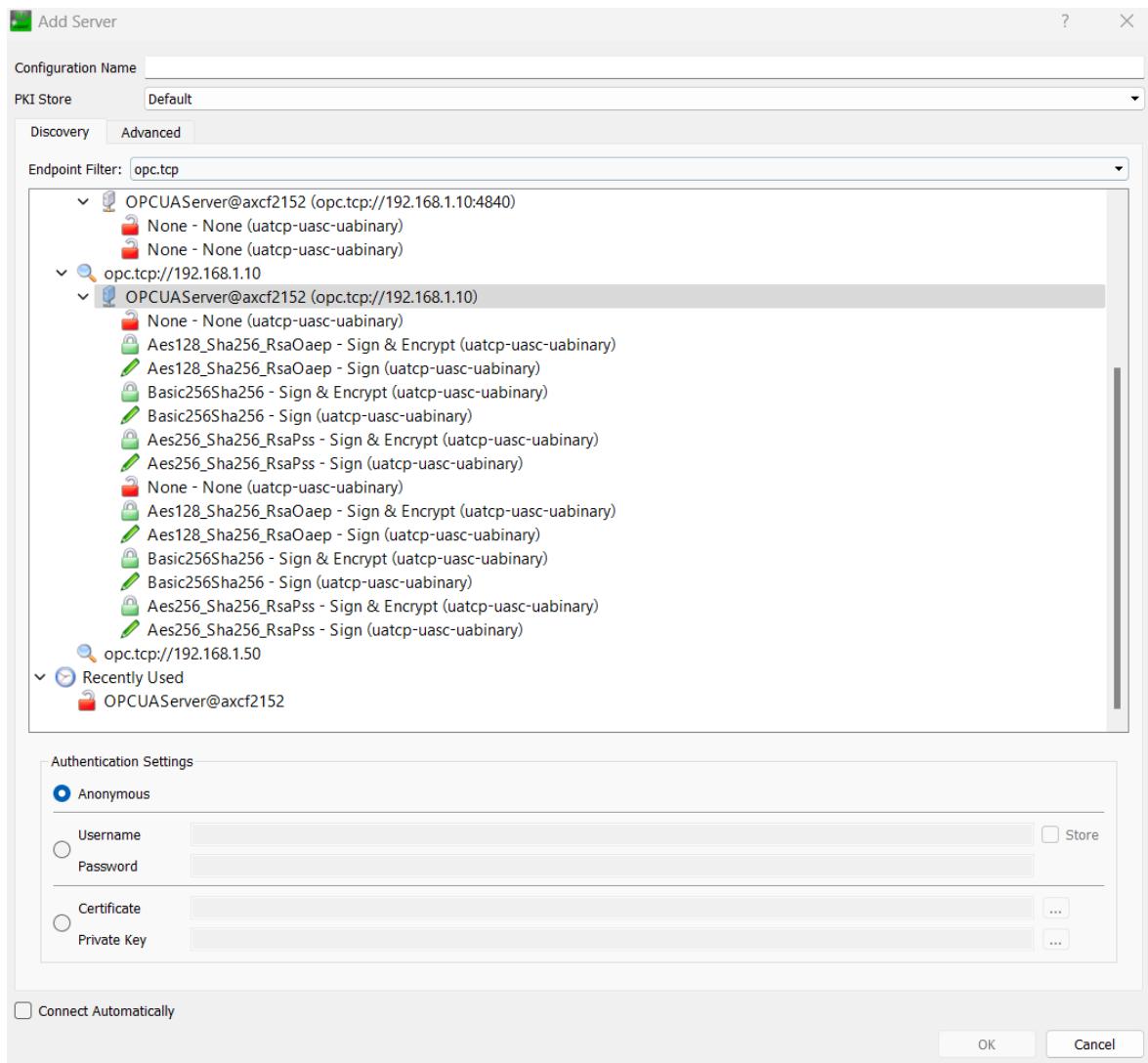


Figure 25: Add server popup window in UaExpert. (Source: Auhtor's own)

After adding the server IP, all the options shown in Figure 25 appear. The one used during the thesis was the second with the red padlock icon, named 'None – None (uatcp-uasc-uabinary)'. After pressing ok, the IP is added to the project tree window under the server branch. The plug icon, on the right of the add symbol icon pressed before, is used to connect to the server.

The first time that the connection is established with the OPC UA server, the popup screen shown in Figure 26 will appear. That means that the PLC does not have the UaExpert certificate, so it needs to be installed. But first, by pressing on the trust the server certificate option will open the OPC UA server, but only for the session, since the certificate is not properly installed.

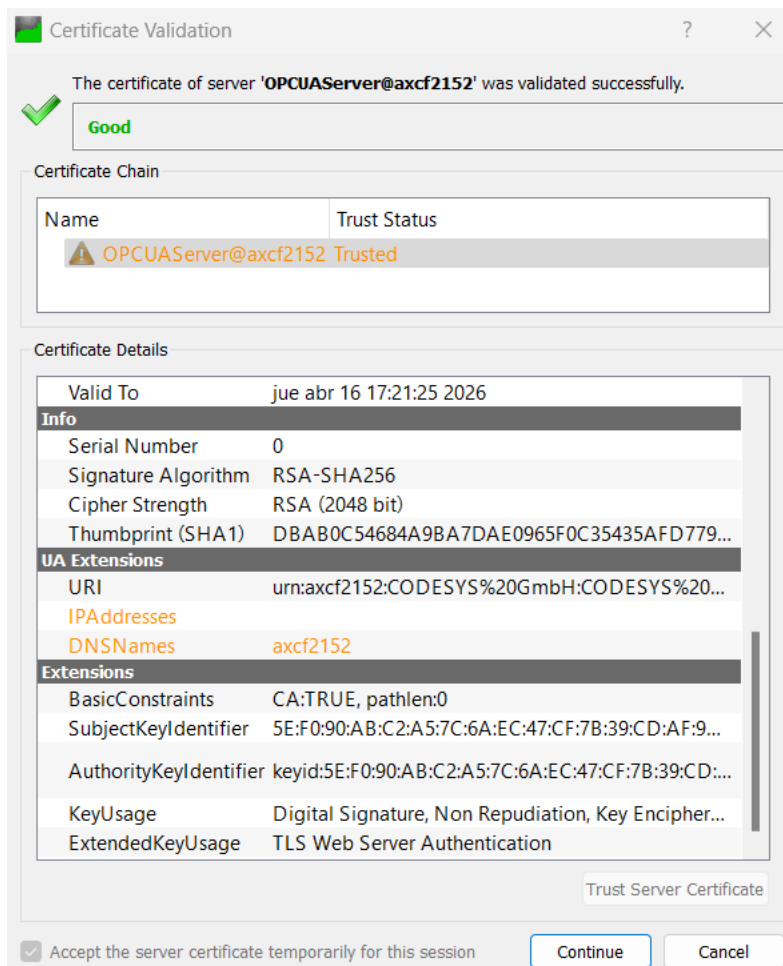


Figure 26: Certificate validation popup window on UaExpert. (Source: Author's own)

To install the certificate, after trusting it as mentioned for the first time, will create a quarantined certificate (Figure 27). In Codesys, going into 'Quarantined Certificate' and dragging the UaExpert certificate recently created to the trusted certificates option will properly install the certificate for further connections to the server by the client software.

