

SCADA Development with OPC UA Standard

Practical study of developing a SCADA system in an HMI control and establishing communication to a server using OPC UA protocols

Oriol Mota Torras

Degree Thesis

Bachelor Thesis for Bachelor of Engineering

Degree Programme in Energy Technology

Vaasa 2023

DEGREE THESIS

Author: Oriol Mota Torras

Degree Programme and place of study: Electrical engineering and industrial automation at NOVIA

Specialization: Energy Technology

Supervisors: Joachim Boling and Philip Hollins

Title: SCADA Development with OPC UA Standard

Date: 23/05/2023

Number of pages: 104

Appendices: 3

Abstract

In the industrial automation sector, there is great variability of industrial components of different brands to perform the same action. However, there is a tendency in the industry to work only with a single brand despite higher costs. This is due to the difficulty of communication between devices from competing companies in the market. In an attempt to solve this problem, OPC UA appeared in 2008. An industrial communication standard that allows interoperability between devices and systems from different manufacturers and platforms. However, is this standard the solution for incompatibility in the industry?

This thesis was mainly intended to deepen how the OPC UA interoperability standard works. To do this, a SCADA system has been developed using the Atvise (atvise.com) software and an OPC UA server located in a Phoenix Contact industrial PLC has been operated remotely through a client.

The SCADA system that has been developed communicates the PLC server with a SCADA visualized on the user's computer internet browser, however, it has not been possible to implement this SCADA on the Bachmann industrial HMI. The system that has been created monitors a miniature industrial gate provided by NOVIA University as mechanical hardware. This door is controlled by the Phoenix Contact PLC on which an OPC UA server is installed using CodeSys V35 SP19 software (codesys.com). Next, the information from this server is transmitted to the user's PC simulating an OPC UA client by using the UAexpert program (uaexpert.com). As the final step of the process, using Atvise Builder and Atvise Connect, the writing and reading rights of the client variables have been obtained and a web SCADA has been generated in the internet browser of the user's PC. In this way, it has been possible to control the desired hardware through a SCADA system connected to an OPC UA server. In future stages of work, it is proposed to deepen on the functionality of the Bachmann HMI that is presented.

Key Words: SCADA, HMI or Control panel, Atvise software, communication protocol, PLC, OP UA client, OPC UA server.

Language: English

Table of Contents

| | | |
|-------|---|----|
| 1 | Introduction..... | 1 |
| 1.1 | Project purpose..... | 2 |
| 1.2 | Aims and objectives..... | 2 |
| 1.3 | Document structure..... | 3 |
| 2 | Background study..... | 4 |
| 2.1 | OPC Unified Architecture..... | 4 |
| 2.1.1 | What is OPC UA?..... | 4 |
| 2.1.2 | Basic functioning..... | 5 |
| 2.1.3 | OPC Foundation..... | 6 |
| 2.1.4 | Limitations of OPC UA..... | 7 |
| 2.2 | SCADA software..... | 8 |
| 2.3 | Difference between HMI and SCADA..... | 10 |
| 3 | Review of assembly equipment..... | 11 |
| 3.1 | Industrial Bachmann screen..... | 11 |
| 3.2 | Phoenix PLC..... | 13 |
| 3.3 | Atvise software..... | 14 |
| 3.4 | CodeSys Software..... | 16 |
| 3.5 | Controlled hardware..... | 16 |
| 4 | Methodology..... | 19 |
| 5 | OT1210 screen basics..... | 20 |
| 5.1 | License and software compatibility issues..... | 20 |
| 5.1.1 | OT1200 internal compatibility issues..... | 20 |
| 5.1.2 | OT1200 license configuration..... | 22 |
| 5.1.3 | Compatibility issues between OT1200 and PC..... | 24 |
| 5.2 | Using the TSSW browser..... | 25 |
| 5.3 | Final OT1210W installation..... | 28 |
| 5.3.1 | Updating OT1200 Atvise..... | 28 |
| 5.3.2 | License obtaining and installation..... | 28 |
| 6 | PLC configuration as an OPC UA server..... | 31 |
| 6.1 | PLC connection and basic programming..... | 31 |
| 6.2 | OPC UA Server PLC installation and client connection..... | 36 |
| 7 | Atvise functioning..... | 40 |
| 7.1 | Atvise PC initial configuration..... | 40 |

| | | |
|-------|--|-----|
| 7.1.1 | Installing Atvise..... | 40 |
| 7.1.2 | Licencing Atvise | 41 |
| 7.2 | Atvise builder functioning..... | 42 |
| 7.2.1 | Main screen | 43 |
| 7.2.2 | Datasource and Node concept | 46 |
| 7.2.3 | Display creation | 48 |
| 7.2.4 | Issues with programming in Atvise | 53 |
| 7.3 | Atvise connect functioning | 55 |
| 7.4 | SCADA visualization on HMI | 59 |
| 8 | SCADA development..... | 63 |
| 8.1 | Hardware behaviour | 63 |
| 8.2 | PLC programming..... | 64 |
| 8.3 | Client to SCADA communication | 73 |
| 8.4 | SCADA design and programming..... | 79 |
| 9 | Results..... | 92 |
| 10 | Discussion | 96 |
| 11 | Conclusions..... | 99 |
| 11.1 | Future work..... | 100 |
| 12 | References | 101 |
| | Appendix 1: OT1200 Bachmann Web Terminal datasheet | I |
| | Appendix 2: Phoenix Contact AXC F 2521 datasheet | VII |

List of Figures:

| | |
|--|----|
| Figure 1: a) b) SCADA example with atvise..... | 9 |
| Figure 2: SCADA -HMI system structure..... | 11 |
| Figure 3: Bachman OT1200 displays..... | 12 |
| Figure 4: OT1200 configuration menus..... | 13 |
| Figure 5: PLC Phoenix contact AXC F | 14 |
| Figure 6: Industrial gate prototype..... | 17 |
| Figure 7: Server version error in OT1210W TSSW visualization..... | 21 |
| Figure 8: Error with license file structure due to compatibility | 23 |
| Figure 9: Compatibility Error pop-up window on PC..... | 24 |
| Figure 10: Network settings for TSSW | 25 |

| | |
|--|----|
| Figure 11: a) PC settings location. b) IP settings location | 26 |
| Figure 12: IP settings for TSSW | 27 |
| Figure 13: TSSW main menu visualization..... | 27 |
| Figure 14: a) System information required. b) Hardware code required | 29 |
| Figure 15: Upload “.lic” license option in TSSW. | 30 |
| Figure 16: a) Network connections screen b) Ethernet properties settings | 32 |
| Figure 17: IP configuration for PLC connection..... | 33 |
| Figure 18: Steps for updating PLC Runtime Package..... | 33 |
| Figure 19: Correct connection and communication example | 34 |
| Figure 20: Ladder Diagram example in CodeSys | 35 |
| Figure 21: PLC output panel activation | 35 |
| Figure 22: Steps for initializing variables and an LD program | 36 |
| Figure 23: Adding Symbol Configuration option in CodeSys..... | 37 |
| Figure 24: Steps to be followed in Symbol Configuration Screen | 37 |
| Figure 25: Correctly installed OPCUA server certificate..... | 38 |
| Figure 26: OPC UA server login options in UAExpert | 39 |
| Figure 27: Final server directory menu | 39 |
| Figure 28: Server variable monitorization | 40 |
| Figure 29: Connect to server window | 42 |
| Figure 30: Atvise Builder main menu | 43 |
| Figure 31: Library window in Atvise Builder | 44 |
| Figure 32: Project Navigator window | 44 |
| Figure 33 : a) Atvise Builder toolbar. b) builder dropdown menu. c) Guided Action dropdown menu | 45 |
| Figure 34: Node types in Atvise Builder | 47 |
| Figure 35: Display editor window in Atvise Builder..... | 49 |
| Figure 36: Simple dynamics window Atvise Builder | 50 |
| Figure 37: a) Green button code. b) Green button dynamic | 52 |
| Figure 38: a) If condition in LD diagram in CodeSys. b) If condition as dynamic in Atvise.. | 54 |
| Figure 39: Atvise Connect main screen. | 56 |
| Figure 40: Station selection in Atvise Connect..... | 56 |
| Figure 41: Localhost Atvise Connect station | 57 |
| Figure 42: Variable list display in Atvise Connect..... | 58 |

| | |
|---|----|
| Figure 43: Main TSSW screen | 59 |
| Figure 44: Network settings for configuring the SCADA on the HMI | 60 |
| Figure 45: Browser settings for configuring the SCADA on the HMI..... | 60 |
| Figure 46: Port configuration for SCADA visualization on the HMI..... | 61 |
| Figure 47: Server status screen in TSSW | 61 |
| Figure 48: Pop up screen to connect to HMI server..... | 62 |
| Figure 49: SCADA visualization in the HMI..... | 62 |
| Figure 50: Variable list table in CodeSys | 65 |
| Figure 51: Emergency Stop signals in CodeSys..... | 65 |
| Figure 52: Variable list in Atvise Connect | 66 |
| Figure 53: Variables for SCADA buttons in CodeSys..... | 67 |
| Figure 54: Network 1 of the PLC program..... | 70 |
| Figure 55: Network 2 of the PLC program | 71 |
| Figure 56: Network 3 of the PLC program | 71 |
| Figure 57: Network 4 of the PLC program | 71 |
| Figure 58: Network 5 and 6 of the PLC program | 72 |
| Figure 59: Network 7 of the PLC program..... | 72 |
| Figure 60: Network 8 and 9 of the PLC program..... | 73 |
| Figure 61: SCADA variable list in UAexpert client | 73 |
| Figure 62: Localhost station in Atvise Connect | 74 |
| Figure 63: IP and gate configuration in Atvise Connect | 74 |
| Figure 64: Variables directory path in CodeSys..... | 75 |
| Figure 65: Variable list added successfully in Atvise Connect..... | 76 |
| Figure 66: Variable list with write status column updated | 76 |
| Figure 67: Data source setting in Atvise Builder | 77 |
| Figure 68: Variable directory in the Atvise Builder browse option..... | 78 |
| Figure 69: Copying and mirroring method in Atvise Builder project | 78 |
| Figure 70: Main display tree branch menu location | 79 |
| Figure 71: Final elements visualization in Atvise Builder | 80 |
| Figure 72: SCADA design evolution in 4 steps..... | 81 |
| Figure 73: Gate Control section of the final SCADA | 83 |
| Figure 74: Green button SCADA dynamic n1 | 84 |
| Figure 75: Green button SCADA dynamic n2 | 85 |

| | |
|---|----|
| Figure 76: Gate Status section of the final SCADA | 86 |
| Figure 77: Emergency Stop section of the final SCADA..... | 87 |
| Figure 78: Information Sate signs of the final SCADA | 88 |
| Figure 79: Emergency banner dynamic configuration of the final SCADA..... | 89 |
| Figure 80: Final SCADA visualization | 90 |
| Figure 81: Final SCADA developed | 90 |

List of Tables

| | |
|--|----|
| Table 1: General characteristics signal list from PLC program..... | 69 |
| Table 2: SCADA elements configuration table | 82 |

Glossary

- **HMI:** Human-machine interface
- **SCADA:** Supervisory Control And Data Acquisition system
- **PLC:** Programmable Logic Controller
- **OPC UA:** Open Platform Communications United Architecture
- **IoT:** Internet of Things
- **SOAP:** Simple Object Access Protocol
- **HTTP:** Hypertext Transfer Protocol
- **LD:** Ladder Diagram
- **FBD:** Function Block Diagram
- **IP:** Internet Protocol

1 Introduction

In the industrial components market, there is increased competitiveness and new competitors are emerging each time trying to gain a foothold in it. With a competitive purpose, producer design their componentry to communicate solely within the own product range. Although in principle this practice is carried out to ensure that a consumer is always faithful to a brand of products, this causes a great problem when the consumer wants to buy the cheapest one or when the brand does not have a product with certain specifications, since, if the consumer buys another brand, it will not be possible to establish communication and control said element.

Due to the aforementioned, companies in the industrial sector tend to only work with one brand, since the costs involved in buying everything from the same company are usually less than the costs of time and personnel required to be able to interconnect different brands.

In trying to deal with this problem, over the last few years, new communication standards have been created and defined that allow interoperability between the different industrial components. Of all the attempts that have been made to standardize communication across multiple vendors, one of the best and most widespread in the current market is the OPC standard.

This standard is supported by an association with more than 850 members, Association, according to its official webpage (OPC Foundation, 2015).

Its mission is to be a global organization in which users, providers and consortia work together to create data transfer standards for secure and reliable multi-vendor interoperability and platforms in industrial automation.

One of the most common tasks that an industrial engineer must face is knowing how to configure, program, and install the various equipment necessary for automotive projects. Due to the aforementioned, competitive purposes, companies tend to only work with one brand, since the costs involved in buying everything from the same company are usually less than the costs of time and personnel required to be able to interconnect different brands.

Despite the fact that the predecessors of OPC UA appeared more than 10 years ago, this standard is just beginning to appear in certain industries. This leads us to the next question, how easy and viable is it to establish a connection between components using OPC UA? Is it easier than focusing on a brand with its own software or are the steps to be taken much more difficult? Is it worth it for the main leader companies? The following thesis aims to provide a practical answer to these questions.

1.1 Project purpose

This thesis was commissioned by NOVIA and the university of Lleida, and the project has been proposed by Joachim Böling from the University of Applied Sciences NOVIA in Vaasa. The results of this study are intended to be of help to future students of the degree in energy technology by introducing them to some brand-new industrial procedures and standards. The results obtained will allow students to be introduced to the OPC UA interoperability standard. In the same way, it is intended to show the students with the results how it is to work with industrial screens and how they can affect the control of the PLCs in the plant.

1.2 Aims and objectives

This thesis aims to improve industrial automation knowledge and learn how to use the OPC UA interoperability standard while creating a SCADA that will be installed in an industrial control panel (HMI). This interface will allow the future user to control one or several industrial PLCs that will carry out basic control over some sensors and receivers. It is intended that this entire process can be transformed into a manual and an activity that future students who use these materials can take advantage of.

The equipment available and that will be used throughout the project are the following:

- OT1210 WM from Bachmann Industries.
- Phoenix Contact AXC F PLC.
- Atvise software (includes Atvise Connect, Atvise Builder, Atvise Terminal, and Atvise SCADA). Different versions has been used.

- CodeSys V35 SP19 software.
- NOVIA industrial gate.

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

- Prepare and update the laboratory equipment so that it can be adequately used by future students.
- Create an OPC UA standard communication server, from which we will connect the clients that will receive and send information to the server.
- Establish a communication flow between the screen, the personal computer, and the plc where projects and information can be transmitted.
- Design and install a SCADA software using the Atvise builder application. It does not have to be very complex, but it is enough to transmit the basic notions of what a SCADA is.
- Create a basic manual that allows future users of laboratory materials to be able to establish communication between them in the easiest and most practical way possible while learning to use OPC UA.
- Develop a practical laboratory activity for future students where they learn the basic notions of what is a SCADA, an HMI and what is OPC UA.

1.3 Document structure

Section 2 provides a basic survey of the topic to be covered (HMI, SCADA, OPC UA, etc.). Section 3 provides a review of the different equipment to be used. Then, section 4 explains in general terms the methodology carried out in this thesis. After, Section 5 then explains the configuration and update processes performed on the OT1210W display. Then, section 6 explains how to configure the PLC and install an OPC UA server on it. Next, in section 7, the operation of the atvise program is explained, and finally, in section 8, the design and programming process of the final SCADA is explained.

1.4 OPC Unified Architecture

In the below part of this section, an attempt is made to explain concisely what the OPC UA standard is for. It also tries to explain who its creator is and what story it has behind it, a very important factor to take into account when evaluating certain points for or against, which will also be seen. Moreover, a review of the basic OPCUA systems functioning is made.

1.4.1 What is OPC UA?

One of the most common tasks that an industrial engineer must face is knowing how to configure, program, and install the various equipment necessary for automotive projects. Due to the aforementioned, competitive purposed companies tend to only work with one brand, since the costs involved in buying everything from the same company are usually less than the costs of time and personnel required to be able to interconnect different brands.

The most general and easy to find definition to the question "What is OPC?" is OLE for process control. OLE stands for Object Linking and Embedding, this is nothing more than the communication protocol between applications that was used in MS Windows in the 80s (Techopedia, 2011). This methodology applied to process control is what is today known as OPC. However, the classic OPC only allowed to use this method in process control if the system software was Windows, that is why it was necessary to jump to a more global control system.

This is where the OPC UA or Open Platform Communications United Architecture intercommunication standard appears. Defined by SoftwareAG (OPC UA integration, 2022), OPC UA is an architecture independent of any platform and can integrate a large number of manufacturers into its standard. Said with more understandable words, OPC UA is a machine-to-machine, machine to server and vice versa communication protocol used for industrial automation and developed by the OPC Foundation. When using it, the user has the possibility of using components from different manufacturers (for example several PLCs of different brands) and being able to establish communication between them and the

server in coordination. Besides, OPC UA also works independently of the software being used (Linux, Windows, Android, etc.).

OPC UA allows manufacturers to take advantage of all the modern technology that helps create a smart factory: mobile devices, large databases, machine learning, vision and artificial intelligence, predictive maintenance, etc. According to the organization on its official website (OPC Foundation, 2019), given the large number of machines, devices and systems within an industrial facility, both for manufacturing and for logical operations, OPC UA is the solution to connect these isolated islands.

As points to highlight of OPC UA, the organization mentions its superiority with respect to the already widely used OPC classic, with a greater capacity to read and write data in a base with greater memory storage capacity (OPC Foundation, 2019). They also mention the different hardware platforms with which it is possible to operate (traditional PC hardware, cloud-based servers, PLCs, micro-controllers) and the operating systems that it supports (Microsoft Windows, Apple OSX, Android, or any distribution of Linux).

Furthermore, continuing with the points that the company highlights about its product, the architecture of OPCUA is made in such a way that new transport protocols, security algorithms, encoding standards, or application services can be incorporated into OPC UA while maintaining backward compatibility with existing products. Last but not least, OPC UA has a great information modelling capacity, which allows even large data structures to be adapted and transformed into information thanks to the UA standard.