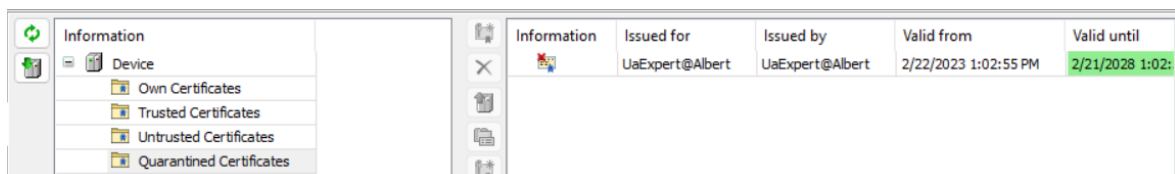


Figure 27: Quarantined UaExpert certificated created after trusting it for the first time. (Source: Author's own)

Once the certificates are managed correctly, the UaExpert client should be able to access all the OPC UA data. In the following section, the functioning of the server will be tested.

3.2 Bachmann OT1210WM installation

In this section, the configuration and commission of the Bachmann OT1210 display also referred to as the operator terminal, will be explained in detail. Firstly, in section 3.2.1 the preparation of the display will be determined, and in section 3.2.2, the process of designing the SCADA system will be described.

But first, a short explanation of how to access the display settings through the computer is needed, which facilitates its configuration of it and makes it possible to update the firmware and license. Firstly, power the display, and click the pop-up message to access the configuration screen. The login information can be found in 8.1, under display configuration. After login, go to settings, network and either eth0 or eth1, depending on which port the ethernet cable going to the computer is connected to. Select manual mode in the drop-down menu, so the IP configuration, which can be found in Figure 28, can be manually introduced. When accepted, a reboot might be needed.

The screenshot shows a web-based configuration interface titled "Network Settings". At the top, there are four tabs: "eth0", "eth1", "br0 (bridge)", and "Proxy". The "eth0" tab is currently selected. Below the tabs, the following settings are visible:

- Mode: Manual (selected in a dropdown menu)
- MAC: 00:10:7E:09:68:40
- IP: 192.168.1.50
- Netmask: 255.255.255.0
- Gateway: (empty text box)
- DNS Server: (empty text box)

At the bottom of the configuration area, there are three buttons: "Back", "Cancel", and "OK".

Figure 28: Network settings on the display configuration. (Source: Author's own)

After successfully updating the network settings, the same display configuration can be accessed by connecting the display to the computer through the established port and introducing the same IP address in any internet explorer, but during the thesis practical framework, Google Chrome was used. The internet explorer selection should not affect the results in any way.

3.2.1 Firmware and licenses

In order to work with the latest version of Atvise, in this case, version 3.9, a display firmware update is needed. To do so, the first thing is to download the updated document from the official webpage, available at <https://customer.atvise.com>

In the download tabs, the file to download is the one ending with the download extension. In the case of the thesis, the file downloaded was named atvise-3.9.1-armhf.update. To get the license file, which has a lic extension, the hardware code must be given to the manufacturer, so they can provide the file. The hardware code can be accessed after login by going to settings, then the Atvise option, the code is found under the License tab, as shown in Figure 29.

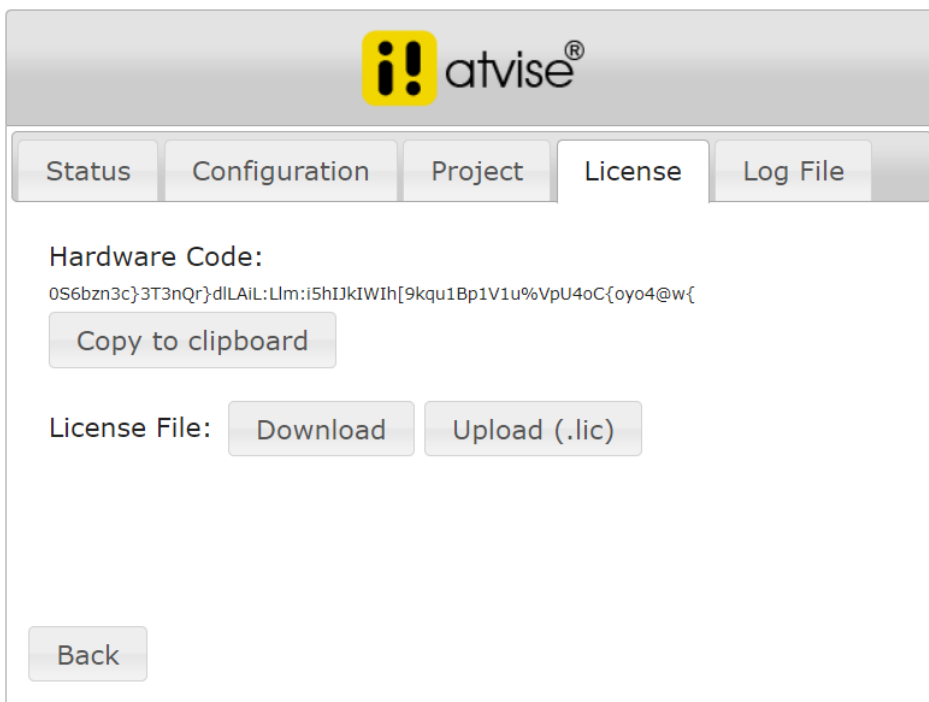


Figure 29: License tab in the display configuration. (Source: Author's own)

When both files are acquired, they can be uploaded to the display by using the display configuration through the computer. To upload the firmware file, on the first screen after login, go to update and click the button with the text 'Select Update File' and select the correct update file. To update the license, go to the same tab as to get the hardware code, settings>Atvise>License, click on the "Update (.lic)" button, found in Figure 29 above, and select the correct license file. After, the display should be working with the latest version of Atvise.

3.2.2 Synchronizing the server data to the display

The Advice software is license dependent. The workaround used in the first stages of the thesis was to use the free one-month trial after creating a new account. But when the licensed firmware is updated, the lifetime license is linked to the software. Therefore, the Atvise can be used freely if the display is connected.

The Atvise bundle installed has different software pieces: connect configuration, builder, monitor and database maintenance. The one used to configure the display client settings is the connect configurator. To avoid unexpected crashes, the software should be opened with administrator rights. The first screen show to big buttons, one for configuring over the PC and the other for configuring it over the network (Figure 30). The last is selected to configure the display over the ethernet cable. The cable is connected to the ethernet port 0 on the display and to a USB port in the PC by using an ethernet adapter. The network settings in the USB port on the PC are the same as those used for the ethernet port when configuring the PLC (8.1). Also, both devices are connected through the second ethernet port from the PLC to the ethernet port 1 in the display. This is done to establish server-client communication.

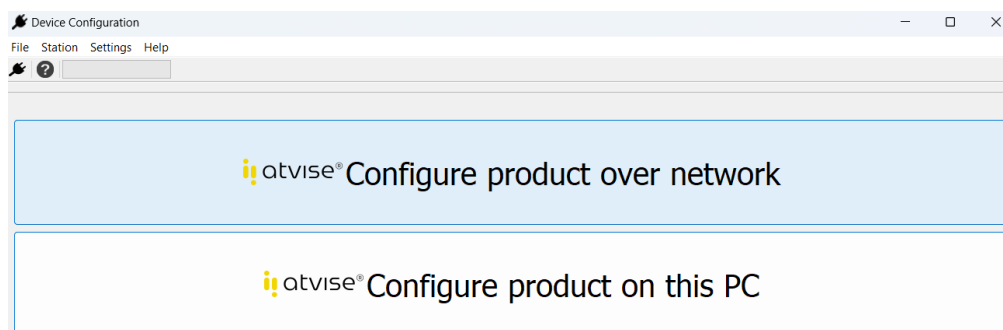


Figure 30: Location screen of the device to configure in Atvise connect configurator. (Source: Author's own)

After selecting the network option, if the device is connected to the PC it shows in the station selection screen with a blue font at the OT name (Figure 31). The destination port is the same as configured in previous sections, and it can also be found in 8.1.

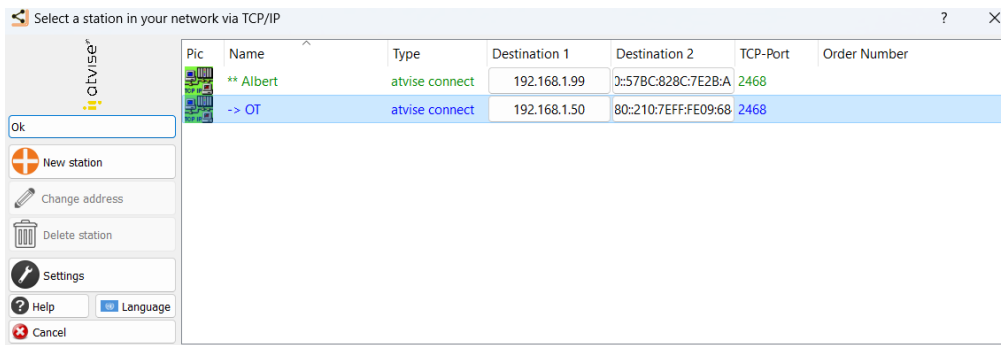


Figure 31: Display device selection in Atvise connect configurator. (Source: Author's own)

If the device cannot be connected by double-clicking on it, a right-click on the device and then selecting the connect option with the IP number should work. If not, the license should be checked. Then, the add icon, named 'Create a New Connection', opens the popup window for adding new connections to the project. To add the OPC UA server, the data server with the OPC UA option is selected (Figure 32).

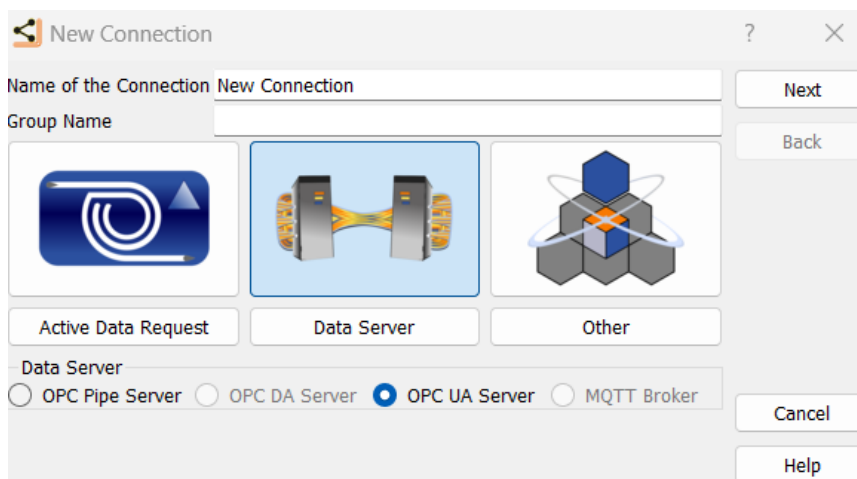


Figure 32: Adding the OPC UA server in the Atvise connect project. (Source: Author's own)

In the next two screens, the server configuration is set up. This server configuration can be found in both 8.1 and 3.1.3. Figure 33 shows that there is an established connection between the PLC and the display.

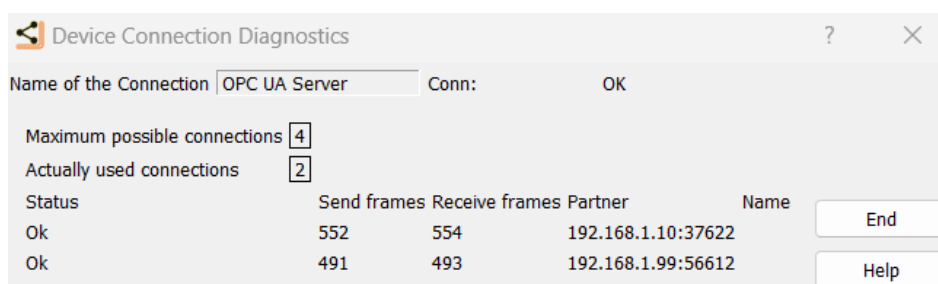


Figure 33: Connections status in Atvise connect with the PLC connected. (Source: Author's own)

4 Results

During this section, the results from the thesis will be obtained and analyzed. First, in Server results the PLC server is tested using both the external box and the UaExpert client. After ensuring the server functionality, 0 will discuss the possible methods to synchronize the server data with the Bachmann OT1210WM display.

4.1 Server results

To visualize easier the results for the PLC server, the external box was used. This device was connected to the PLC mount pins following a colour code for the cables. All the connections and the colour code list can be found in Appendix. For the Boolean values, the switches from Q1 to Q3 and the LED I1 were used. Therefore, as seen in Figure 34, if the switches Q2 and Q3 were on, LED I1 was set on too, following the PLC project logic explained in 3.1.2. In Figure 34 it can also be seen that the LED I3 is turned on but not at its maximum brightness. This is due to being connected to the A1 potentiometer, which is not fully turned. Briefly, Figure 34 shows the result of the working PLC project.