From an objective point of view, all this information sounds very perfect, but at the same time really abstract, in such a way that makes someone who is not specialized in the area question. How does an installation with OPC UA actually work?

1.4.2 Basic functioning

As explained in the YouTube video " "Do you want to connect your industry? Use OPC UA." (Andrés, 2021), the main functionality of OPC UA is to interconnect the factory and the company. The connection to the different applications can be made through SOAP (Simple Object Access Protocol) for simple access connections or HTTP (hypertext transfer protocol) based on network communication. A key factor to highlight about UA is that the transport

layer mapping is independent of OPC UA services (messaging, information, and object modelling). This implies that if new information transport methods are defined in the future, OPC UA will be able to adapt to them and include them in its operation.

This protocol consists of a client and a server. The client requests information and the server provides it. An OPCUA server controls the data, the information, the processes, and the systems as if they were objects. Then, it presents these objects to clients in a way that they are useful for all sorts of very different applications called services. Clients request services from OPC servers and receive the responses they send back.

Continuing with what can be seen in the YouTube source ([¿Quieres conectar tu industria? Usa OPC UA, 2021](#)), the chronology of operation and connection of the client-server relationship is as follows: First, the client searches for the server using the method with which it has been configured. Then, the client gets a list of available endpoints and selects one that supports the security and transport profile that meets the application's requirements. To then access the server, the client creates a channel (authenticated connection between 2 devices) and a session (authorized connection between client and server). Once these points are established, the client is already able to send notifications to the server, receive changes in the value of the data and any results of the programs executed by the servers.

1.4.3 OPC Foundation

As previously mentioned, the OPC UA communication standard is the result of a consortium of more than 850 companies around the world, these form what is known as the OPC Foundation (OPC Foundation, 2015).

The origin of this organization comes from de the mid-90s, at that time, the industrial automation market was dominated by Microsoft COM and DCOM (intercommunication standards). Automation vendors Fisher-Rosemount, Intellution, Opto 22 and Rockwell Software joined forces to create a data access standard based on Microsoft COM and DCOM. This standard would be called OPC, known today as OPC classic (OPC Foundation, 2022).

As explained by the organization on its [official website](#), after a year from the beginning of the project, and seeing its success, other software and industrial hardware vendors joined the project. The need was soon seen for a larger organization that could coordinate, certify, and validate interoperability between vendors. This is how, the OPC Foundation was officially born on April 22, 1996.

During the following years and up to the present, the organization has been growing with new members at the same time that the OPC standard has been improving. Continuing with all this growth, the need for OPC to be intraplatform and not just with Windows was seen. For this reason, in 2004 the organization released the first version of OPC UA, a version that offered all the possibilities that OPC Classic had, but that allowed it to operate from any of the existing software on the market (OPCconnect, 2023). During the following years, the interoperability standard continued to be improved and updated until it reached what we know today.

The last data that the organization gives us regarding the number of members is from September 2022. The date on which the organization has the presence of 895 companies from the industrial sector. In its [list of members](#) are manufacturers such as Allen Bradley, Siemens, ABB and Rockwell, among others.

1.4.4 Limitations of OPC UA

Now that the advantages and benefits of using OPCUA for process control are known, it is necessary to be aware of the negative points that it can have. According to the information provided by the video “What is OPC?” from (4.0 Soutions, 2021), OPC UA will not be the future of IoT communication. One of the reasons for that is because the OPC UA standard has too many remnants of the classic standard. This is called technical debt, and this makes engineering job really difficult to make the integration of the standard.

The second problem mentioned in the video is that OPCUA has lies in the foundation that coordinates it, even though the foundation’s mission is for interoperability of industrial systems, in practice what they’re doing is allowing the corporations who help write the standard to make it for them to sell their products and then, steer the customers towards their solutions. Not to make it easy for their solutions to interoperate with one another.

Furthermore, not all products are fully compatible with OPC UA. Said with more understandable words, a person could think that by buying 5 different products from any of the companies present in the list of OPC members, it would be possible to interconnect said products. The question is that this is not always the case, sometimes even though the product has the OPCUA label, it only uses a specific part of the said standard and it can happen that even though the other product also has the label, it is using a part of the standard completely different.

However, this does not mean that OPCUA is a bad standard or a technology that we should not use. The advantages that it provides are many and very practical, breaking thanks to this the limitation of using a single industrial vendor in the facilities around the world at the same time that it offers fast and effective communication between various parts of the process control.

2 Background study

2.1 SCADA software

Usually, a person who is not specialized in the sector would think that a SCADA is just a screen with some information and buttons capable of controlling a system. The reality is that although it may seem something as trivial as that, the term SCADA refers to the system that receives operational data about a process and the associated equipment to control and optimize said process (AVEVA, 2021). With these data received, the SCADA is capable of allowing the user full control of the system while keeping the user informed of changes and the current state of the process. That is, a SCADA is capable of monitoring one or more industrial production systems or any automated system.

SCADA automation is simply the method to achieve a certain final product, and not the product itself. All industries in the sector agree on the need to maximize the performance of production systems by perfecting this type of system as much as possible. In the industrial automation market, staying competitive means continually finding new ways to operate faster and with fewer resources (SCADA international, 2023). There is always a pressure to increase productivity, efficiency, agility, quality, and profitability, while

minimizing costs. SCADA systems provide rapid interaction with the plant user while ensuring fast and efficient production.

To better understand what a SCADA is and to know what is intended to be developed during this thesis, below, in *Figure 1* are some examples made by using Atvise SCADA software. In such a way that it will be seen visually and exemplified that it is a SCADA



Figure 1: a) b) SCADA example with atvise. (Source: <https://www.teslakontrol.com>)

An important fact to mention is that these examples have been made by the company "TESLA measurement control systems environmental technologies". To make these SCADA systems the Atvise software has been used and they have also been implemented in control screens of the Bachmann brand (TESLA measurement control systems environmental technologies, 2023). This implies that these SCADA examples are perfectly replicable and installable in the current work equipment available.

2.2 Difference between HMI and SCADA

Commonly a SCADA System can have one or several HMIs, as this is the most visual part people use to confuse one with the other. Their differences are briefly explained in this section of the thesis.

In one hand, a SCADA system, as explained before, is a combination of various devices (PLCs, HMIs, sensors, actuators, communication networks, etc.). The data from these devices is sent and monitored by the SCADA on the main screen that governs the entire system. Once this information is received and processed, the system or the operator can send data to the system to control their PLCs and actuators.

On the other hand, as explained by a mechatronics expert in the video (Harb Mecatrónica, 2020), an HMI (Human-Machine Interface) is basically a user control device, normally it comes in the form of a touch screen with some programmable buttons. Their sizes are variable and adapt to each installation. Each HMI will have its own graphics galleries and its communication protocol with which it works, these variables will always depend on the manufacturer of the product. Commonly, HMI is located in certain machines or specific processes, which have their own PLC that controls the process in question. In the HMI the people or the controllers of the process can visualize data, transmit orders, and obtain results or conditions of the respective process or machine. In this way, they can control certain functions through the HMI in the field.

The following *Figure 2: SCADA -HMI system structure* (Source: modified from <https://www.youtube.com/HarbMecatr>) shows an example of a SCADA structure, in this, the HMIs, PLCs and SCADA are identified. The image shows an example of a plant control in which 3 different processes are involved. Each of the processes has its own HMI that monitors certain actuators and sensors. These three processes are globally coordinated by SCADA, which is located at a central control point, such as the engineer's office.

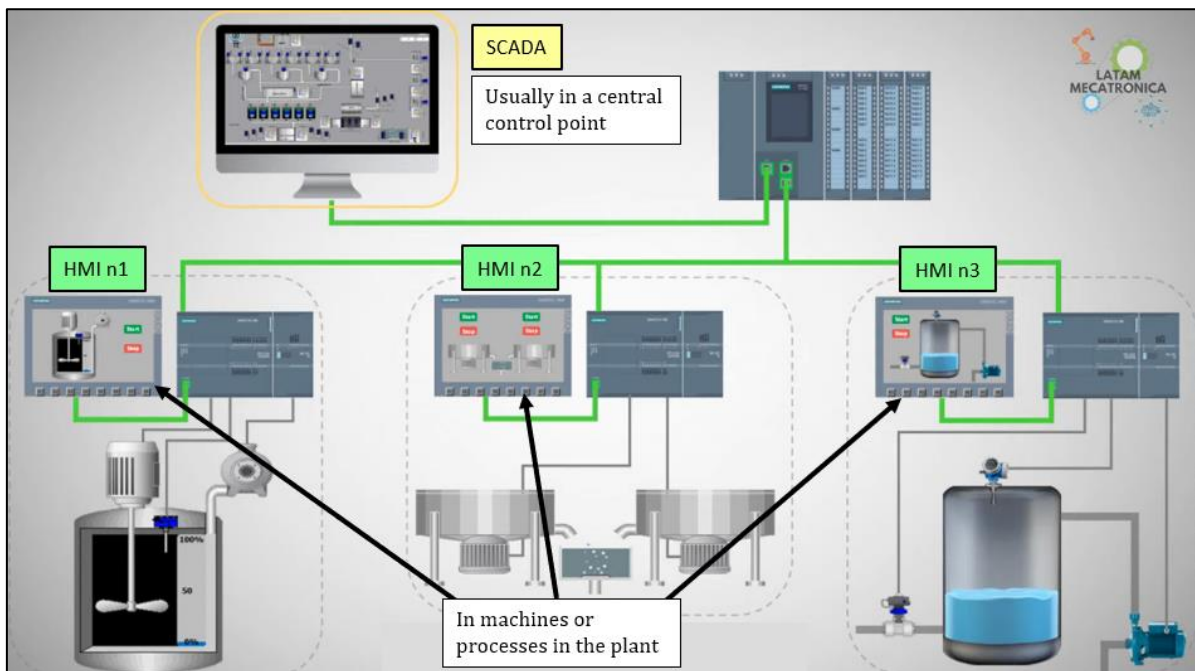


Figure 2: SCADA -HMI system structure (Source: modified from <https://www.youtube.com/HarbMecatr>)

Therefore, it is seen that the two concepts are different. In conclusion, it has been seen that one or several HMIs may be part of a SCADA, however, a SCADA will never be part of an HMI.

3 Review of assembly equipment

Before beginning with the practical development of the thesis, it is considered opportune to carry out a review of the characteristics and limitations of the different equipment that will be used during the course of the project.

3.1 Industrial Bachmann screen

The OT1210WM screen is an HMI (Human Machine Interface) model distributed by the company Bachmann SL (<https://www.bachmann.info/en>). Despite the fact that the model can be distributed with Windows or with Linux, the model with which it will work will have Linux exclusively.

As explained by Bachmann on its website (Bachmann Visutec GmbH, 2023), the terminal stands out for having a simplified software that makes its configuration much easier.

However, in case the customer desires to develop more complex systems, the screen also has the possibility of importing updates to increase the possibilities. Also, Additional useful services and software packages are preinstalled on the operating system in order to reduce the development time for common application scenarios. The physical appearance of the OT1200 display models is seen below in *Figure 3*. It should be said that the model is named OT1210W due to the 10 inches that it measures.



Figure 3: Bachman OT1200 displays. (Source: <https://www.teslakontrol.com/en/ot1200-hmi-scada-set>)

All the information related to its technical specifications can be found on the [Bachmann webpage](#) or in (Appendix 1: OT1200 Bachmann Web Terminal datasheet) of this thesis. The datasheet presented in this section gives the physical and electrical characteristics of the OT1200 series displays.

Installed software

Inside, the screen contains software adapted from Linux. In this, the user has the possibility of accessing a branch of configurations adapted to establish different types of connections in the industrial field. Outside of this configuration, the screen has a Bachmann internet browser that allows connecting to the server to the screen. The configuration branch offered by the web terminal is shown below in the *Figure 4*.

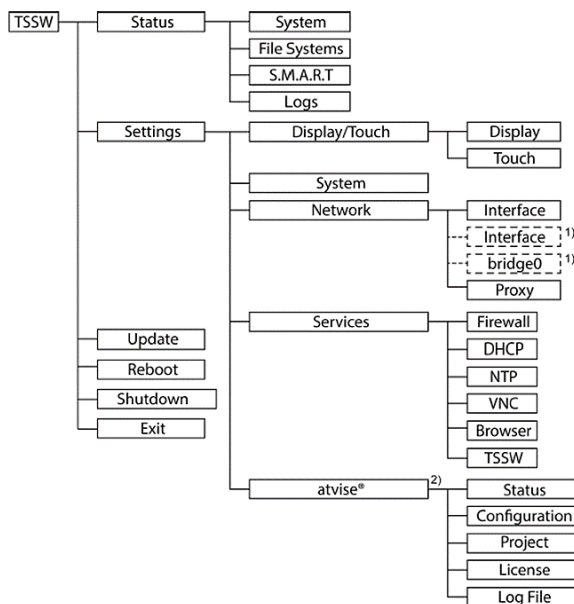


Figure 4: OT1200 configuration menus (Source: User manual System software Linux V1.52)

3.2 Phoenix PLC

What is a PLC?

The letters PLC stands for Programmable Logic Controller. PLCs are one of the cornerstones of all industrial automation. As explained by the company inductive automation in its article "What is a PLC?" (Inductive Automation, 2020), these devices are industrial computers that have the capacity to compute logical information and control the different installations and electro-mechanical processes that exist in the industry or other automotive sectors.

A person who does not know about the subject could think, what is the difference then from a normal computer? As explained in the official forum of the electronics company (Electrónica Edimar, 2020) The difference between these is the simplicity and reliability that the PLC offers us. The information that it receives is given by several simple channels such as sensors or meters of certain parameters related to the process they control. The PLCs also guarantee full and uninterrupted functionality at maximum performance, something very important in the industry, where production is uninterrupted.

Phoenix Contact AXC F 2152 model

In the official datasheet of the Phoenix contact product, it is explained to the customer this PLC model has two ethernet interfaces. This PLC boasts of being fast and very reliable, in

addition, it is robust and very easy to use. According to the company (Phoenix Contact, 2013), it is perfectly designed to be subjected to maximum performance in the different scenarios of the industrial and automation sector. The product specifications as well as its technical data can be found in the official [Phoenix Contact datasheet](#) on the internet or in Appendix II of this thesis.

Next, in *Figure 5* an image of the PLC integrated into the said physical interface is shown. A factor to also consider and mention is the fact that this PLC, provided by NOVIA University, has a physical interface that allows us to see the different digital inputs and outputs that the PLC has in a labelled and orderly manner. Also visible are the ports that it has for analog inputs and outputs, pulse amplitude modulation and finally, power supply.



Figure 5: PLC Phoenix contact AXCF. (Source: Author's own)

3.3 Atvise software

Atvise is a visualisation and monitoring software for industrial automation developed by the German company Bachmann. The software allows communication with control and monitoring devices such as PLC (Programmable Logic Controllers), SCADA (Supervisory Control and Data Acquisition) and OPC UA (Unified Process Architecture).

As explained by Vester Business (Vester Business, 2021), Atvise is known to be a very flexible and scalable visualisation software, which makes it suitable for use in automation projects of any size. The software is also characterised by its high visualisation quality and the ease of use of its development environment.

In addition, Atvise has a standards-based approach, which means that it supports a wide range of communication protocols and can be integrated with other third-party automation systems. This makes the software very useful in complex and highly heterogeneous industrial environments.

In summary, Atvise is a highly flexible and scalable industrial automation visualisation and monitoring software tool. Atvise enables communication with control and monitoring devices and focuses on visualisation quality and ease of use.

Atvise is not just a single piece of software, it is a set of different programs that work together in a coordinated way to communicate and work with PLCs and servers in a coordinated way at an industrial level. Among all its subprograms and extensions, Atvise Connect, Atvise SCADA, Atvise Embedded, Atvise, Atvise Builder, Atvise HMI and Atvise Monitor should be highlighted. Among the above, although they were all used for the development of the thesis, most of the functionality is centred around Atvise Builder (dedicated to the design of SCADA) and Atvise Connect (to connect servers and components to our Atvise server).

The main functionality of the program is focused on designing its SCADA themes that can be implemented in industrial HMIs connected as a server. Industrial-level connections with Atvise can be cascaded, centralized or global. On its website (atvise - HMI + SCADA, 2023), Atvise explains some of the best general features of the software, among which are the possibility of running both on Linux and Windows, the great flexibility of functionality and operation of Industry 4.0, the ease of installation and continuous updating and finally the freedom from restrictions that working with OPC UA offers today.

In our particular case, it is important to mention the fact that the OT1200 web terminal used for the thesis works in collaboration with Atvise. This allows Bachmann to develop their panels with a minimal version of Atvise that will allow us to export and import projects to the screen.

3.4 CodeSys Software

The CodeSys operating system is a program for the connection and communication between industrial PLCs and personal computers with classic operating systems. The program's configuration allows it to select the input gateways for each device and modify its online configuration and communication settings (CODESYS, 2023). This will allow the user to communicate and configure the PLC in the most convenient way for the project that is intended and adapt it in the most optimal way for the work that will be developed.

The program also has an extensive catalogue and libraries to be able to specifically identify the industrial PLC model with which it will work. In addition, it has different forms of programming at a high and low level, some examples are structured text, function block diagrams, contact diagrams, etc.

This program gains weight in the development of the thesis due to the communication and connection difficulties that Atvise presents. With the help of CodeSys, it is intended to be able to regulate this communication and implement some basic program to thus establish the PLC in an active state and thus improve the ease of detection by Atvise.

3.5 Controlled hardware

Before starting the discussion and explanation of the hardware that has been chosen, it is important to understand what element of our project this is. When talking about hardware during the development of the thesis, it refers to the set of actuators, sensors, and devices that will receive and send data and information to our PLC. This could be anything from switches and LEDs to turbine motors or much more complex mechanized elements. All this information and all this equipment are intended to be controlled from the HMI screen, where the status of these devices and their operation will be always seen.

One of the biggest problems is to determine what hardware is going to be controlled with the developed SCADA. Deciding this may seem like something that should be done in the advanced stages of the project; however, it is necessary to have a defined idea of what is going to be controlled in order to know if the limitations of the system fit what is intended to be developed. In the event that the chosen hardware has a greater number of inputs

and outputs than the system is capable of processing, or that the software is capable of storing, the hardware cannot be chosen, and other hardware options should be considered.