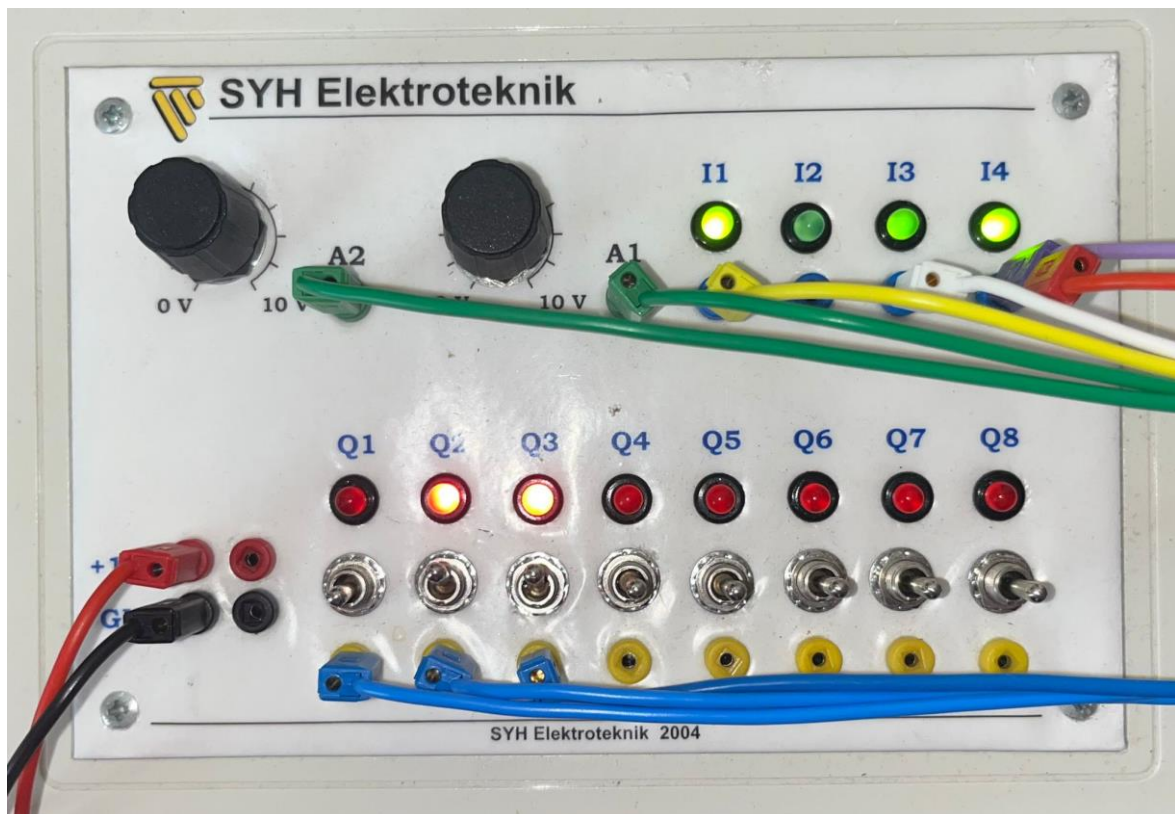


Figure 34: Photo from the external box connected to the working PLC. (Source: Author's own)

After validating that the PLC program is working correctly, the UaExpert is used to test the server. Following the steps detailed in OPC UA Server, the client is connected to the server. When the connection is secured, the address space section found in the bottom left of the software will show all the server data structures. The variables created in the PLC project are found after deploying five directories. Figure 35 shows the five open directories; Objects, DeviceSet, Codesys control for PLCnext SL, Application, programs and PLC_PRG that leads to the variables. To view them, they can all be selected at once and dragged to the data access view tab.

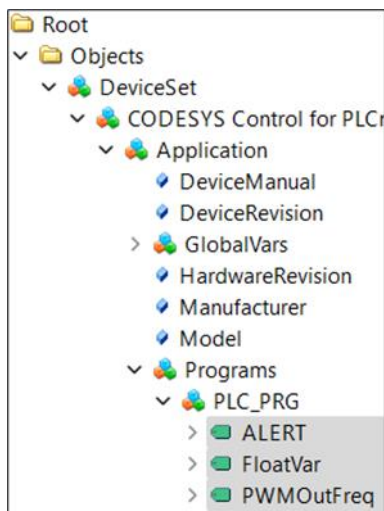


Figure 35: Location of the PLC variables in the UaExpert client. (Source: Author's own)

Figure 36 shows that the variables are sent correctly to the OPC UA server and that can be read in the client. These values can be compared with the ones shown in Figure 36, which are read from the PLC using Codesys by taking a screenshot of the PLC_PRG tab in tabular view while running. The results can also be compared with Figure 34, which was taken with the same values. Notice that the analogue values in the software might vary between the Figures since the screenshots were taken between a couple of seconds difference. This is due to the potentiometer's hysteresis caused by small changes in temperature and other external factors, causing a small value fluctuation. This difference is more noticeable with bigger values.

a)

Expression	Type	Value	Prepar...	Address
ALERT	BIT	FALSE		%IX0.0
digitalIn_01	BIT	TRUE		%IX0.1
digitalIn_02	BIT	TRUE		%IX0.2
digitalOut_01	BIT	TRUE		%QX0.1
analogueIn_1	INT	13150		%IW1
analogueOut_1	INT	13150		%QW1
StringVar	STRING	'6'		
PWMPulseDuty	BYTE	94		%QB6
analogueIn_2	INT	6		%IW2
PWMOuFreq	BYTE	6		%QB8
FloatVar	REAL	6		

b)

Server	Node Id	Display Name	Value	Datatype	urce Timestar	rver Timestan	Statuscode
OPCUAServ...	NS4 String ...	ALERT	false	Boolean	9:52:44.252 ...	9:52:44.252 ...	Good
OPCUAServ...	NS4 String ...	FloatVar	10	Float	9:52:47.553 ...	9:52:47.553 ...	Good
OPCUAServ...	NS4 String ...	PWMOuFreq	10	Byte	9:52:47.553 ...	9:52:47.553 ...	Good
OPCUAServ...	NS4 String ...	PWMPulse...	98	Byte	9:52:47.556 ...	9:52:47.556 ...	Good
OPCUAServ...	NS4 String ...	StringVar	10	String	9:52:47.556 ...	9:52:47.556 ...	Good
OPCUAServ...	NS4 String ...	analogueIn_1	13156	Int16	9:52:47.559 ...	9:52:47.559 ...	Good
OPCUAServ...	NS4 String ...	analogueIn_2	6	Int16	9:52:47.257 ...	9:52:47.257 ...	Good
OPCUAServ...	NS4 String ...	analogueOu...	13156	Int16	9:52:47.559 ...	9:52:47.559 ...	Good
OPCUAServ...	NS4 String ...	digitalIn_01	true	Boolean	9:52:44.257 ...	9:52:44.258 ...	Good
OPCUAServ...	NS4 String ...	digitalIn_02	true	Boolean	9:52:44.259 ...	9:52:44.259 ...	Good
OPCUAServ...	NS4 String ...	digitalOut_01	true	Boolean	9:52:44.259 ...	9:52:44.259 ...	Good

Figure 36: PLC variables data are shown in a) Codesys b) UaExpert client. (Source: Author's own)

The alerts in the OPC UA environment are called events. Thus, to test them an event viewer tab is added by clicking the add document icon in the taskbar and selecting the event view from the dropdown menu in the popup window. Dragging the Codesys control for the PLCNext SL option to the Server/Object space in the Configuration section on the event view tab will add all the alarms, hence events, to the client viewer.

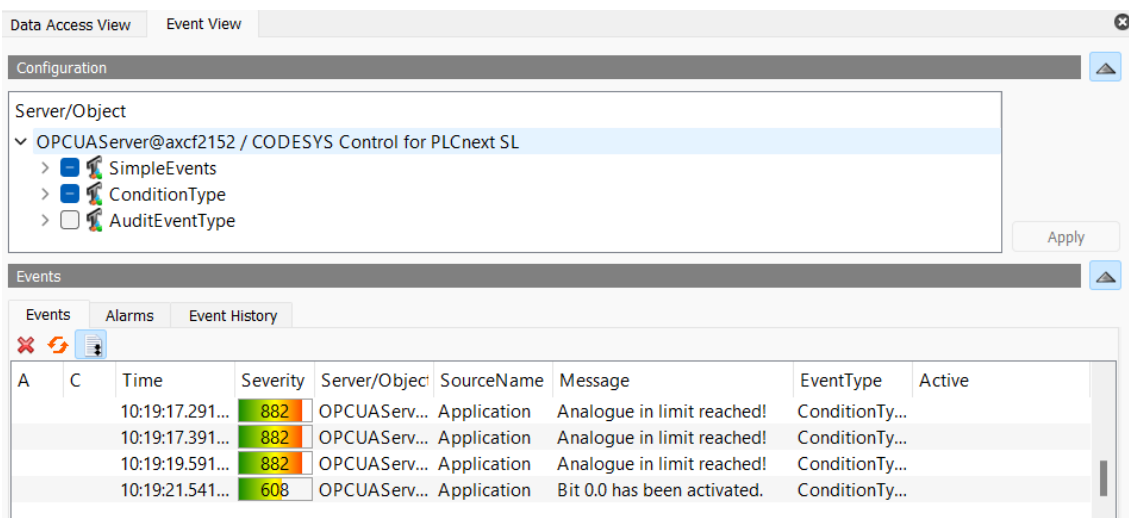


Figure 37: Events viewer in the UaExpert client. (Source: Author's own)

In Figure 37 it can be seen alarms are sent correctly when activated to the OPC UA server, and the client reads correctly the now-called events. It can also be seen that the severity changes. This is due to the fact that the alarms were sent to different classes with individual priority levels.