Although it may seem very difficult to run out of capacity in the software. The limitations of our particular project must be considered. After talking several times with the license provider company, they have agreed to provide us with only temporary free licenses for a maximum duration of 1 month. In addition to the time expiration, these licenses are limited in terms of the maximum number of nodes or variables that can be processed at the same time (AtviseCustomerService, 2023). As will be explained more extensively in future sections of the thesis, the nodes or variables are elements processed internally in the software to activate a button, save a memory, a counter, etc. A simple button on the visualization can require 6 or 7 nodes to activate colour, memory, shape, activate the next step, etc. This established node limit is certainly low to prevent large companies from taking advantage of these free licenses.

With all these concepts and limitations clarified, then we go on to choose and explain the chosen hardware.

Industrial gate

Finally, by evaluating all the possibilities and limitations, the equipment provided and prefabricated by NOVIA University has been chosen. This equipment, which can be seen in *Figure 6*, consists of a scale simulation of an industrial warehouse gate. This gate is normally used as teaching material in the electronics and PLCs courses taught at NOVIA University. Normally it is used in coordination with a certain Siemens PLC model, in the case of the development of this thesis it will be implemented with a different PLC (Phoenix Contact). Due to its simplicity, problems are not considered depending on the software or PLC used in each of the cases.



Figure 6: Industrial gate prototype (Source: Author's own)

The equipment in question provides the user with a total of 8 bits of information. Of which 4 are inputs to the PLC and 4 are outputs. These 8 signals allow to establish a complete control of the system.

Regarding the inputs to the PLC, there are two buttons (one green and one red) and two limit switches located in the fully open and fully closed position of the door. Listed, the inputs are the following:

- The green button gives a signal that it is pressed.
- The red button gives a signal that it is pressed.
- Switches indicate that the door is completely closed.
- Switches indicate that the door is completely open.

As for the PLC outputs, there are two motors that drive the movement of the door. In addition, a led and an alarm that can be activated by using the pertinent hardware inputs. Listed, the PLC outputs are the followings:

- Control the motor to open the door.
- Control another motor to close the door.
- Control the LED light.
- Control the alarm tone.

4 Methodology

This thesis has been carried out in the following way:

First of all, it is intended to prepare all available equipment, this process consisted in update and configure the various PLCs, the industrial monitors and the PC programs available. In order to do this, research of the information available in the various tutorials and manuals available on the official web pages of each product has been carried out.

Once all the equipment is fully operational, the SCADA creation process will begin by creating an OPC UA server and placing it in the PLC, this process is explained step by step in section 6.0 (PLC configuration as an OPC UA server) of this thesis. For the configuration of the PLC as a server, the CodeSys V35 SP19 software will be used. Using this same application, a contacts and coils (LD) program that allows controlling the industrial hardware of our system will be designed. This program must function correctly in local mode, at the same time, this program will take into account that the system must be operated remotely with SCADA.

From this server, the variables will be obtained in an OPC UA client created locally on the user's PC using the UAexpert application. Then, the variables will be exported to the Atvise SCADA creation software by using the Atvise Connect software. This next process is explained in the next section 7.0 of the thesis (Atvise).

In this same section, connecting to the explanation of the SCADA design program. It is showed to the reader how to configure, install, and activate atvise licenses. The main functions and the most used menus are also discussed. Finally, the tools that have been used for the creation of the final SCADA are presented. To show the reader how to use the SCADA creation software, it is intended to explain the Atvise software and discuss its main uses and functions. This will be also made in section 7.0 (Atvise).

With the variables finally available in Atvise, a SCADA will be designed and programmed to monitor and control the industrial door provided by NOVIA University. The entire server creation process as well as the SCADA design and programming is exposed in the last section of the thesis 8.0 (SCADA development), which describes the particular case developed in this project.

5 OT1210 screen basics

The following section shows some basic concepts that must be taken into account in order to ensure that the Bachmann OT1210W display works according to the user's wishes. This section is very important because of the many problems and settings that need to be considered to get the display fully operative. The obstacles that have been encountered during the course of this thesis and the solution chosen for each of them are explained in general terms.

This section also shows how to establish the TSSW method for the remote configuration of the Bachmann OT1210W industrial screen. This method provides the user, among other things, access to the settings of the different communication protocols that the screen has. It also allows the user to remotely navigate the display and access internal files located in the display's Linux software. This will be used later (in sections 5.3.1 and 5.3.2) to configure and adapt the display to the needs of this thesis.

5.1 License and software compatibility issues

One of the main challenges faced by users trying to use and configure OT1200 displays is the issue of compatibility between the pre-installed factory version of Atvise, the installed operating system version, the device's licensing structure, and the program version on the PC.

5.1.1 OT1200 internal compatibility issues

The software version is strictly linked to the Atvise version, and the display can only function properly if both versions match in the configuration of the OPC UA server that includes the display. It is important to mention that despite version 3.4.4 of Atvise and version 1.29 of Linux are outdated and in the future without technical coverage, it has been possible to make the displays function properly and work with them, which is the configuration they come from the factory.

Initially, the screens come with Atvise version 3.4.4 installed, this version is somewhat outdated and was created in 2019. Furthermore, as explained in the release notes from Atvise 3.4 version (Bachmann Visutec GmbH, 2019), four years after the release date of

each version of the Atvise software the company establishes the end of maintenance and support for this version. In the case of Atvise 3.4.4 present in OT1200 displays, this will be at the end of 2023, the year in which this thesis is being written.

Regarding the display software, it comes with a pre-installed Linux software version 1.29. By following the steps explained in its manual (Bachmann Visutec GmbH, 2021) is it possible to update the software to Linux 1.52 by using internal files contained in the screen, this means that no internet or external connection is required. The problem is that this new Linux version will not be compatible with the preinstalled Atvise 3.4.4. Linux is also updated automatically when updating the atvise version.

In case these versions do not match internally, compatibility errors occur when trying to start the display server, which immediately stops the server when trying to start it. In the following image (*Figure 7*) it can be seen the corresponding error in case the Atvise version is 3.4.4 and the database (Linux version) is 1.52 or version 9 of the database:

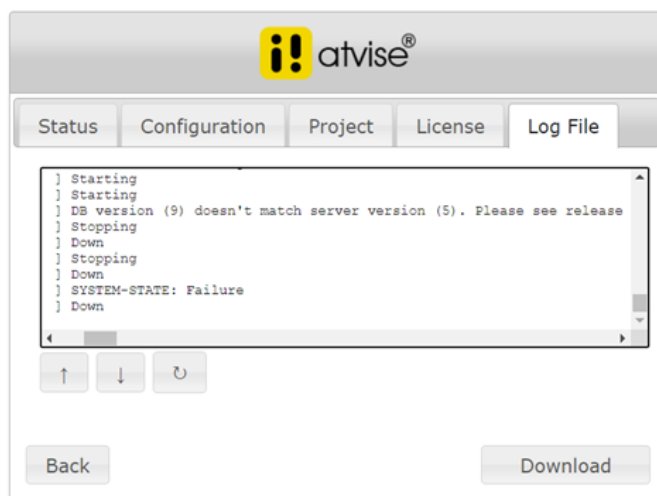


Figure 7: Server version error in OT1210W TSSW visualization. (Source: Author's own)

To resolve compatibility problems, it is necessary to synchronize the database version with the Atvise version. Updating one or both versions to mutually compatible ones solves the problem. However, updating the Atvise version requires external access via the use of TSSW or other methods explained in the Linux OT1200 manual (Bachmann Visutec SL, 2021), on the other hand, updating the Linux software version is as easy as selecting certain options in the configuration or TSSW. The operation of TSSW is explained in section 5.2 (Using the TSSW browser) of this thesis.

One of the main issues of this synchronisation process is that in the official Bachmann webpage for product downloads there is not any version of this database Linux files. Once a screen has been updated, it cannot be reverted easily to a prior software version (e.g., to version 1.29). According to the Linux 1.52 manual (Bachmann Visutec GmbH, 2021), to revert to earlier software versions, it is necessary to create an operating system image on an external memory device. However, generating such an image requires an advanced level of understanding and programming skills in C. As this is going deep in the industrial private sector, only the manufacturers of the product have this information and the necessary files to do a backup in an easy way.

The problem with the versions does not end here, in the download section of the atvise official page, the "Atvise software for OT1200" is only available for versions after 3.6, including it. This means that once the Atvise screen is updated, it is not possible to go back to version 3.4 unless the industrial supplier gives the ".update" file that is required.

As will be explained in section 5.3, the final decision taken will be the most industrially logical and future-proof for the displays. Both atvise and Linux will be upgraded to the latest version (Linux version 1.52.1 and Atvise version 3.9.1), however, this raises a third problem, the compatibility with the factory license that the display has included, without which it is not possible to start the atvise server they include.

5.1.2 OT1200 license configuration

Not only internal compatibility of the displays between the database and the atvise version has to be considered, another important point is whether the licence files are compatible with the version of the software being used.

Using PC licenses in OT1200

Regarding the licenses that can be acquired in the normal way per computer, Atvise works with a certain license structure. When a license is created for the PC software, that license is valid for any of the currently existing atvise versions, but only for the computer versions and with a maximum duration of one month, after that month the license expires and atvise stops working. In addition, this license only allows the user to view a single server at the

same time, which makes it difficult, for example, to communicate with the OT12010 screen and the PLC in case of having a server in each of the devices.

After doing compatibility tests of using the computer licenses on the OT1200 screen, it has been possible to make an OT1200 monitor work with the most up-to-date versions of Linux and Atvise, even being able to visualize a SCADA on the screen (without connection to the PLC, only to the local server). However, this operation will only be for one month and cooperation with other servers and devices such as the OPCUA server installed in the PLC will be limited. That is the main reason why contact with the Bachmann consumer service has been tried to establish.

The procedure for acquiring the monthly licences linked to the atvise accounts on PCs is explained in section 6.1 of this thesis (Atvise licensing). Although they last for a month, they are necessary to operate atvise on our PC. To activate it on the screen, the procedure is the same as explained in section 4.3.2 but select the ".lic" file relevant to the PC licence instead of the OT1200 licence.

OT1200 internal licenses issues

In the case of updating the screens, it can be argued that the licenses included with the display are valid for any version of atvise, in such a case, when upgrading the display and trying to start the server the following error seen in *Figure 8* is returned in the "Log File" window. This case corresponds to a screen updated to the latest version of Linux (1.52) and the latest version of atvise (3.9.1).

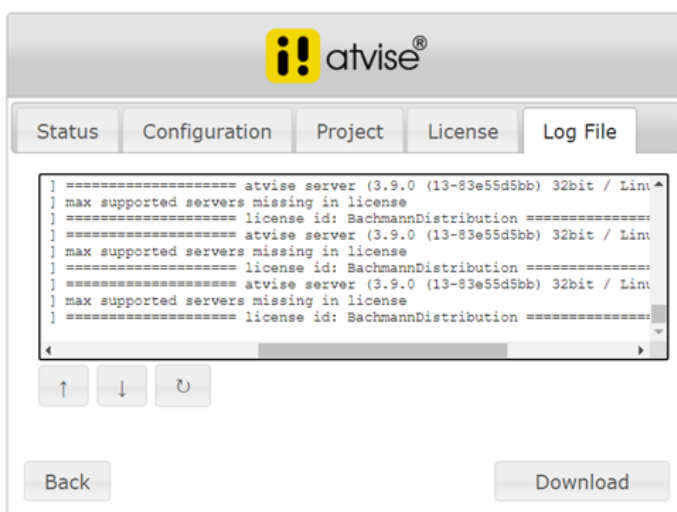


Figure 8: Error with license file structure due to compatibility. (Source: Author's own)

After contacting the technical service, this is the answer received in the words of the customer service technicians: “The panels available to the NOVIA university are models that were created when the current version was a version prior to 3.7 (in the development of this thesis it is known to be 3.4). From version 3.7 onwards, Atvise decided to change the license file structure from the products they deliver.”

In other words, this means that no matter how much the screens are updated, the consumer is forced to contact the distribution company to get new licenses for the products that supposedly already had one. This is usually a delayed process, as it can take up to 15 days for the technical service to respond, and it requires interaction and exchange of information to verify that the user is the true owner of the product and that he is not giving a licence to someone who should not have it. In addition, in order to obtain free PC licences for the atvise PC version, it is necessary to create a new user every month, so it has been necessary to change the e-mail address, making it even more difficult to get in touch with them.

Since one of the several objectives of this thesis is to prepare the industrial equipment to be used for a long time by the future students of NOVIA University. It has been decided to contact the technical service and update all the licensing structures of the displays. This will allow future users to operate with the latest version of atvise (3.9.1). The procedure for installing the new licences is explained in section 5.3.

5.1.3 Compatibility issues between OT1200 and PC

Another problem related to the issue of versions is that the Atvise Builder software on the user's personal computer must also be the same version with which they are working on the industrial screen, otherwise when trying to connect to the screen server from the computer a pop-up window as the seen in *Figure 9* appears and atvise doesn't allow to establish any kind of communication between the two devices.

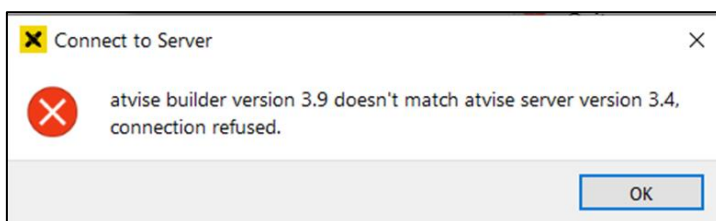


Figure 9: Compatibility Error pop-up window on PC. (Source: Author's own)

Due to the nature of the program's operation and compatibility, two different versions of Atvise can't coexist on the same operating system. For this reason, whenever it is desired to change the version, the previous version that was in use will be automatically uninstalled. However, it is important to mention that the created projects will be saved in the directories of each version. In this way, it has been possible to work with screens with different versions, changing the software version depending on the version of the screen with which we work without any problem.

5.2 Using the TSSW browser

This method consists of visualizing the configuration of the screen in the internet browser of the user's PC. This allows the user several options that are only possible in remote mode, such as installing licenses externally or updating the display. To do so, the following steps must be followed:

1. First, connect the ethernet cable between the PC and the screen. Then, connect the screen to the current and access the configuration during the 5 second count that is displayed on the screen just after starting the device. The user and password needed will be always "root" and "bachmann" respectively. Once in the configuration, follow the next path until arrive on the screen like the one in *Figure 10*: Settings > Network > ethernet1/ethernet0.
2. Configure the ethernet 0 (eth0) or ethernet 1 (eth1) window as the following picture (depending on which port of the screen is being used):

| Network Settings | | | |
|------------------|-------------------|--------------|-------|
| eth0 | eth1 | br0 (bridge) | Proxy |
| Mode: | Manual | | |
| MAC: | 00:10:7E:09:6D:AA | | |
| IP: | 192.168.1.50 | | |
| Netmask: | 255.255.255.0 | | |
| Gateway: | | | |
| DNS Server: | | | |
| Back | | Cancel | OK |

Figure 10: Network settings for TSSW (Source: Author's own)

From this point, the user can exit the screen configuration and keep it on in the main menu.

3. Configuration on the PC is required below. Search in the PC configuration “network configuration” and then access the ethernet port of the screen like in *Figure 11 a)* (it is possible to check which one is it by plugging it in and out multiple times). Right-click and go to properties.

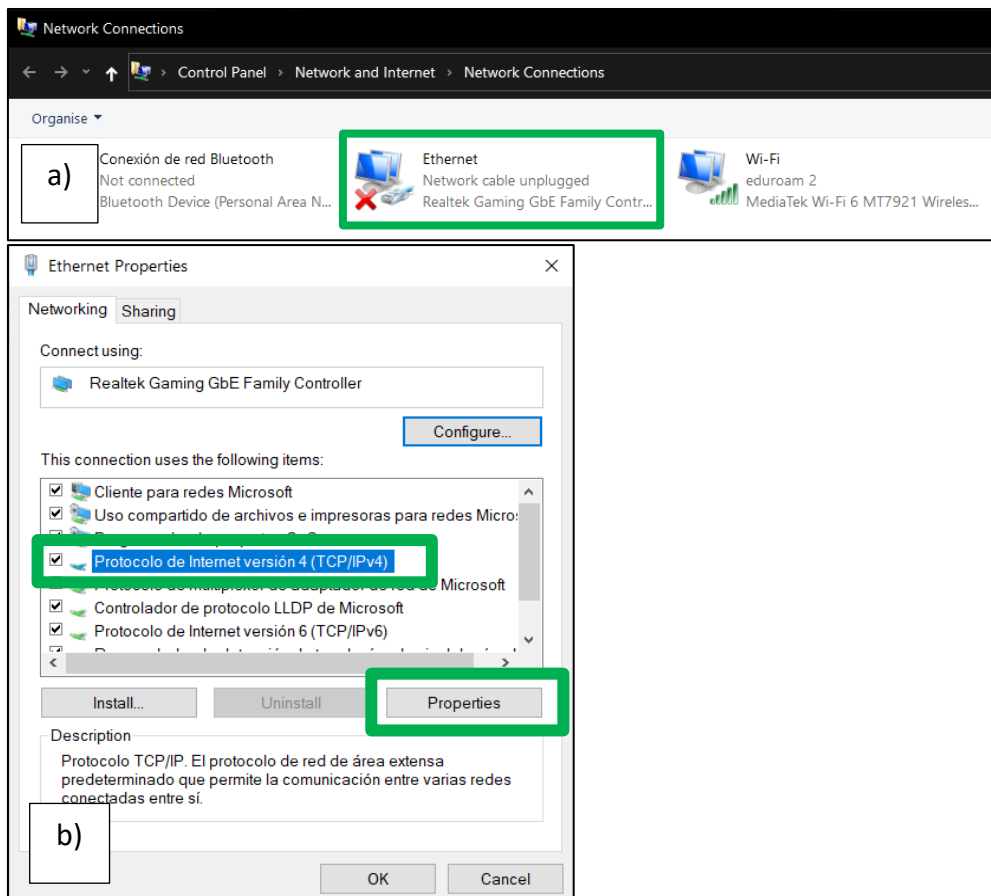


Figure 11: a) PC settings location. b) IP settings location (Source: Author's own)

4. Click on "IPv4 internet protocol" and access properties, and a screen like the following *Figure 12* will be displayed. Enter the same configuration that can be seen in the following picture. Then, save the configuration and close all the screens.

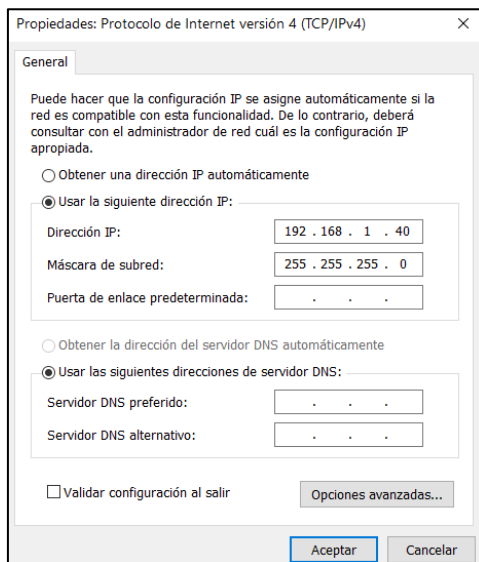


Figure 12: IP settings for TSSW (Source: Author's own)

- The final step is to go to the user's PC's browser and enter the IP that has been set in Bachmann's display browser configuration (in our case 192.168.1.50). The user may notice that this is an unsafe source alert, just ignore it and go to the destination.

If all the steps have been followed correctly, as in *Figure 13*, the following will be seen in the search engine:

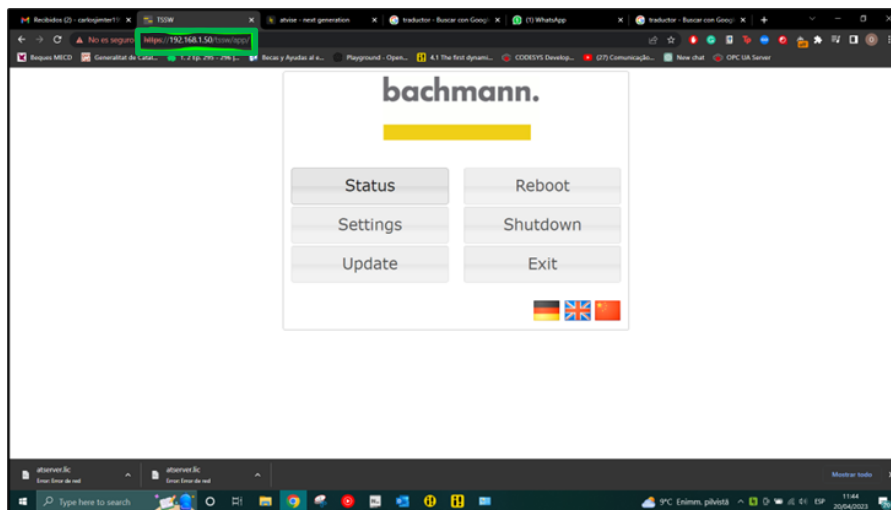


Figure 13: TSSW main menu visualization (Source: Author's own)

Something that this type of configuration contributes is the fact of being able to access the files that the PC has. When wanting to export or import projects in Atvise, this possibility is very useful. Also, for the matter of exporting and importing license files and updating the display.

5.3 Final OT1210W installation

5.3.1 Updating OT1200 Atvise

This section explains the process of upgrading the atvise and software versions of the OT1210W displays. It is important to mention that unless the licenses and versions expire many years after this thesis has been carried out, this procedure will not have to be performed again, as it has been carried out on all monitors available in Technobothnia Laboratory in the year 2023.

Initially, the Bachmann industrial displays come with atvise version 3.4.4. To upgrade to the latest version, it is necessary to create a user account on the atvise portal, as explained in the previous sections. Once the user account is obtained, it is possible to access the download directory available at the following link:

<https://customer.atvise.com/en/customer-area-downloads-en>

There, following the path “atvise” > “atvise X.X” > “atvise® X.X.X OT1200 Update File”, the latest version of atvise for OT1200 has to be downloaded. In the case of this thesis, the latest version is 3.9.1.

Then, after saving the “.update” file on the PC, access the TSSW in the browser as explained in section 4.2 of the thesis. Once there, access the "Update" option and from there select the file downloaded from the atvise page. After that, the update and installation of the new version will be done automatically, the display will restart automatically and the update process will be done.

5.3.2 License obtaining and installation

This part explains the final configuration and installation of the server licenses that have been made on the various OT1210W displays available in the Technobothnia laboratory. In case future users want to use this equipment, it is important to say that all these procedures have already been done once and unless new displays are acquired it will not be necessary to repeat this procedure. Other cases in which it may be necessary to repeat this procedure are in the case of being forced to reset the display to factory settings or if

the displays get outdated. This procedure has been done in all Technobothnia displays in 2023.

First of all, it is required to have activated the TSSW and to have opened in the web browser the configuration of the display, as explained in the previous section 5.2. Once the connection is established and the technical service is contacted, some information about the OT1210W model is requested.

The information required to update the 6 NOVIA screens was the serial number and the hardware code that distinguishes each product uniquely, to obtain this hardware code and to be able to send it by electronic messaging it was necessary to use the TSSW.

Once inside the configuration, using the TSSW go to "status" > "system", then, a screen like the one shown in *Figure 14 a)* will be displayed. From there it will be necessary to register the exact model type "Device type" and the unique serial number for each display "Serial number". Once this information is registered, go to the main screen and this time follow the path "settings" > "configuration" > "atvise" > "license", if followed properly it will be seen a screen like in *Figure 14 b)*. There the user will see the hardware code required by the manufacturer for the licence, copy it with the option "Copy to clipboard" and register it in the document that has to be sent to the customer service.

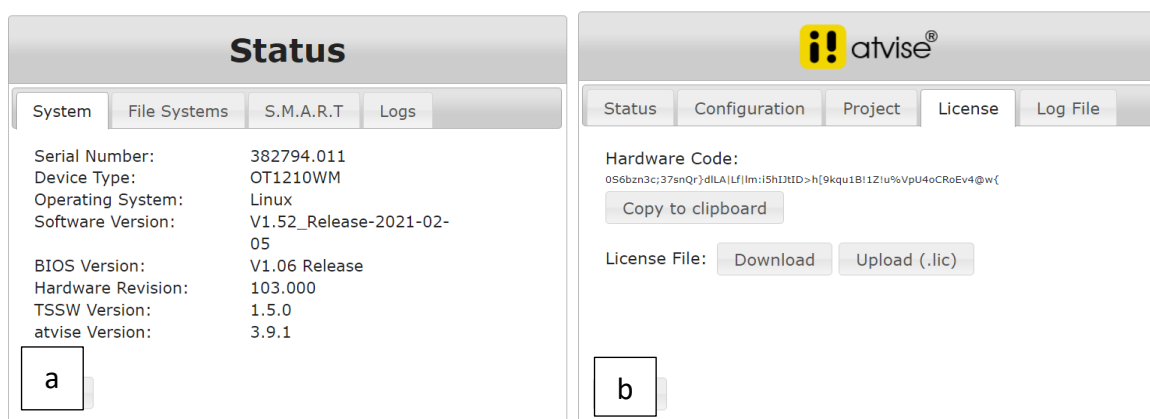


Figure 14: a) System information required. b) Hardware code required. (Source: Author's own)

This acquired information is sent to the technical service. First, communication was established, and the needs of the product were discussed. Theoretically, obtaining this licence can also be managed online via the user service on the atvise portal (<https://customer.atvise.com/en/customer-area-atvise-lizenzen-en>) by following the

steps explained in the YouTube video “atvise® Product Licensing (English)” (atvise, 2021). However, during the development of this thesis, it has been tried to use this method without any response from the technical service.

After receiving a reply from customer service, the licences will be available on the atvise licensing portal (link in the previous paragraph). These licences are in the form of a “.lic” file which must be downloaded and saved on the PC, where it can be accessed.

Once this file has been downloaded go to the “License” screen within the atvise settings on the TSSW screen and select the “upload .lic” option as seen in *Figure 15*.

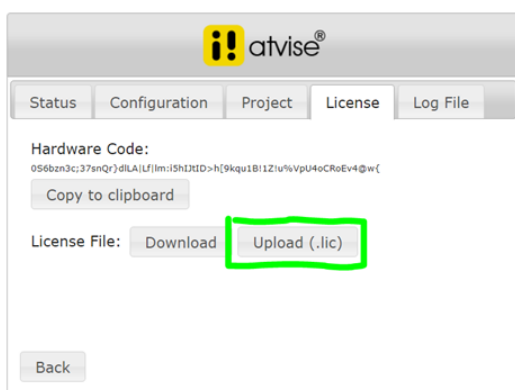


Figure 15: Upload “.lic” license option in TSSW. (Source: Author’s own)

This will open the file browser of the PC on which the TSSW is running. Select the “.lic” file previously downloaded from the atvise website and the licence update process is complete. To be able to carry out this process, it has been necessary to create different users on the Atvise website, so that, it has been possible to access the licences and downloads section that the company offers. This is because the users that can be created free of charge have limited licences of one month's duration. In this case, despite the use of limited free licences, customer service has promised to help update industrial monitors out of a commitment to the consumer.

6 PLC configuration as an OPC UA server

As has been explained in the first parts of the thesis, the final objective is to establish a communication network and operation of an OPC UA system, which can be commanded using a SCADA implemented through an HMI. As we have seen in section 2.1.2, where the operation of OPC UA is explained, any system that follows this standard requires a main server to send and receive information and a client that requests information and sends data.

Due to the possibility of the different components used to work independently, during the development of the thesis, it has been seen how the OPC UA server can be located in any of the different devices. It could be in the PLC, in the industrial display, or in the PC which finally has been used as a client. In order to mark some limits for the work and realisation of this procedure, it has been decided to continue only with the possibility of using the server that can be created in the PLC. In our case, it is intended to configure the Phoenix PLC as a server, and the PC and the HMI with the SCADA will act as a client. To configure the PLC in this way, the CodeSys software will be used.

Most of this section has been made according to the indication explained in the CodeSys official online manual (CodeSys Help, 2023).

6.1 PLC connection and basic programming

Below are the steps to follow in the CodeSys software to establish a server in the PLC.

1. The first of the steps is to install the CodeSys software. This can be done through CodeSys main web portal (<https://store.codesys.com/de/>). The download can be done by accessing the previous link and downloading the "CODESYS Development System V3" option, 32 or 64 bit version depending on the version of the operating system. To access the free software download, it will be necessary to create a user for the CODESYS store.

For establishing the server in the PLC it will be also necessary the software UAexpert, which works coordinated with CodeSys for data communication, and license server verifications.

The 1.6.3 version of UA Expert can be downloaded at the following link:
<https://www.unified-automation.com/downloads/opc-ua-clients/uaexpert.html>

2. Once the necessary programs have been downloaded, start the "CodeSys Installer" application that has been previously installed together with the CodeSys pack. Access "add-ons". Once in the menu, add to the desired version of CodeSys the add-on "Control for PLCnext".
3. Connect the PLC to the computer using the ethernet cable and access the connection and network settings of the user's PC, as seen in *Figure 16*. Right-click on the ethernet connection and then access properties (*Figure 16*). Click on internet protocol version IPv4, and then go to properties.

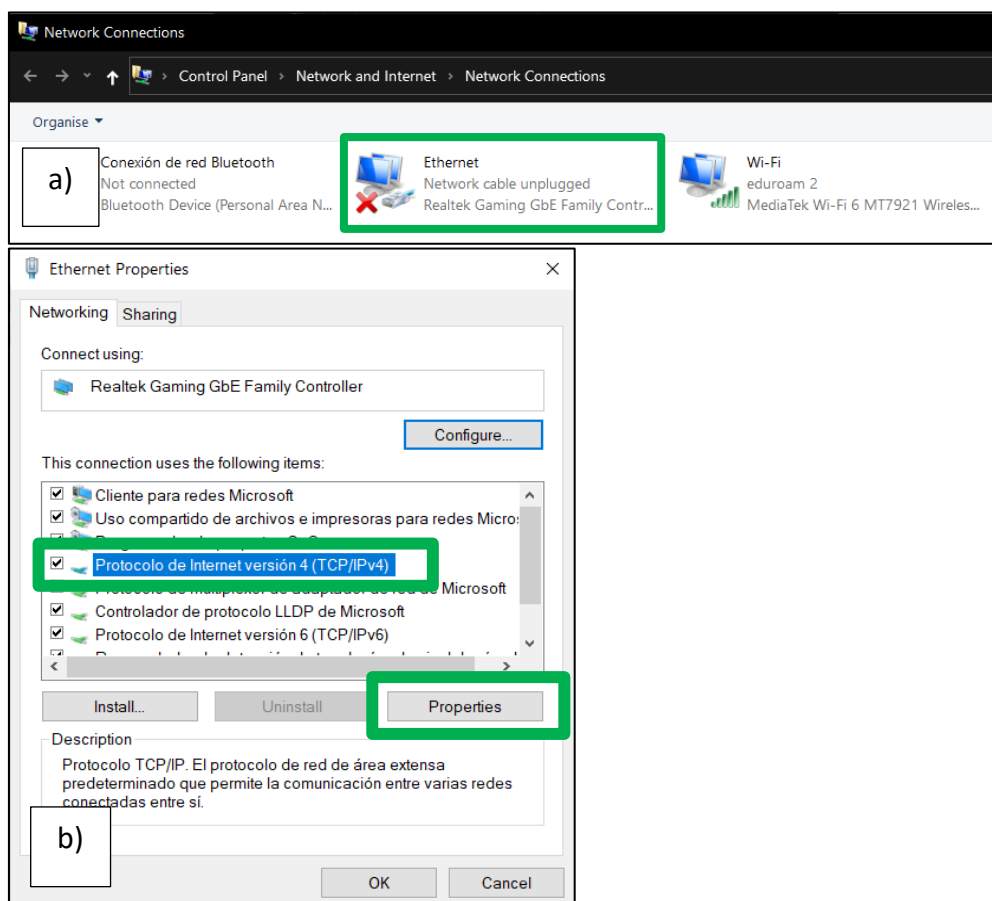


Figure 16: a) Network connections screen b) Ethernet properties settings (Source: Author's own)

4. Configure the IPv4 settings as the following visualization (*Figure 17*):

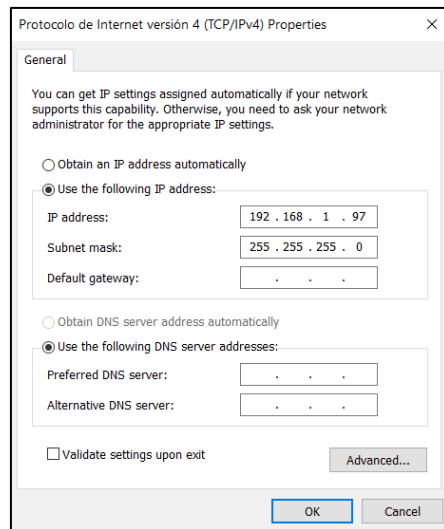


Figure 17: IP configuration for PLC connection (Source: Author's own)

5. Start CodeSys and start a new project. Configure as a standard project. In the device options select "CodeSys control for PLCnext SL" and in the kind of programming select Ladder Logic diagram (LD).
6. Once the project has been started, by following the order shown in *Figure 18*, go to "Tools" → "Update PLC next". In the screen that now appears in the left area of CodeSys, enter the password that is on the label of our PLC. Scan to find the plc on the network and select it (usually the IP is 102.168.1.10). In "codesys runtime package" select the plc driver version (4.7.0.0) and click the install option.

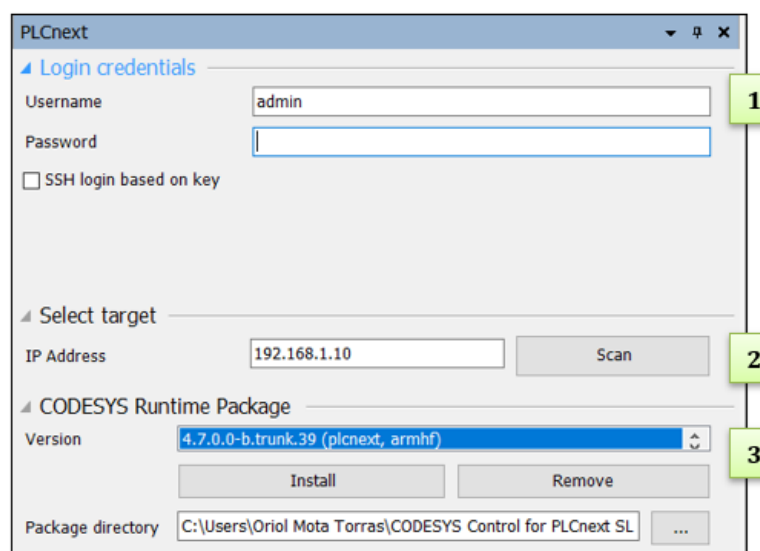


Figure 18: Steps for updating PLC Runtime Package. (Source: Author's own)

7. Next, go to "devices" and click on "Device (CODESYS control for PLCnext SL)", then, open communication settings on the right menu. Go to "gateway" → "add new gateway". Then, configure a new gateway with the desired name and the IP address that has been introduced in the PC configuration.
8. Select the new gateway and scan the network. Then select the PLC and "connect". The program will ask the user for name and password. Usually, the user and password are "admin" and "admin". If this does not work, try the password of the PLC label and admin as a user. If the connection is successful, the display will be like the following *Figure 19*:

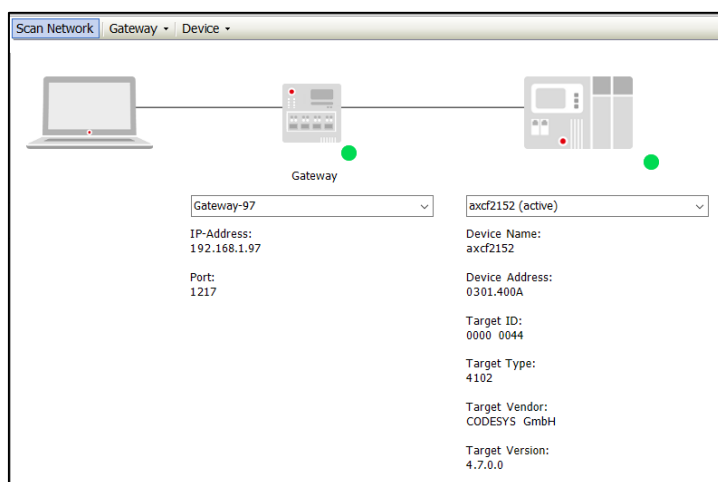


Figure 19: Correct connection and communication example (Source: Author's own)

9. The next step is to add the peripherals that are connected to the PLC. These are the digital, analogue and pulse generation inputs. To do it, in the menu in branches on the left (Device window), right-click on "PLCnext_Axioline" and click to add devices. Then add the devices in the following order:
 - I/O digital folder → AXL F DI16/1 DO16/1 2H (2n on the list)
 - I/O analog → AXL F AI2 AO2 1H (1rst on the list)
 - I/O functionmodule → AXL F PWM2 1H (5th on the list)

Is so important to add the devices in this order. Otherwise, The PLC won't function.