4.2 Client results

Following the steps explained in 3.2.2, the connection between the PLC server and the display client is established. In the UaExpert software, pressing the googles icon in the toolbar opens the variable list screen. There, a variable list is added by clicking on the add button found in the bottom left. With a new variable list, the pencil icon opens a new popup window that lets add new connections. Here, by the add connection options, an OPC UA client is configured. The server configuration is the same as the one used previously. Adding this OPC UA client shows a structure, but the variables configured in Codesys cannot be found (Figure 38).

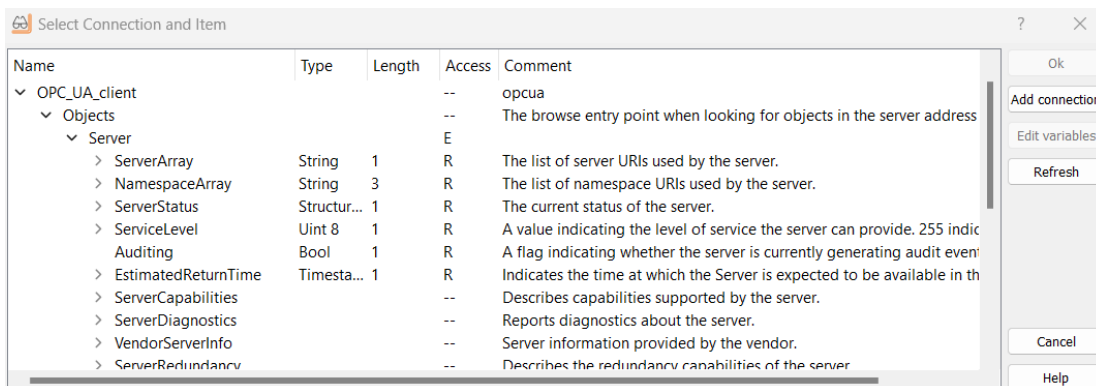


Figure 38: OPC UA client option variable list in Atvise connect. (Source: Author's own)

Therefore, a similar process was conducted in the Atvise builder software to test if the variables could be found there. After opening the software with administrator rights, the connection to the display server was established (8.1). The PLC connection can be added with a right click in Data Sources, and then add OPC UA data source. This Data Sources is found under the server and my server deployable options on the left side of the program (Figure 39: Adding the OPC UA source in the Atvise builder. (Source: Author's own)).

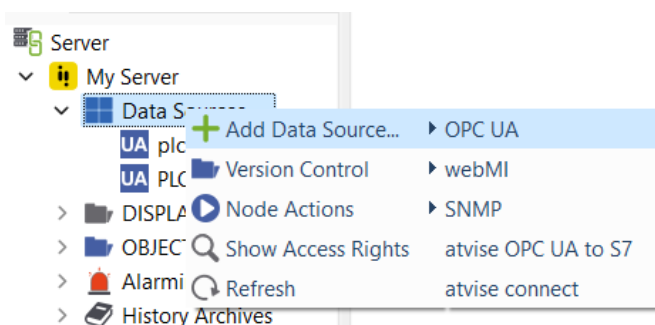


Figure 39: Adding the OPC UA source in the Atvise builder. (Source: Author's own)

In the new popup window, all the OPC UA server configuration is set up. When the connection is added, a double click in it, found under Data Sources, opens the variables browser (Figure 40). This is the same list as the one found in the Atvise connect (Figure 38), thus the variables from the PLC program cannot be found.

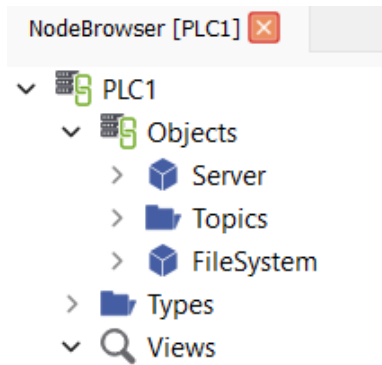


Figure 40: OPC UA server variable browser in Atvise builder. (Source: Author's own)

The next step would be to create the SCADA system for the display to read and write the OPC UA data, but since there was no method found of synchronizing correctly the server and the client, the SCADA system could not work. Therefore, it was not designed.

5 Discussion

In this section, the outcomes of the project will be discussed. First, in 5.1 the aims and objectives will be evaluated. 5.2 discusses the limitations encountered while the thesis process, and during 5.3 the consistencies and inconsistencies within the project are discussed.

5.1 Aims and objectives evaluation

The initial aim of the thesis, to engineer a system with one PLC acting as the server communicating with the OPC UA client in the operator terminal, has been partially achieved. This partial achievement is due to not accomplishing the objective of synchronizing the server data with the Bachmann display.

The first objective was to establish a working OPC UA server in the Phoenix PLC, which has been demonstrated in 4.1 with the use of the UaExpert client to see all the variables. This first objective goes in pair with the second objective, which consisted of creating a data structure with different types of variables, so the limitations of the devices within the OPC UA environment could be detected. As with the first objective, the second was also fulfilled.

But the third objective of synchronizing the data from the server with the client could not be accomplished, which lead to not achieving the final aim of the thesis. The reason behind this is that the method to read the server variables using the software provided by the manufacturer could not be found.

5.2 Project limitations

During the development of the thesis, different limitations have materialized, with some precluding to reach the thesis aim. Others had minor impacts on the outcome but could affect future thesis applications.

The most important limitation was related to the display, hence the client. Even though a connection between the server and the client could be established, there was no found method to synchronize the variables. This meant that the PLC could not be controlled by the display.

Another limiting factor with the display was the licensing for the software. This is linked to the hardware, so only if connected, the Atvise bundle can be used. A solution could be using the free trial provided by creating a new account. But this thesis was meant to lay the foundations of the environment to create laboratory tasks for future students, therefore, these students will only be able to work on the display if connected. And bearing in mind the cost of the display, which prices around 1600€ (8.2), could mean that the academic staff does not allow students to take it home, so tasks can only be developed in the laboratory.

5.3 Consistencies and inconsistencies with the project

The main idea of the thesis had shifted from the first iteration, which created some inconsistencies. One of the biggest inconsistencies is related to the connection between the server and the client. In sections 1 and 2, the concept of SCADA is introduced. But not achieving the synchronization objective led to not designing the human-machine interface program for the display.

On the PLC side, the main issue has been the licensing of the server certificates. The only method found to connect the clients to the server has been using anonymous login, which can be a serious privacy problem in real-world applications.