Now, the configuration of the PLC is completed, next, how to create a program is explained:

The next step is to create an example of a small program to put the PLC in active mode. To do so, in the branch menu on the left access "PLC_PRG (PRG)". There, with the help of the menu on the right, add some coils and associate their values to the digital outputs of the PLC. The address of each of the bits associated with the PLC can be seen in the information of the digital input device added in the previous section. In our case, the outputs will always be %QX0.0 - %QX0.7 and the second bit %QX1.0 - %QX1.7.

In the following *Figure 20*, a very basic example of the activation of inputs and outputs with LD programming is shown.

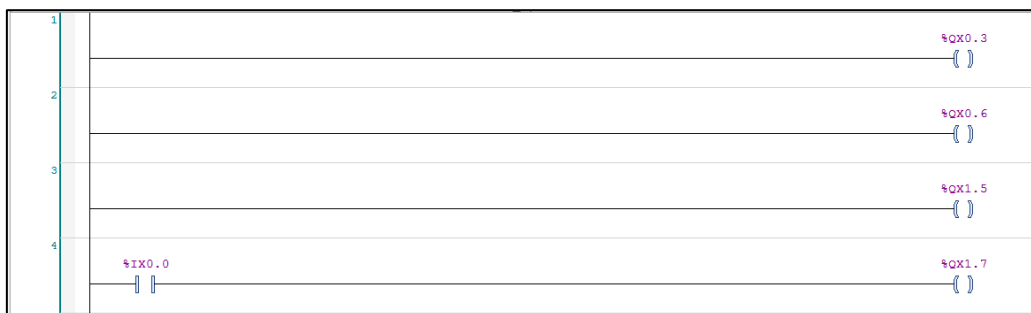


Figure 20: Ladder Diagram example in CodeSys (Source: Author's own)

The previous image shows an example of a contact and coil diagram in which, if the program is loaded into the PLC, there will be three permanently active outputs and one output that will be activated by activating the input located in bit 0.0. The activation of these bits can be checked in the PLC with the orange LEDs located in the different blocks. In the case corresponding to the previous program, as seen in *Figure 21*, the following bits will be activated.



Figure 21: PLC output panel activation (Source: Author's own)

6.2 OPC UA Server PLC installation and client connection

This section clearly and graphically exposes the necessary steps to follow in the CodeSys application in order to establish an OPC UA server in our PLC, this is explained in detail in the official guide for setting up a server (CodeSys Group, 2022). It is necessary to mention that before this guide, it is necessary to have completed the steps explained in the previous section (6.1 PLC connection and basic programming).

1. (If the user has already created a programme, it is not necessary to follow this step.) First, browsing the branch menu on the left of the software, go to the PLC_PRG programming section. Then select the variable list icon option (indicated with a one in *Figure 22*). Then, using the right click and add variable add several digital inputs and outputs, as seen in *Figure 22* below. The names of the variables can be anything, but the address must be within the limitations explained in section 5.1. Finally, create a LD diagram like the one shown below, for more detail on how to do it, see section 5.1.

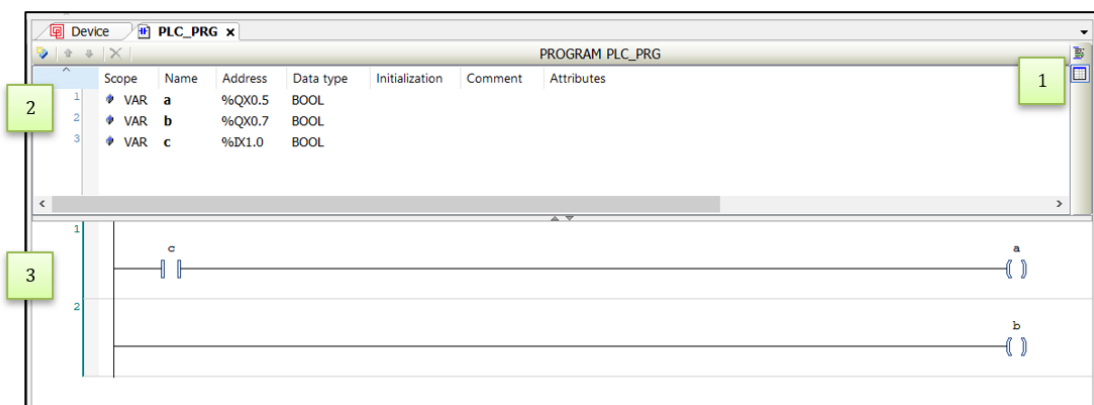


Figure 22: Steps for initializing variables and an LD program. (Source: Author's own)

To be able to see the variables in the server, is necessary that these variables have some action in the main program.

2. Next, by right clicking on "Application" from the menu on the left (with a nut symbol) add the "Symbol configuration" section, as seen in *Figure 23*.

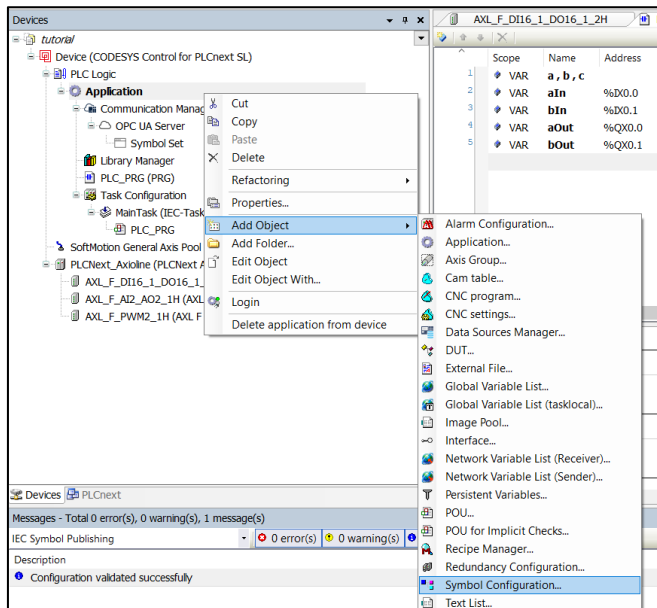


Figure 23: Adding Symbol Configuration option in CodeSys. (Source: Author's own)

When adding it, check the option "Support OPC UA features".

3. Next, open "Symbol configuration" and following the steps in Figure 24, click the build option, which comes out with an icon in the upper central part of the new window. Then, mark with a check the variables of the program that are desired to be uploaded to the server.

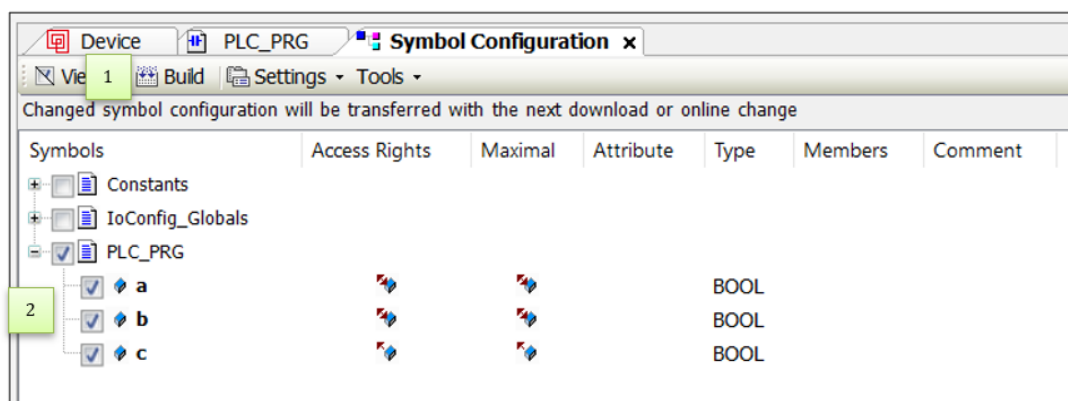



Figure 24: Steps to be followed in Symbol Configuration Screen. (Source: Author's own)

4. The next step is to create the necessary certificates for the server. Now, by clicking on the yellow shield at the right down of the user's screen (🛡️), go to the security screen. Once there, click on "Device" and then on the refresh symbol (🔄). After a while of loading a menu will open, and there, access to "own certificates". A display

will open with some certificates, click on "OPC UA Server", and click on the icon (). A certificate creation menu will open, change the duration to 2000 days minimum and accept. If it has been done correctly, it will be seen a validated OPC UA server certificate in green, like the following *Figure 25*.






| Information | Issued for | Issued by | Valid from | Valid until | Thumbprint |
|---|----------------------|----------------------|---------------------|--------------------------------|-------------|
|  | OPCUAServer@axcf2152 | OPCUAServer@axcf2152 | 02/02/2020 23:40:39 | 26/07/2025 00:40:39 (> 1 year) | 1DAB23AA90F |
|  | axcf2152 | axcf2152 | 28/08/2019 16:57:36 | 27/09/2019 16:57:36 (0 days) | 2B2D3FBAED |
|  Encrypted Communication | axcf2152 | axcf2152 | 02/02/2020 23:46:50 | 26/07/2025 00:46:50 (> 1 year) | 523AFCB4254 |
|  OPC UA Server | OPCUAServer@axcf2152 | OPCUAServer@axcf2152 | 11/02/2020 15:50:36 | 03/08/2025 16:50:36 (> 1 year) | 7037E78C3C3 |
|  | axcf2152 | axcf2152 | 01/01/2020 20:49:27 | 31/01/2020 20:49:27 (0 days) | D8E754E2292 |

Figure 25: Correctly installed OPCUA server certificate. (Source: Author's own)

- Going back to the menu on the left of CodeSys, double click on "Device (CODESYS Control for PLCnext SL)" to return to the menu where it can be seen the IP and gateway of our system. Once there, click on the drop-down menu in the centre of the screen and follow the path "Device"--> "Change Runtime security policy". A new window will open, in this, is necessary to enable the anonymous login at the bottom, in "Device User Management".
- For the next step to follow it is necessary to install the UA expert software, this software is where an OPC UA client will be created. The download link to install the software is as follows (<https://www.unified-automation.com/downloads/opc-ua-clients/uaexpert.html>). It will be necessary to create an account to be able to download the program, in this case, the account is validated instantly by confirming it by email. Once it is installed, start the program.
- After starting UAexpert, click on the "+" symbol in the top bar. A new server selection menu will open. Expand the "Custom Discovery" option and click on "<Double click to Add Server>". In the menu that will be displayed, the following address must be entered: `opc.tcp//:192.168.1.10:4840` , made up of the OPC header (opc.tcp), the PLC IP (192.168.1.10) and the OPCUA port (4840). If the previous steps have been followed correctly, when displaying the list of the new device that will appear, it should be seen see a list of server encryption methods similar to the one in *Figure 26*.

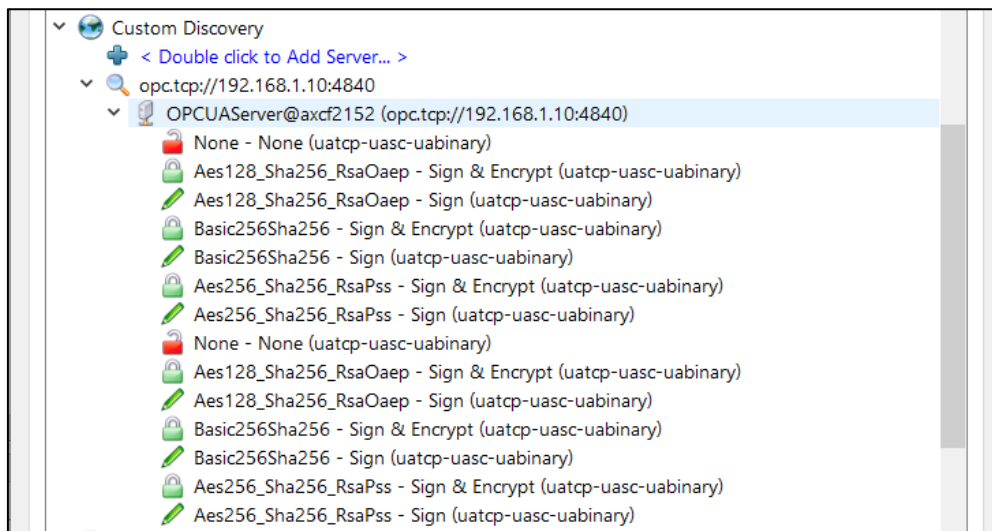


Figure 26: OPC UA server login options in UAExpert (Source: Author's own)

8. Next, select the first or second red padlock seen in *Figure 26* (usually the second one gives fewer verification errors) and with the "anonymous" option checked at the bottom of the same window, select "OK". All that remains is to go to the "Project" display on the upper left of the main screen, do right click on the new device and select "connect". If there are no errors, the server should start without problems and be 100% functional. To check that the server has started correctly in the lower left area of the main screen we should see something similar to *Figure 27*, in this new menu it is possible to navigate through the directory of the server and view the different configurations and variables.

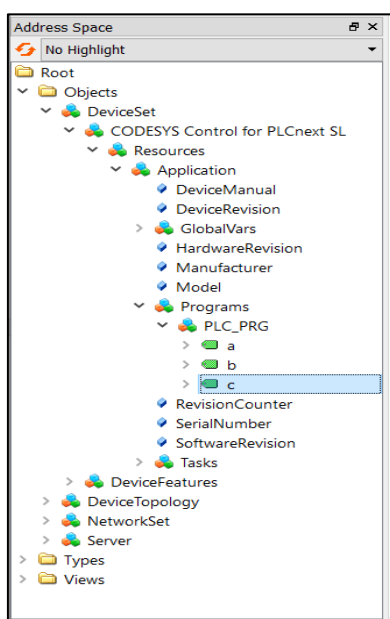
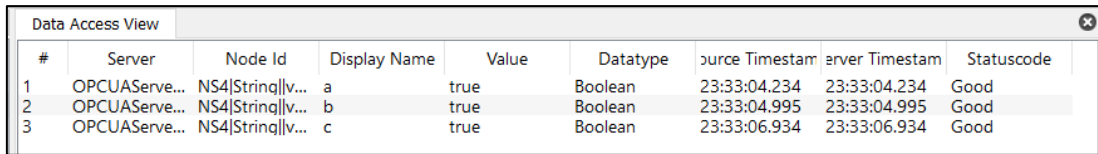


Figure 27: Final server directory menu (Source: Author's own)

If the variables of our program (whose location is shown in *Figure 27* above) are dragged to the central screen that is currently blank. It is possible to check the status of our server's variables in real time, as seen in the following *Figure 28*.



| # | Server | Node Id | Display Name | Value | Datatype | Source Timestamp | Server Timestamp | Statuscode |
|---|---------------|------------------|--------------|-------|----------|------------------|------------------|------------|
| 1 | OPCUAServe... | NS4 String lv... | a | true | Boolean | 23:33:04.234 | 23:33:04.234 | Good |
| 2 | OPCUAServe... | NS4 String lv... | b | true | Boolean | 23:33:04.995 | 23:33:04.995 | Good |
| 3 | OPCUAServe... | NS4 String lv... | c | true | Boolean | 23:33:06.934 | 23:33:06.934 | Good |

Figure 28: Server variable monitorization (Source: Author's own)

In *Figure 28* shown before, the user can see the actual server address of each variable, the name, the datatype, the time of the server associated and finally, if the status is good or there is any error.

7 Atvise functioning

To develop a SCADA in the following section the previous concepts and mechanics of Atvise SCADA are exposed. These mechanics will be used for the development of the SCADA system that will control the door in miniature explained in section 3.5 of assembly equipment. Mostly, these explanations are focused on the customisations and functioning of the visual part that will be shown on the screen once the final system is implemented.

7.1 Atvise PC initial configuration

7.1.1 Installing Atvise

The first step to be done is to instal the desired version of Atvise Builder and Atvise Connect. Depending on the version in which the user is going to work, here are the different portals with the technical explanations of each version and their download links:

<https://customer.atvise.com/en/customer-area-downloads-en>

The previous link shows a directory of online folders with all the available versions of Atvise and Atvise Connect. During the development of this thesis, we initially worked with atvise


version 3.4.6, as this is the version compatible with the software pre-installed on the industrial hardware. In the final stages of the thesis, all screens were upgraded to the latest version, so the final versions used were Atvise version 3.9.1 for Windows and OT1200.

7.1.2 Licencing Atvise

Next, the process of how to get a license and installing it o atvise is explained, these steps has been written by using the YouTube video “atvise Product Licensing” (atvise, 2021) as a reference. One of the biggest problems when using Atvise is the license issue. After maintaining contact with the distribution company "Bachmann" and "Atvise" we have not been able to obtain by any method permanent licenses or with a certain duration for the personal computer. Not even justifying the ownership of the OT1210 screens with which we work in the development of this thesis. This is because the licenses attached to the screens purchased by NOVIA University are only used to operate and establish an Atvise server on the OT1210 screen.

There are several processes explained on the web pages of each of the companies that collaborate in the operation of Atvise (Vester’s, Bachmann, Atvise, etc.), however, after testing the online login option has been considered the most appropriate and fastest.