Regarding the design of the Codesys project, it was developed consistently due to the author's background in similar software, resulting in a simple but effective code for testing the OPC UA environment. Thus, the code cannot be adapted to uses other than testing data communication between different devices.

6 Conclusions

The emerging fourth industrial revolution is shifting the world of automation towards absolute connectivity where devices are permanently connected between them without any human intervention. This has created a need for the previous technologies and devices to adapt to this situation.

In the case of PLCs, most manufacturers preclude connecting devices from other companies. This thesis aims to lay the foundations in Novia UAS for the OPC UA environment, an architecture that allows the connection between devices from different manufacturers.

The thesis proposes a repeatable OPC UA environment consisting of one Phoenix contact PLC acting as a server and one Bachmann Electronics display acting as the client. In order to design an effective PLC project, Codesys software was used. The reason behind the software selection was its current use in university courses. The project included different types of variables and alarms to see how they worked under the OPC UA architecture. The simplicity of the code facilitated the server testing phase, which was conducted using the UaExpert software acting as an OPC UA client.

The Bachmann display was configured using the Atvise bundled software, but mainly the Atvise connector configurator to establish the connection to the server. Even though the client could be connected to the server and the connection could be visualized through the Atvise connector configurator, the server data could not be synchronized with the client display. This was due to a lack of information and knowledge in both the device and software. Therefore, the aim of the thesis could not be completely fulfilled and the connection between the PLC and the HMI could not be established as desired.

Future research could be conducted to establish the connection between the Bachmann display and the Phoenix PLC, but it could also develop a more intricate environment with devices from other manufacturers and multiple PLCs acting as clients. Another aspect that could be beneficial for the university to further develop would be to improve the server certificates to allow a more secure connection. Also, this thesis could be a guide for projects that require an OPC UA server.

7 References

Atvise. (2015). *The atvise Team*. Retrieved 2023, February 19 from <https://www.atvise.com/en/company/aboutus>

Atvise. (2023a). *Atvise builder: one tool for all*. Retrieved 2023, March 23 from <https://www.atvise.com/en/products/engineering>

Atvise. (2023b). *Atvise connect*. Retrieved 2023, March 23 from <https://www.atvise.com/en/products/expansions/atvise-connect>

Atvise. (2023c). *Atvise scada*. Retrieved 2023, March 23 from <https://www.atvise.com/en/products/atvise-scada>

AVEVA group. (2020). *Logic Guide - Using EN and ENO to Control when a Function is Performed (Built-in Functions, Function Blocks, and Conversions)*. Retrieved 2023, April 27 from http://gtg.ddns.net:8080/file/help_en-US/Content/LogicGuide/UsingENandENOtoControlwhenaFunctionisPerformed.htm#:~:text=Functions%20used%20in%20Ladder%20Diagrams,whether%20the%20function%20is%20performed.

Bachmann electronic GmbH. (2023). *OT1200*. Retrieved 2023, January 29 from <https://www.bachmann.info/en/products/ot1200-web-terminal>

cacamille3. (2022, March 31). *OPC-UA-Clients: List of commercial and open source OPC UA Clients*. Retrieved 2023, February 15 from <https://github.com/cacamille3/OPC-UA-Clients>

Chakraborty, S. (2022, March 1). *The Semiconductor Shortage Explained - Electronics & Semiconductors*. Retrieved 2023, January 23 from <https://blogs.sw.siemens.com/electronics-semiconductors/2022/03/01/the-semiconductor-shortage-explained/>

Codesys group. (2022). *OPC UA Server*. Retrieved 2023, February 19 from https://content.helpme-codesys.com/en/CODESYS%20Communication/_cds_runtime_opc_ua_server.html

Deane, P. (2011). *The First Industrial Revolution*. New York: Cambridge University Press, Retrieved from <https://doi.org/10.1017/S0022050700068509>

IEC. IEC 61131-3:2013 (2013). Retrieved from. <https://webstore.iec.ch/publication/4552>

iED Team. (2019, June 30). *The 4 Industrial Revolutions*. Retrieved 2023, February 3 from <https://ied.eu/project-updates/the-4-industrial-revolutions/>

- Knapp, E. (2011, December 31). *How Industrial Networks Operate*. Retrieved 2023, February 5 from https://www.researchgate.net/publication/301090604_How_Industrial_Networks_Operate
- Mohajan, H. (2019). The First Industrial Revolution: Creation of a New Global Human Era. *Uni Muenchen*, 5(4). https://mpra.ub.uni-muenchen.de/96644/1/MPRA_paper_96644.pdf
- OPC Foundation. (2017a, June 15). *What is OPC?*. Retrieved 2023, February 5 from <https://opcfoundation.org/about/what-is-opc/>
- OPC Foundation. (2017b, November 22). *UA part 3: Address space model - 8.2 NodeId*. Retrieved 2023, April 28 from <https://reference.opcfoundation.org/Core/Part3/v104/docs/8.2>
- OPC Foundation. (2019a). *UA Part 11: Historical Access - 3.1 Terms and definitions*. Retrieved 2023, April 28 from <https://reference.opcfoundation.org/v104/Core/docs/Part11/3.1/>
- OPC Foundation. (2019b, September 26). *Unified Architecture*. Retrieved 2023, April 28 from <https://opcfoundation.org/about/opc-technologies/opc-ua/>
- OPC Foundation. (2020, October 23). *OPC Classic*. Retrieved 2023, March 28 from <https://opcfoundation.org/about/opc-technologies/opc-classic/>
- Phoenix Contact. (2013). *AXC F 2152 - Controller – 2404267*. Retrieved 2023, March 10 from <https://www.phoenixcontact.com/en-pc/products/controller-axc-f-2152-2404267>
- Phoenix Contact. (2019a). *AXL F AI2 AO2 1H - Analog module - 2702072*. Retrieved 2023, March 10 from <https://www.phoenixcontact.com/en-pc/products/i-o-module-axl-f-ai2-ao2-1h-2702072>
- Phoenix Contact. (2019b). *AXL F DI16/1 DO16/1 2H - Digital module - 2702106*. Retrieved 2023, March 10 from <https://www.phoenixcontact.com/en-pc/products/i-o-module-axl-f-di16-1-do16-1-2h-2702106>
- Phoenix Contact. (2019c). *AXL F PWM2 1H - Special function module - 1007352*. Retrieved 2023, March 10 from <https://www.phoenixcontact.com/en-pc/products/special-function-module-axl-f-pwm2-1h-1007352>
- Pranowo, D., Bagastama, T., & Wibisono, T. (2020, June). *Communication between PLC different vendors using OPC server improved with application device*. Retrieved 2023, February 16 from https://www.researchgate.net/publication/342125873_Communication_between_PLC_different_vendors_using_OP_C_server_improved_with_application_device

Profanter, S. (2019, April 15). *OPC UA Address Space Explained*. Retrieved 2023, March 10 from <https://opcua.rocks/address-space/>

Rinke, A. (2022, May 6). *What is OPC UA? A practical introduction*. Retrieved 2023, March 2 from <https://www.opc-router.com/what-is-opc-ua/>

Rohrer, R. M., & Swing, E. (1997). Web-based information visualization. *IEEE Computer Graphics and Applications*, 17(4), 52–59. <https://doi.org/10.1109/38.595269>