The procedure consists of creating an account on the Atvise portal, after completing the information it will be necessary to wait a period from 1 to 5 days for a person from the company to validate the account. Then it is possible to start Atvise and activate the said license linked to the account. The bad point of this option is that the license linked to the account only has a duration of one month, after this period it will be necessary to use a new email account. The link to create an account in the Atvise online service is the following: <https://customer.atvise.com/en/login>

Now entering Atvise, start "Atvise monitor" as administrator. An icon like the following  will be seen at the bottom right of Windows, in the taskbar. Click on this symbol and access the “license” option. There go to “Online Activation” and enter the user and password previously created in the online Atvise portal. After confirming a one-month license will be available.

Once the license is selected, it will be possible to create a project (the user will be asked to choose a name and directory) and start a local server (automatic when creating the project). If the license is valid and the version of the program does not have any compatibility problems, the previous icon will turn green (🟢).

7.2 Atvise builder functioning

To start any project, it is necessary to have followed the explanation made in the previous section 6.1. Once the license has been activated, it will be possible to carry out the steps explained below. It is necessary to explain that in this way we are only connecting to a local Atvise server that works internally on our computer. In this section, only the principles and fundamentals of Atvise will be exposed to develop SCADA and create a basic interface that works. In the 6.4 section, it is explained how to connect to the screen with our Atvise and view the displays and SCADA that have been created on our PC.

One important clarification to mention is that this general explanation of the program only focuses on explaining the points that have been considered necessary and useful for the creation of a SCADA like the one developed in this thesis. For this reason, there are certain sections such as object structures, alarming, permissions, user licenses, etc. Which will not be explained because are considered out of scope.

When starting the program, a pop-up window will open, where it is needed to enter the address of the server to which the user wants to connect. In the case of wanting to connect to the local-host server (the server that will be used to test the PC browser itself) enter the following link on "server": "opc.tcp://localhost:4840" as seen in *Figure 29* shown below. In case the program asks for a user and password, are always "root" and nothing respectively.

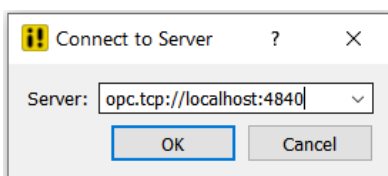


Figure 29: Connect to server window. (Source: Author's own)

7.2.1 Main screen

Once the program has been connected to the respective atvise server in case it is the first time the program is started, the user will be asked to configure the window size. After selecting the most suitable option for the project being carried out, the main menu, shown in *Figure 30*, will be displayed.



Figure 30: Atvise Builder main menu. (Source: Author's own)

Library window

When entering Atvise Builder it can be identified 3 main subdivisions with different options in each one. At the bottom, it can be seen the library browser and symbol browser linked to the library as seen in *Error! No se encuentra el origen de la referencia.*. On the left side of the library browser screen, there is a window that allows the user to navigate through the Atvise libraries. There is possible to find the different objects and images that are used during the creation of a SCADA. This window is linked to the one in the lower central part of the screen, in this part, the user can navigate and see graphically the different predefined blocks that are in atvise. For example, is possible to insert sliders, pushbuttons, bar graphs, predesigned text labels, etc.

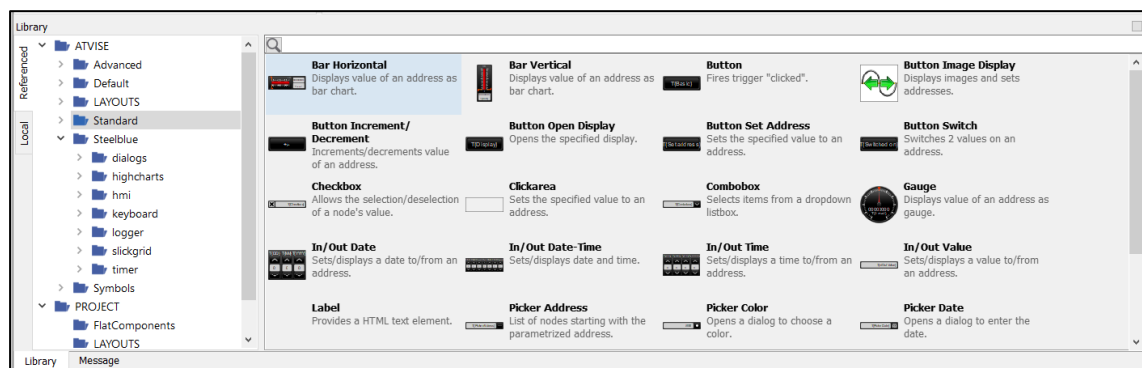


Figure 31: Library window in Atvise Builder. (Source: own)

All these elements can later be linked to nodes and variables available in the project. This will allow the users to have dynamic control of the information in our system. Moreover, this allows the users to be able to have more complex elements in the SCADA that in the case of self-designing the structures by themselves. It would be possible but requires more advanced programming, something that atvise also counts on but does not focus on.

Project Navigator

On the left side of the main screen, it can be seen the project navigator. As seen in *Figure 32* This screen consists of a branch drop-down menu that allows the user to navigate through the different contents of our project. In this menu, it is possible to find the different servers on which the user will work, the variables and objects of the system, and the screen that will form the SCADA display, among others. It also contains directories and libraries of the program itself as well as global configuration parameters and user permissions.

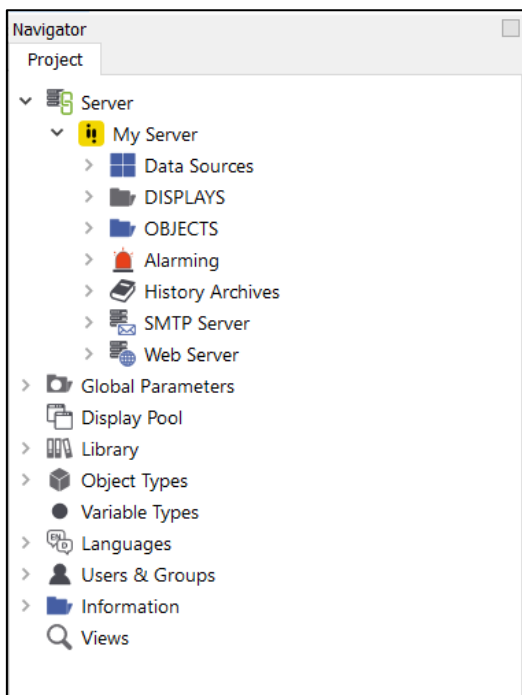


Figure 32: Project Navigator window. (Source: own)

The main use of this part of atvise will be within the "My server" service, which can be seen in *Figure 32* above. This server is where the different data sources will be added. It is also on this server where the different displays that will make up the SCADA will be created and displayed.

Toolbar

In the upper toolbar, there are several options of general interest that can normally be carried out from other points of the software. However, here are grouped in drop-down menus to be able to access them more easily. The most important functions that will be explained below are those found in the “builder” (*Figure 33 b*) and the “Guided Actions” (*Figure 33 c*) dropdown menu.

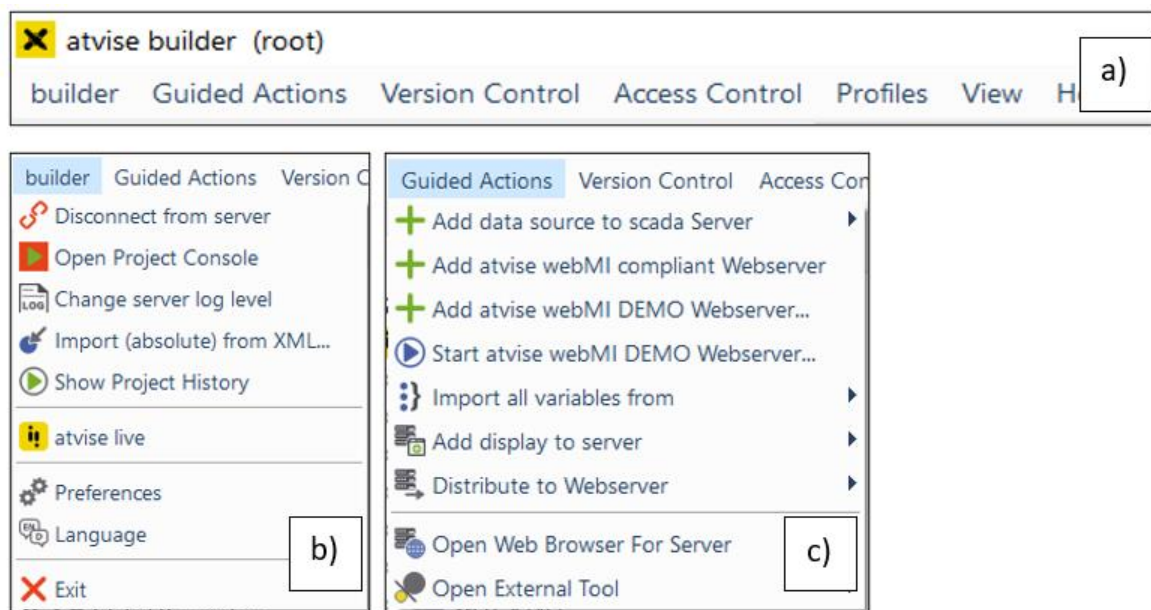


Figure 33 : a) Atvise Builder toolbar. b) builder dropdown menu. c) Guided Action dropdown menu. (Source: Author's own)

From the builder drop-down menu, it is important to highlight the "Project Console" and "Connect/Disconnect to server" options. The project console option allows the user to return to the project console, from where the user can stop and restart the local server on which he normally works. The user can also switch between projects and create new ones, as explained at the beginning of section 6.

Regarding the Guided actions menu, some of the most common actions in atvise are shown, mainly related to communication with elements external to the program itself. It is worth mentioning the option to add a data source, although it can be done in a specific way in the "project navigator". This option allows the customer to add an external source of information to atvise via its IP address. Its operation will be explained in more detail in section 6.2.2 below.

The other main and most used option in this tab is the "Open Web Browser For Server" option, by clicking on this box our default web browser opens with the atvise localhost address "127.0.0.1". Thanks to this option it is possible to simulate and control the SCADA that the user designs, by simulating it in the browser, the PC will behave as if it were an industrial HMI located in the factory.

Central window

In the central area of the main menu, initially, it can only be seen information related to the version of the program. Once the different configuration options are accessed by using the project navigator located on the left side, the different options for each section will be displayed. In the central area, the user can edit and view information from each of the nodes, user permissions, alarms, data sources directories, and any of the project navigator options. This central area will also be the workspace where the screen or screens that will form our SCADA will be developed.

7.2.2 Datasource and Node concept

Nodes:

A data source in atvise is the source of information internal or external to the server and the program that provides the data that will be read and modified, these data are presented in the atvise environment in the form of "nodes". These nodes can be any simple or complex information structure to be worked with in the program. To understand it better, a node can be anything from a structure made up of various objects (shapes, graphics, sliders, text) that are joined in a single unit to a basic Boolean type of variable that can only be worth one or zero.

At first glance it may sound like a complex concept, this is because despite being redundant, an object, which is considered a node, can contain several more basic nodes. During the development of this thesis, even though it could be possible to work with structures and types of objects, it has been decided to limit itself to the most basic operation of the program and to use basic type nodes. These are mostly variables that come from the PLC and are of Boolean type, so they will be memory bits that can only have a true or false value. And that, will be modified in real time by the SCADA operation that has been designed.

To see the variety of basic types that a node can be, *Figure 34* below shows all the different types that the user can create.

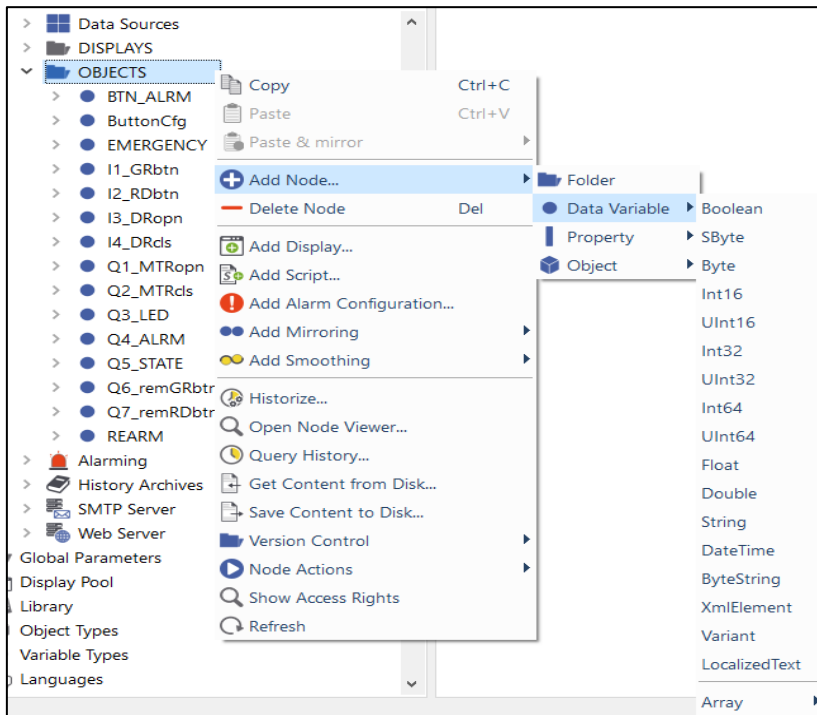


Figure 34: Node types in Atvise Builder. (Source: Author's own)

Data sources:

One of the most important elements for Atvise is synchronization with external elements. After all, for a SCADA to function and be operational, it is required to control certain variables of a PLC, server, or system. The external data sources can be from OPC UA, OPC COM, WebMI, SNMP, siemens S7 and atvise connect. In the case of the thesis, the data source used is atvise connect, this is an atvise program that manages a wider range of data sources as well as servers, clients, MQTT, OPC Classic, OPC UA, etc. The operation of this program is explained in section 6.3.

To add a data source to Atvise the user can select the option in "Guided actions" or by right clicking on "data sources" > "add data source". The user will be asked to fill in the necessary fields to register the new source of external information (IP, port, certificates, etc.). In case the source has been added successfully, when editing it by "right click" > "edit" it will be seen that the "status" has changed to "connected".

Once a data source has been added successfully to Atvise, it is possible to navigate through its libraries by using "browse". By navigating the libraries, it is possible to find the program

loaded on the PLC server and consequently, the variables. These variables can be synchronized in real time with the online operation of the SCADA through a special copy paste, the variables must be copied normally, but when adding them to the program (usually in the object folder) a "paste and mirror" must be done, this means that these variables that have been exported from the external server to the software are always linked to the SCADA. This whole process is explained in the official atvise YouTube video (atvise, 2023). By doing these steps the users will be able to write and read about these variables at any time the server is up and running.

In the data source section of atvise Builder, there are also options to use the projects themselves and other atvise servers as their own data source, these have not been used during the development of the thesis but allow to synchronize several projects to create a joint network, very useful on a large-scale industrial level.

7.2.3 Display creation

Undoubtedly the most important point of this program is the design of an industrial interface capable of controlling a system. The design and programming of this interface is carried out in the work area located in the central window of the software. To open a blank screen where the user can start working, just go to the current server and create a new display or access the main display already created by default (the main screen is the one that will be seen when starting the SCADA).

Next, this subsection will explain the main tools that have been used during the development of the thesis and that have been considered the most important to create a SCADA. In the following *Figure 35*, the sections of the toolbar that perform a key action are marked with different colours and letters, the order in which they are explained and named corresponds to the order in which the user should work to create the SCADA properly. The parts that have not been highlighted in the following figure have not been considered of such common or important use that they need to be explained in the development of this thesis. For more specific details on the different options, consult the Atvise Builder manual (Bachmann Visutec GmbH, 2023) or the "Help" option included in the program Atvise Builder.

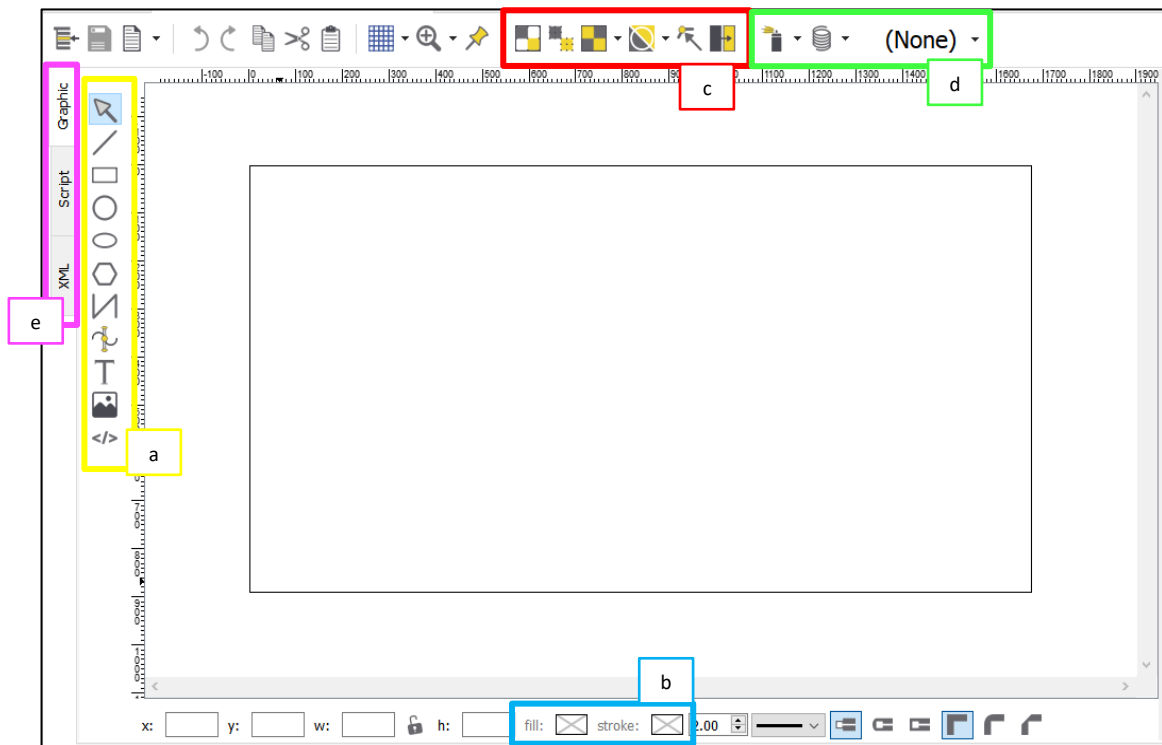





Figure 35: Display editor window in Atvise Builder. (Source: Author's own)

First of all, on the left side of the *Figure 35* editor, it can be seen the bar marked in yellow with an "a)", this bar allows the users to add figures and draw any type of shape. This can remind a lot of the classic software like Paint, where is it possible to add circles, rectangles, polylines, etc. Once these shapes have been added, they can be rotated, resized, deformed, overlapped, etc. All these options are necessary to create a good visualisation. This toolbar also allows us to create text and add images of our device. As a whole, this toolbar will allow us to make the first steps more focused on the design of our SCADA. It is important to mention that to see the names of each of the existing options in the editor, simply place the mouse cursor over it without clicking for a few seconds.

This first bar works together with the blue bar marked with a b) and the red bar marked with a c) located in *Figure 35*. These toolbars focus on the editing of the previously added shapes, in the b) area the user can colour the fill and outline of any shape, in case he wants more sophisticated things the user can do gradients, transparency, etc. The area with c) focuses more on the coordination between the various figures of the visualization. In addition to the classic option to group and ungroup figures, it has several drop-down menus that allow the user to bring the different shapes to the front or the bottom, thus choosing how the user wants to superimpose them.

In the same drop-down menu named “Arrange” (), there are also very useful options to align several elements symmetrically and parallel to them, making the visualization more pleasant when there are several elements. Lastly, there is also a last drop-down menu named “Transform” (), that one allows the user to rotate and mirror the different figures.

One of the most important tools that make sense of so much design is the submenu that is marked in green with the letter d) in *Figure 35*. This menu allows us to add simple dynamics to any of the previously added graphic elements. This is the part of the SCADA design where the visualization begins to make sense and is linked to the existing variables in the program. A simple dynamic can be, for example, the programming of a button, where when the selected variable is pressed, it changes from true to false. Next in *Figure 36*, a pop-up window is shown in which there are all the possible simple dynamics configuration options. This pop-up window can be accessed via the "Quick dynamics" () drop-down bar or by right-clicking on the element to which it is desired to add the dynamics, and then, selecting "Add Simple Dynamics" at the bottom.

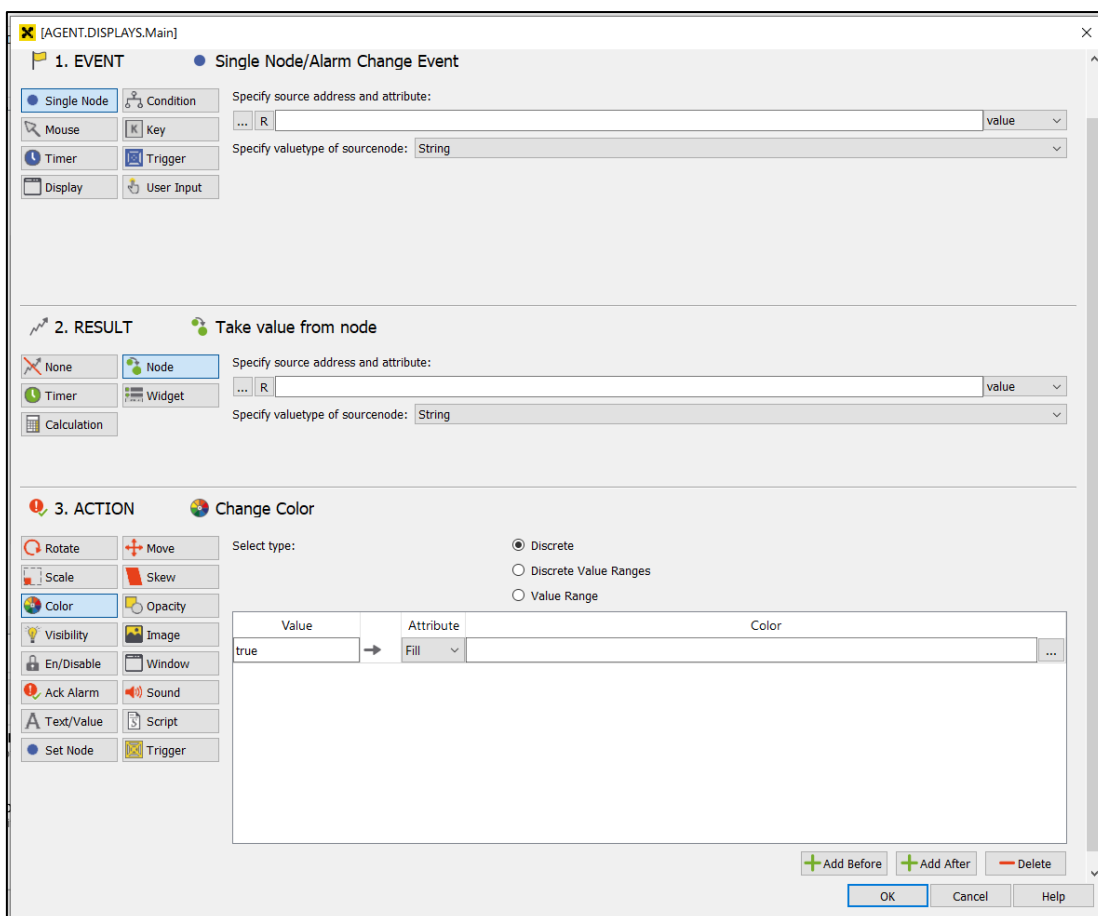


Figure 36: Simple dynamics window Atvise Builder. (Author: Author's own)

As can be seen in *Figure 36* above, the simple dynamics pop-up window is divided into three parts: event, result, and action. In the first part, event, the user selects the event or action that will mark the start of the dynamic. In other words, start the action if a button is pressed, if it is a passive action that responds to an external variable change, or even if more complex conditions are met, such as pressing certain keys on the keyboard or meeting an exact value of certain variables.

In the second part of the pop-up window in *Figure 36*, "result", the user chooses what the action will affect. The clearest case is to select a certain variable that due to the action is going to change values, in which case the "Node" option would be selected. There are also more sophisticated options such as performing calculations or initiating timers that have not been explored in depth during the development of the thesis.

In the last section of the window shown in *Figure 36*, "action", the different actions that will be performed by the elements present in the SCADA and that the user will be able to see are configured. Unlike the two previous sections, where the configuration is much more reduced, here the freedom of options is much greater.

Some examples of actions that can be configured in this part are changing the value of a variable, changing the colour of an element, sound emission by an element, executing a certain script created by the user, modifying the visibility of an element, deforming an element, etc. For example, the following *Figure 37 a)* shows a block of code corresponding to the action performed by the green button of our SCADA, this action has been programmed only by selecting the "Add simple dynamic" options shown in *Figure 37 b)*.

```

15 - webMI.data.read("AGENT.OBJECTS.Q6_remGRbtn", function(f) {
16     var value = f.value;
17     if (value == false)
18         webMI.data.write("AGENT.OBJECTS.Q6_remGRbtn", "true");
19     if (value == true)
20         webMI.data.write("AGENT.OBJECTS.Q6_remGRbtn", "false");
21     });
22 });

```

a)

1. EVENT
Mouse Event

- Single Node
- Mouse
- Timer
- Display

- Condition
- Key
- Trigger
- User Input

Specify mouse/touch event:

| | | |
|--|---------------------------------------|--------------------------------------|
| <input type="checkbox"/> Click | <input type="checkbox"/> Mouse-In | <input type="checkbox"/> Touch-Start |
| <input type="checkbox"/> Double Click | <input type="checkbox"/> Mouse-Move | <input type="checkbox"/> Touch-End |
| <input checked="" type="checkbox"/> Down | <input type="checkbox"/> Mouse-Out | <input type="checkbox"/> Touch-Move |
| <input checked="" type="checkbox"/> Up | <input type="checkbox"/> Touch-Cancel | |

2. RESULT
Take value from node

- None
- Timer
- Calculation

- Node
- Widget

Specify source address and attribute:

... R AGENT.OBJECTS.Q6_remGRbtn value

Specify valuetype of sourcenode: Boolean

3. ACTION
Set Address(es)

- Rotate
- Scale
- Color
- Visibility
- En/Disable
- Ack Alarm
- Text/Value
- Set Node

- Move
- Skew
- Opacity
- Image
- Window
- Sound
- Script
- Trigger

Select type:

- Take value from event
- Discrete
- Condition based value (Discrete)
- Condition based value (Range)

| Value | | Address | | Value |
|-------|---|-----------------------------|---|-------|
| false | → | ... R T.OBJECTS.Q6_remGRbtn | → | true |
| true | → | ... R T.OBJECTS.Q6_remGRbtn | → | false |

b)

Figure 37: a) Green button code. b) Green button dynamic. (Source: Author's own)

The exact options that have been used to configure the buttons as the green one shown in Figure 37 and the other elements of the final SCADA are explained in detail in section 6.5. Where it is specified exactly how to create each type of button (push button, switch, etc.), how to affect objects correctly, etc.

A very important factor to explain is the fact that atvise offers the users the possibility to visualise the consequences that all these dynamics have on the programming level. Accessing the area marked with an e) in Figure 35, a new window called "script" can be accessed. There it is possible to see, after implementing a certain dynamic, the translation into code language of the new action implemented in the SCADA element.

This may seem a minor thing, but more than just learning how the code works, it can be used to create much more sophisticated functions than the atvise editor allows. Classical functions such as those that exist when programming in c (while, for, if else, etc.) can be created and executed in this section of code. This allows the consumer to take the creation

of the industrial level SCADA to another level, both the dynamics and the design of the display itself can be affected in this section.

In general terms, these are the tools that will be used mainly for the creation of a SCADA, in section 8, a more detailed explanation of how each part has been configured to obtain the final result is made.

7.2.4 Issues with programming in Atvise

As explained in the final part of the previous section, Atvise has a certain programming capacity that could allow it to manage the entire plant operation in a precise and accurate way.

However, this high level of customisation can cause some problems when creating a system in the industry. When a user creates an industrial system that is controlled by a SCADA, it is required that the machine can operate independently of whether the system is online or offline. This sounds obvious, but when start programming a SCADA it is necessary to very concerned about this problem. Let's take the following example:

In the case of this thesis, a miniature model of an industrial gate provided by NOVIA University is controlled. This gate has two buttons that perform a basic control, the green button opens or closes the gate, and the red button stops the movement. Although the OPC UA server is located in the PLC and can be accessed remotely, the program that makes this system function must be loaded locally in the PLC. So, when an operator is physically in front of the machine (gate in this case), he can open or close the gate even if the system is not online for whatever reason.

Let's suppose the case of making a programming mistake and adding in Atvise a dynamic that when pressing the variable "green button" (green button changes value to 1) the variable "move gate motor" is activated. The operation will be correct and adequate both locally and remotely only when the server is online and is connected to atvise. In the event that there is an error in the plant and the connection fails, the door will be unusable, this is a great risk in the operation of the industry.

In conclusion, If the program that controls the industrial system is loaded in the remote client, in this case, PC Atvise, the functioning of the program can only be developed if the

server is online. On the other hand, if the conditions and all the interactions are located locally in the PLC, even if there is a connection failure, the system will still respond to the physical controls and the machine will continue its operation. In the worst case of performing all actions and operating conditions in Atvise, the machine would stop working completely in case of connection failure.

The following *Figure 38 a)* shows how the "activate button"- "activate motor" condition is programmed in the CodeSys contact diagram. *Figure 38 b)* shows how the same condition is programmed in Atvise Builder by a simple dynamic associated to a button. Both options are the same as an "IF" condition in that changes some value if the condition "green button" is true, however, in the industrial environment it should never be done in the Atvise client b).

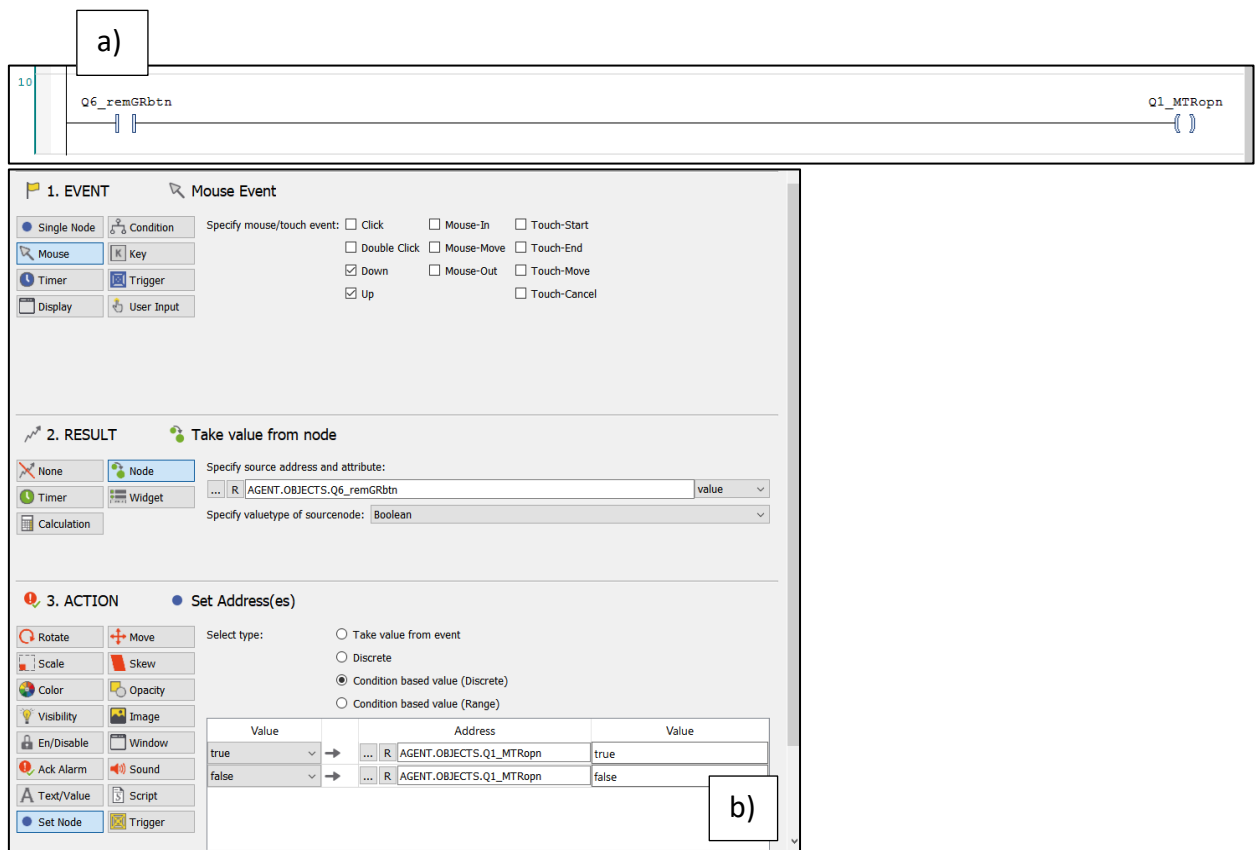


Figure 38: a) If condition in LD diagram in CodeSys. b) If condition as dynamic in Atvise. (Source: Author's own)

For this reason, as explained in more detail in section 8, in this thesis the PLC will have a certain program loaded internally as the one seen in Figure 38 a). So, the SCADA buttons and options will only modify the variable "green button" (change green button from 0 to 1)

and the PLC program will be in charge of the action of opening the door (set door motor to 1 if green button is 1).

Another problem encountered during the development of the project was the fact that a CodeSys program digital input variable can only be equipped with read rights in Atvise Connect. This means that to simulate the start and stop buttons, even though they are inputs of our system, they must be programmed as outputs of the system.

As a result, this causes that when monitoring the status of the door and the behaviour of the program, both the physical button and the virtual button created to operate from SCADA must be considered.

7.3 Atvise connect functioning

Atvise Connect is a software that works in coordination with the Atvise Builder software to be able to add data sources to the Atvise server the user is working on. As explained on the official website of this Bachmann product (Bachmann Visutec GmbH, 2023), Atvise Connect extends the communication interfaces of atvise® hmi and atvise® scada. The software provides server connectivity to a wide range of industrial controllers and ensures top-level performance in industrial data acquisition. Following, the main and most useful functions of Atvise Connect, as well as the more used for the development of this thesis are explained.

The Atvise Connect software, in the same way as Atvise Builder, has been created by Bachmann Visutec GmbH and can be found in the atvise download directory. The exact path that leads directly to the Atvise Connect download corresponds to the link below (<https://customer.atvise.com/en/>).

First of all, the software must always be started with the option to run as administrator, otherwise, it will run for a short period of time and then close automatically. Once this has been clarified, when a program is started, a window like the one seen in *Figure 39* will appear and the user will see two main options: "Configure product over network" and "Configure product on this PC".

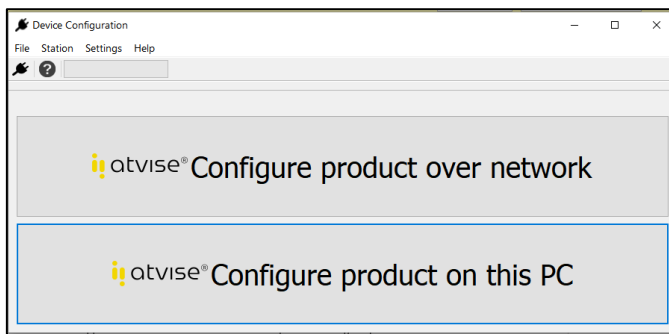


Figure 39: Atvise Connect main screen. (Author: Author's own)

The second option shown in *Figure 39* is the one that will generate the least errors and the one that has been used for the development of this thesis. It consists of connecting to the local Atvise server which is created through the Atvise Builder project console. Once this option is accessed, it can be seen the different OPCUA servers and clients that are automatically generated when the server is created. The peculiarity of this option is that it works much faster, and it is easier to connect to Atvise Builder, however, it is a local server on the developer's PC, which is not usable on a purely industrial level.