Romero, V., & Theorin, A. (2012). *History of Control History of PLC and DCS*. Retrieved 2023, February 19 from [http://archive.control.lth.se/media/Education/DoctorateProgram/2012/HistoryOfControl/Vanessa Alfred report.pdf](http://archive.control.lth.se/media/Education/DoctorateProgram/2012/HistoryOfControl/Vanessa%20Alfred%20report.pdf)

Schneider Electric. (2019). *EN/ENO*. Retrieved 2023, March 10 from https://product-help.schneider-electric.com/Machine%20Expert/V1.1/en/SoMMenu/SoMMenu/CFC_Editor_Commands/CFC_Editor_Commands-5.htm

Schwab, K. (2016). *The Fourth Industrial Revolution*. Retrieved 2023, January 30 from <https://www.weforum.org/about/the-fourth-industrial-revolution-by-klaus-schwab>

Unified Automation. (n.d.). *Products*. Unified-Automation.com. Retrieved April 20, 2023, from <https://www.unified-automation.com/products.html>

8 Appendices

8.1 Important configuration parameters

Computer parameters: IPv4

IP Address: 192.168.1.99

Subnet mask: 255.255.255.0

PLC configuration

Gateway: 192.168.1.99

IP Address: 192.168.1.10¹

Subnet mask: 255.255.255.0

Username: admin²

Password: 2080daf5²

Display configuration

eth0 IP Address: 192.168.1.50

eth1 IP Address: 192.168.1.10

eth0 and eth1 Netmask: 255.255.255.0

OPC UA port: 4841

Username: root


Password: Bachmann



¹ It is the same for the OPC UA server

² Found in the PLC mount, different for each PLC

8.2 Components purchase information

Table 2: Purchase information for the equipment. (Source: Author's own)

Name	Manufacturer	Model	Price	General link	Datasheet	Picture
PLC	Phoenix Contact	AXC F 2152	550.00€* & **	https://www.phoenixcontact.com/en-pc/products/controller-axc-f-2152-2404267	https://docs.rs-online.com/f2e2/0900766b816b6f19.pdf	
Digital module	Phoenix Contact	AXL F DI16/1 DO16/1 2H	354.09€*	https://www.phoenixcontact.com/en-za/products/io-component-axl-f-di161-do161-2h-2702106	https://media.digikey.com/pdf/Data%20Sheets/Phoenix%20Contact%20PDFs/2702106_Ds.pdf	
Analogue module	Phoenix Contact	AXL F AI2 AO2 1H	365.00€* & **	https://www.phoenixcontact.com/en-pc/products/i-o-module-axl-f-ai2-ao2-1h-2702072	https://www.digikey.be/ht/mldatasheets/production/3134588/0/1/2702072-datasheet.html	



PWM module	Phoenix Contact	AXL F PWM2 1H	340.76€* & **	https://www.phoenixcontact.com/en-pc/products/special-function-module-axl-f-pwm2-1h-1007352	http://datasheets.shorotec.com/phoenix-contact/1422908.pdf	
Display	Bachmann electronic	OT1210WM	1627.00€***	https://www.bachmann.info/en/products/ot1200-web-terminal?rootCatId=351	https://www.atvise.com/media/1461/download/DB_200917_OT1200_Panels_EN.pdf?v=5	

*Different from the purchase price.

**Not in stock, price from retailers.

***See purchase information in Appendix 8.2.1.

8.2.1 OT1210WM purchase information

		TILAUSVAHVISTUS		1 / 2	
		Numero	Päiväys		
		3223	25.3.2022		
Laskutus: Ab Yrkeshögskolan vid Åbo Akademi PL 853 00026 BASWARE Y-tunnus: 2059910-2		Asiakas No Toimitustapa Toimitustieto Maksuehto Toimitusehto Toimitusaika	32246 Posti Ennakkomaksu EXW 15.4.2022 Noin 3 viikko		
Toimitus: Ab Yrkeshögskolan vid Åbo Akademi Puuvillakujja 3 65200 VAASA Y-tunnus: 2059910-2		Myyjä Viitteemme Tilausnumeronne Viiteenne Tilausmerkki Yhteyshenkilö	Mika Syrjämäki 2010 Joachim Böling Hans Linden		
Panelit atvisella					
Koodi	Määrä	Yks	à-hinta	Summa	
02	atvise OT1210 bundle	6,00	kpl	1 627,00	9 762,00
Operator Terminal: 10.1" (1280x800) Operator Terminal: 10.1" (1280x800); Projectiv-capacitive Multi-Touch; Freescale ARM® Cortex®-A9 i.MX6Dual+ (2x 1.2 GHz); 2 GB DDR3 RAM; 2x Eth 10/100; 2x USB2.0; 4GB eMMC onboard; CFast Slot; OS: Linux Embedded; Operating Temp. 0 .. +60 °C atvise® scada 150CCD included, limited to 2 Clients, 250 aggregates, requires OPC UA data source only atvise® connect small -Multi-protocol server that communicates with a variety of industrial protocols and provides an OPC UA server as centralized point for data access -Included in atvise®OT12xx bundles					
Elkome Oy Hellelörpankatu 37 05840 HYVINKÄÄ		puh 029 007 4410 info@elkome.com https://elkome.com/		Y-tunnus 0710837-2 Kotipaikka Hyvinkää	



TILAUSVAHVISTUS

2 / 2

Numero 3223 Päiväys 25.3.2022

Laskutus:
Ab Yrkeshögskolan vid Åbo Akademi
PL 853
00026 BASWARE
Y-tunnus: 2059910-2

Asiakas No 32246
Toimitustapa Posti
Toimitustieto
Maksuehto Ennakkomaksu
Toimitusehto EXW
Toimitusaika 15.4.2022
Noin 3 viikko
Myyjä Mika Syrjämäki
Viitteemme 2010
Tilausnumeronne Joachim Böling
Viitteenne
Tilausmerkki
Yhteyshenkilö Hans Linden

Toimitus:
Ab Yrkeshögskolan vid Åbo Akademi
Puuvillakuja 3
65200 VAASA
Y-tunnus: 2059910-2

Panelit atvisella

Koodi	Määrä	Yks	ä-hinta	Summa
-------	-------	-----	---------	-------

Muut ehdot

Takuuaika on 12 kuukautta laitteen toimituspäivästä.
Muilta osin laitteistotoimituksissa noudatamme ELKOMIT ry:n toimitusehtoja <https://elkome.com/cdfi>.
Ohjelmistojen ja työn osalta noudatamme tietotekniikka-alan yleisiä sopimusehtoja IT2018 YSE ja liitteitä sekä Elkomen palveluhinnastoa.

Omistusoikeus tavaraan siirtyy ostajalle, kun koko kauppahinta on maksettu myyjälle, Toimitussopimus purkautuu ja myyjä on oikeutettu hakemaan jo toimitetun tavaran takaisin, mikäli ostaja haetaan konkurssiin tai ei ole noudattanut, mitä tavaran maksamisesta on sovittu.

Muiden kuin Elkomen sopimustoimittajien Posti, Kiitolinja/Schenker ja UPS rahtisopimusten käytöstä veloitamme käsittelymaksuna 35 euroa.
Alle 100 euron tilauksista veloitamme pientoimituslisää 20 euroa.

Verokanta%	Veroton	Vero	Yhteensä
24,00	9 762,00	2 342,88	12 104,88
Yhteensä		EUR	12 104,88

Elkome Oy
Hellelorpankatu 37
05840 HYVINKÄÄ

puh 029 007 4410
info@elkome.com
<https://elkome.com/>

Y-tunnus 0710837-2
Kotipaikka Hyvinkää



8.3 OPC UA Built-in data types

Table 3: OPC UA Built-in data types. (Source: reference.opcfoundation.org)

ID	Name	Description
1	Boolean	A two-state logical value (true or false).
2	SByte	An integer value between -128 and 127 inclusive.
3	Byte	An integer value between 0 and 255 inclusive.
4	Int16	An integer value between -32 768 and 32 767 inclusive.
5	UInt16	An integer value between 0 and 65 535 inclusive.
6	Int32	An integer value between -2 147 483 648 and 2 147 483 647 inclusive.
7	UInt32	An integer value between 0 and 4 294 967 295 inclusive.
8	Int64	An integer value between -9 223 372 036 854 775 808 and 9 223 372 036 854 775 807 inclusive.
9	UInt64	An integer value between 0 and 18 446 744 073 709 551 615 inclusive.
10	Float	An IEEE single precision (32-bit) floating point value.
11	Double	An IEEE double precision (64-bit) floating point value.
12	String	A sequence of Unicode characters.
13	DateTime	An instance in time.
14	Guid	A 16-byte value that can be used as a globally unique identifier.

15	ByteString	A sequence of octets.
16	XmlElement	An XML element.
17	NodeId	An identifier for a node in the address space of an OPC UA Server.
18	ExpandedNodeId	A NodeId that allows the namespace URI to be specified instead of an index.
19	StatusCode	A numeric identifier for an error or condition that is associated with a value or an operation.
20	QualifiedName	A name qualified by a namespace.
21	LocalizedText	Human readable text with an optional locale identifier.
22	ExtensionObject	A structure that contains an application specific data type that may not be recognized by the receiver.
23	DataValue	A data value with an associated status code and timestamps.
24	Variant	A union of all of the types specified above.
25	DiagnosticInfo	A structure that contains detailed error and diagnostic information associated with a StatusCode.

8.4 PLC Code

```
PROGRAM PLC_PRG
```

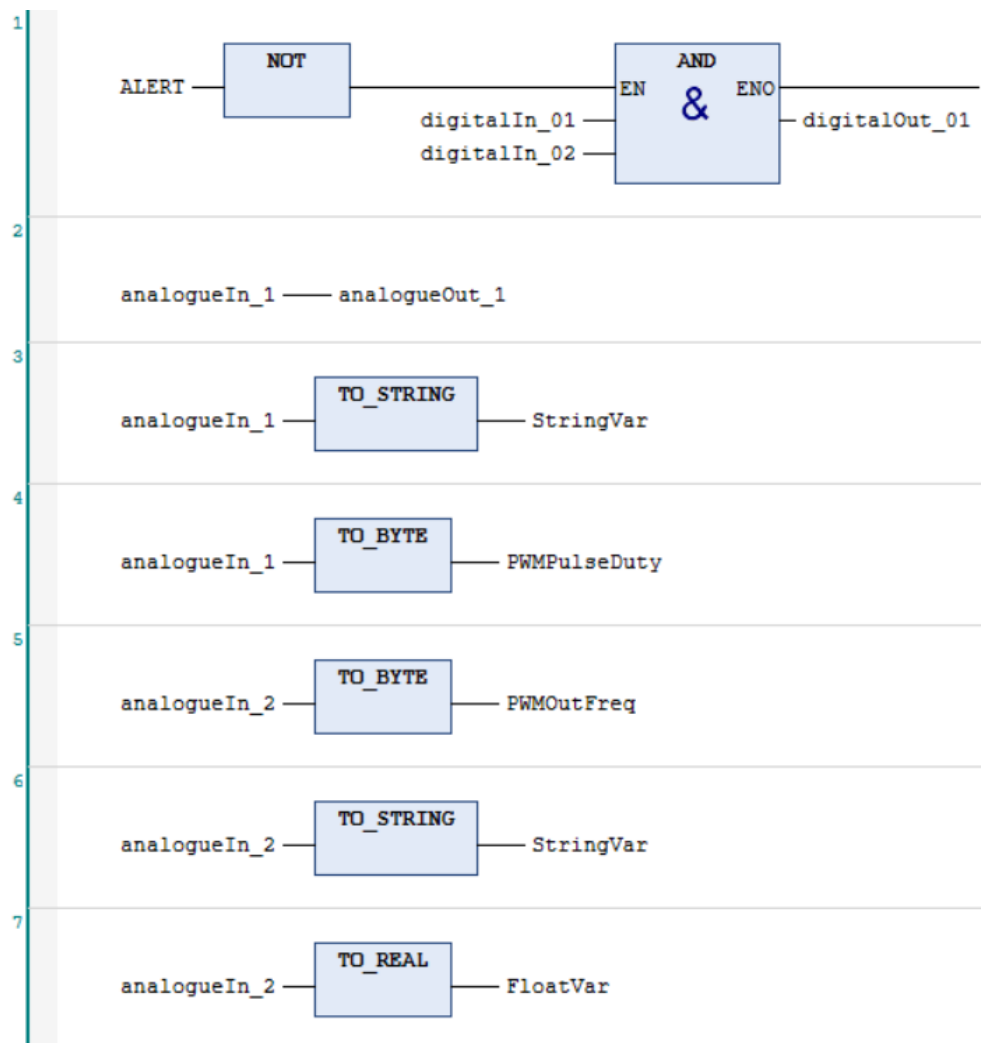
```
VAR
```

```

ALERT AT %IX0.0: BOOL;
digitalIn_01 AT %IX0.1: BOOL;
digitalIn_02 AT %IX0.2: BOOL;
digitalOut_01 AT %QX0.1: BOOL;
analogueIn_1 AT %IW1: INT;
analogueOut_1 AT %QW1: INT;
StringVar: STRING;
PWMPulseDuty AT %QB6: BYTE;
analogueIn_2 AT %IW2: INT;
PWMOutFreq AT %QB8: BYTE;
FloatVar: REAL;

```


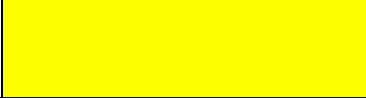




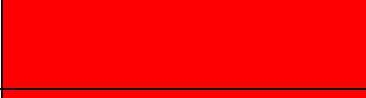
```
END_VAR
```



8.5 Connection between the PLC and the external box

To assist the testing phase and to not confuse the different signals, a color code for the cables was used, which can be seen in the table below. The external box is supplied from 24V DC from the PLC, and the rest of the connections are as follows:

- Digital inputs
 - Byte 0 pin 0 to switch Q1
 - Byte 0 pin 1 to switch Q2
 - Byte 0 pin 2 to switch Q3
- Digital outputs
 - Byte 0 pin 1 to LED I3
- Analogue inputs
 - Voltage 1 high to encoder A1
 - Voltage 1 low to the ground
 - Voltage 2 high to encoder A2
 - Voltage 2 low to the ground
- Analogue outputs
 - Voltage 1 high to LED I3
 - Voltage 1 low to the ground
- PWM outputs
 - 24V 1 to LED I4

Signal type	Color name	Color
Digital input	Blue	
Digital output	Yellow	
Analog input	Green	
Analog output	White	
PWM output	Purple	
Ground	Black	
Power supply (24/5 V)	Red	
Multimeter tester	Red	