The first option shown in *Figure 39* opens a secondary menu, where, at first, only the servers that Atvise Connect can detect automatically with a first scan of the network are displayed. This secondary screen shown in *Figure 40* also offers the option to add new servers connected to the network by adding their IP address and name in a pop-up window shown after selecting the option "New Station".

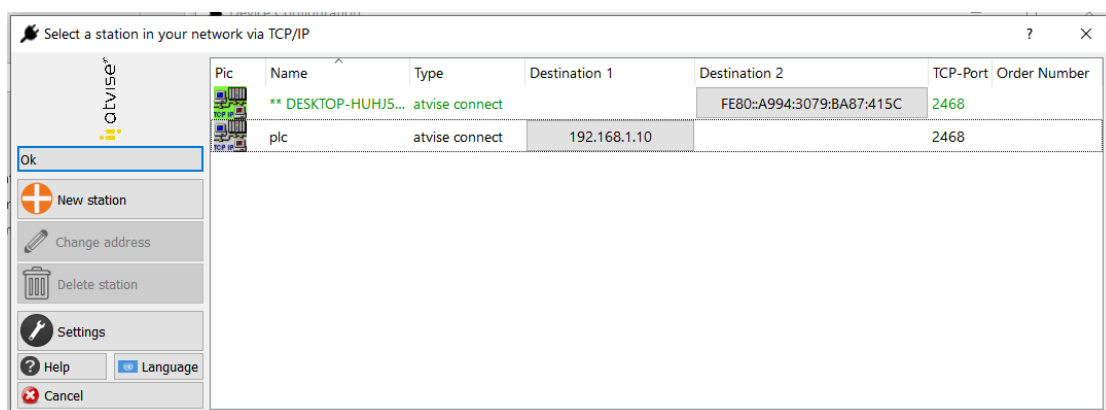


Figure 40: Station selection in Atvise Connect. (Source: Author's own)

Figure 40 above only shows the local server (which is also in the network) and an attempt to add the server located in the PLC. During the development of this thesis, it has only been possible to add the internal server of the OT1210 screen, despite the attempts to locate

the PLC server, it has not been possible to locate it and add it to the menu of stations shown in the previous display. Once a station is added and properly identified, just double-click on it to see the servers and clients within it, in the same way as in the local configuration on the PC. The screen that the user will find in both cases is similar to the one seen in *Figure 41*, which corresponds to the local atvise server located on the PC where the thesis was written.

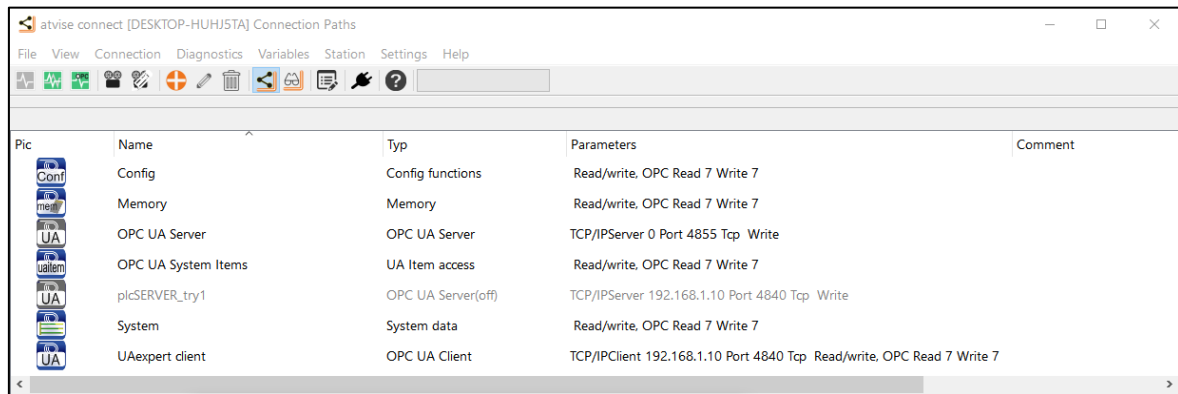




Figure 41: Localhost Atvise Connect station. (Source: Author's own)

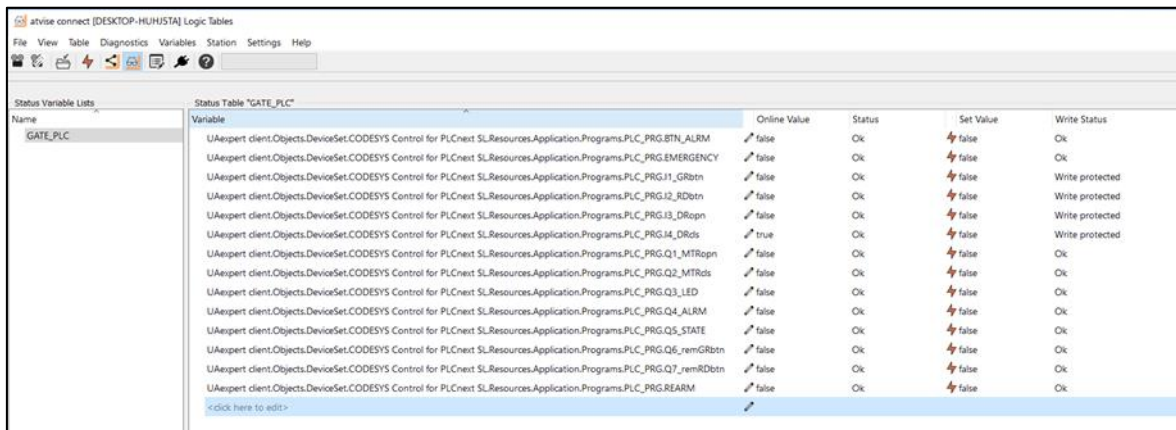
Inside the station can be seen a main window with diverse sources of information, again, these are the elements that Atvise Connect has been able to detect with a first scan. In case it is known that there are more elements connected to that station, there is the option to create a new connection, the process of adding a new station is explained in detail in the YouTube video "How to configure the MQTT client of atvise Connect." (Vester Business, 2022).

For adding a new station, the user can access this option by clicking on the symbol () in the upper area of the window shown in *Figure 41*. This is one of the most important options, as it allows a wide variety of data sources to be added: servers, PLCs, clients, sensors, etc. In addition, these elements are identified by brand and communication protocol (OPC UA, MQTT, Siemens, Phoenix, etc).

When adding a new connection, it is very important, above all, to take into account what type of connection it is, once this is determined, the user can proceed to introduce more specific information. What determines if the connection will succeed, is the correct typing of the IP address and the port of the new element.



In the case of OPC UA clients, on which this thesis is focused, it can be easily added through the "Active Data Request" and "OPC UA client" settings. After completing the required configuration, it is possible to configure the list of variables that the user desire to export to Atvise Builder to work with.

In order to access the variable lists menu, click on the symbol () on the screen's taskbar and a menu like the one in Figure 42 will be displayed.



| Name | Variable | Online Value | Status | Set Value | Write Status |
|----------|---|--------------|--------|-----------|-----------------|
| GATE_PLC | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.BTN_ALARM | ⚡ false | Ok | ⚡ false | Ok |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.EMERGENCY | ⚡ false | Ok | ⚡ false | Ok |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.I1_GRbtn | ⚡ false | Ok | ⚡ false | Write protected |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.I2_RDbtn | ⚡ false | Ok | ⚡ false | Write protected |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.I3_DRbtn | ⚡ false | Ok | ⚡ false | Write protected |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.I4_DRds | ⚡ true | Ok | ⚡ false | Write protected |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.Q1_MTRbtn | ⚡ false | Ok | ⚡ false | Ok |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.Q2_MTRds | ⚡ false | Ok | ⚡ false | Ok |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.Q3_LED | ⚡ false | Ok | ⚡ false | Ok |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.Q4_ALARM | ⚡ false | Ok | ⚡ false | Ok |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.Q5_STATE | ⚡ false | Ok | ⚡ false | Ok |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.Q6_remGRbtn | ⚡ false | Ok | ⚡ false | Ok |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.Q7_remRDbtn | ⚡ false | Ok | ⚡ false | Ok |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.REARM | ⚡ false | Ok | ⚡ false | Ok |
| | <click here to edit> | ⚡ | | | |

Figure 42: Variable list display in Atvise Connect. (Source: Author's: own)

Once in the variable list menu, it is possible to create several variable lists with the name desired by the user. To link these created variables to the value they have in the client, the user must select the option () which each variable has in their "Online Value" column of Figure 42. After selecting this option, a new window will open. In this new window, if the data client has been correctly added, it is possible to navigate through its directories until select the desired variable and link it to the list. After having added the desired variables, click on the symbol () to enable reading and writing of the variable sin Atvise. Once this has been done, the configuration in this program would be finished. The particular case of this thesis where Atvise Connect is linked to the OPC UA UAexpert client is explained in procedural part 6.5.

These are the main aspects of atvise connect that have been considered necessary to explain. However, it is necessary to mention that in case of having a different communication protocol such as MQTT or a Siemens protocol, the actions required to connect the program to these sources of information would have been very similar.

7.4 SCADA visualization on HMI

The following section shows the procedures to follow in order to visualise the different creations and visualisations created with Atvise Builder on our PC on our industrial HMI.

To carry out the steps explained in this section properly, it is recommended to set up the configuration of the screen in the web browser of the personal computer by using the remote TSSW, as explained in section 4 of this thesis.

Also remember that for this section the IP of our ethernet port has been configured with the address (192.168.1.40), which can be any except the one used for the PLC port (192.168.1.97), the PLC (always 192.168.1.10) and the screen (192.168.1.50), this of course, in the configuration examples of this thesis.

To understand what is being done in this section, we are going to transmit via the web the Scada designed on our PC. This will be done by uploading the project via the local network to the Atvise server that we will start on our screen. First, we will modify the HMI settings to start a server on the screen. Then, we will connect to the display via the local network.

Although it is redundant, for this section it will be necessary to connect the display to the PC via the ethernet cable, to have fewer problems it is recommended to simulate exactly the case developed in the example of the thesis. To do this, have the ethernet cable connected to the "eth0" port of the HMI and to the main ethernet port of the computer.

Steps to be taken:

1. Start the TSSW (seen in *Figure 43*) on the web browser as explained in section 4 of this thesis (Introducing the configured IP of the screen in your browser).

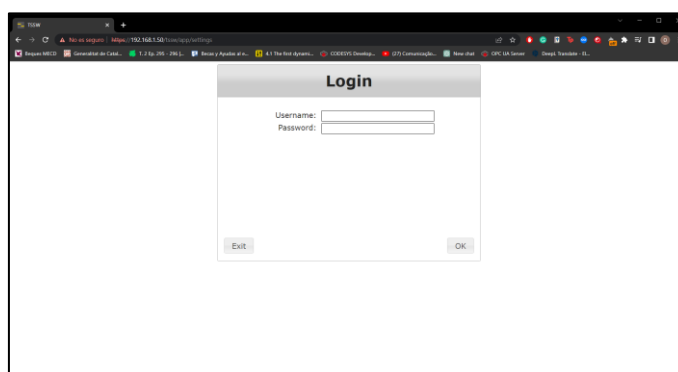


Figure 43: Main TSSW screen (Source: Author's own)

Once started, change the next parameters:

2. In the network settings, change the settings go to the “eth0” section and change to mode to manual. Put the desired IP (in the case of the example of *Figure 44* 192.168.1.50) and the Netmask (always 255.255.255.0).

Network Settings

eth0 eth1 br0 (bridge) Proxy

Mode: Manual

MAC: 00:10:7E:09:6D:9D

IP: 192.168.1.50

Netmask: 255.255.255.0

Gateway:

DNS Server:

Back Cancel OK

Figure 44: Network settings for configuring the SCADA on the HMI (Source: Author's own)

3. In the service settings, go to the Browser section and change the “Enabled” option to YES. Also, put on the “URL” section the link “http://192.168.1.50:8080” as shown in the following *Figure 45*:

Services

Firewall DHCP NTP VNC Browser TSSW

Enabled: YES

URL: http://192.168.1.50:8080

Fullscreen: NO

Remote Inspector: YES

Remote Inspector Port: 9222

F5 Refresh: YES

Back Cancel OK

Figure 45: Browser settings for configuring the SCADA on the HMI (Source: Author's own)

4. Inside the “settings” go to the Atvise configuration and configure the ports of the Atvise server as seen in *Figure 46*. Put the OPCUA port to “4841” and the HTTP port to “8080”.

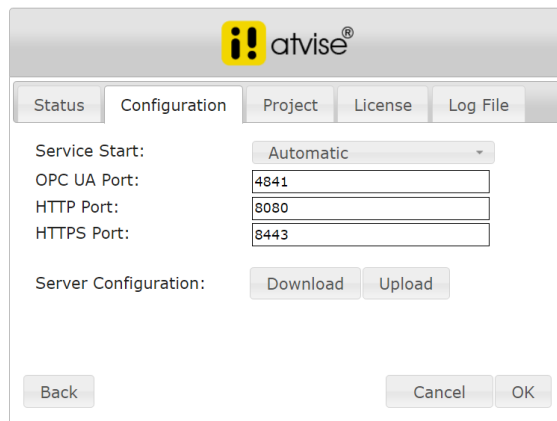


Figure 46: Port configuration for SCADA visualization on the HMI (Source: Author's own)

5. Finally, it will be needed to restart the screen to apply all the changes and modifications made in the previous steps. Then, all that remains is to go to the "status" section within the Atvise settings where the current status of the server is displayed and start it with the "start server" button as shown in *Figure 47*, in case the server has started without problems and remains stable, go to the next and last step.

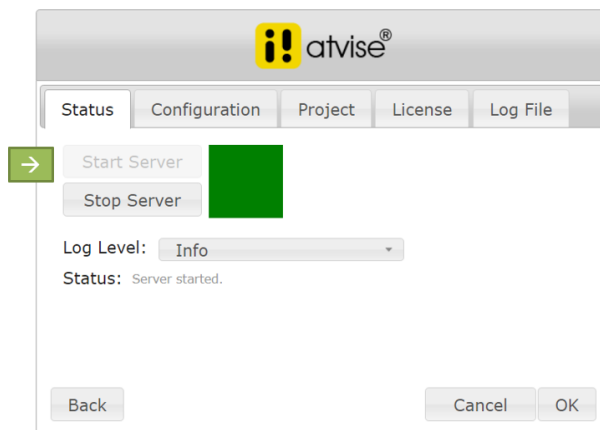


Figure 47: Server status screen in TSSW (Source: Author's own)

In case the server does not start properly or if there are any problems, remember that there must be no compatibility error between the server licence and the version of atvise. To avoid these kinds of errors and problems the explanations made in section 4.3 of this thesis should have been followed.

6. In order to start designing the SCADA that will be displayed on the HMI. Start normally Atvise monitor as administrator and by clicking on the gear symbol create a new project, when accessing the builder, enter the URL of our new server (opc.tcp://192.168.1.50). In case the working version is other than 3.4, atvise asks for a username and password, these are "root" and a blank space respectively.

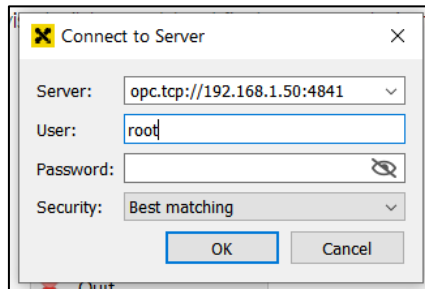


Figure 48: Pop up screen to connect to HMI server. (Source: Author's own)

Once communication has been established and the various displays of our project have been designed. When the HMI is started (if you do not access the settings via the initial 5-second alert), the Bachmann industrial browser window will open automatically, and the default Atvise menu with the displays and alarms of the project will be seen, an example of this can be seen in Figure 49.

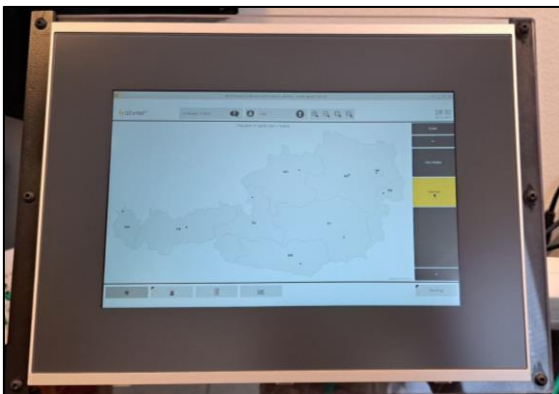


Figure 49: SCADA visualization in the HMI. (Source: Author's own)

In the case that atvise asked before for user and password, it will be necessary to log in by clicking on the padlock symbol located in the upper central part of the HMI screen.

From this moment on the displays can be updated in real time and observe the effect of the changes made on the PC at the moment. To do this, save the project on the computer and double-click on the "main" part of the HMI display interface.

Although it has been possible to visualize the displays created on the OT1200 screen, it is not possible to create a fully functional SCADA linked to the PLC server. This is because, after many attempts and tests with the software, it has only been possible to visualize in the SCADA the displays created on the local server located on the screen. In the same way, the variables that can be worked with are those included in this local server, without having managed to export the variables from the existing server in the PLC. This is one of the points that have not been reached and that needs to be deepened in future theses and research.

8 SCADA development

In the next and last section of this thesis, the aim is to explain the particular case of SCADA that has been programmed and created. This section will show a practical example of the operation of an industrial OPC UA system controlled using an HMI.

To replicate the work explained in this section, the previous sections of the thesis will be used as support. Where, initial configurations of the different programs and some general processes are explained, even if a different SCADA is to be developed. The processes for example of creating a server or exporting variables to the client will always be the same regardless of the number of variables or the PLC being used. For this reason, processes such as software installation, license activation, and various system configurations will not be repeated. In other words, this section focuses on the particular case developed during this thesis, and no general settings and explanations that were made before in previous sections.

8.1 Hardware behaviour

In order to start developing the system it is necessary to know how it is desired the system to behave in front of the different digital inputs and outputs that will be used. As explained in section 3.5, the hardware to be controlled is a miniature industrial door delivered by NOVIA University which counts with 4 digital inputs and 4 digital outputs.

With those 8 digital signals complete control must be done. The 4 digital inputs are: 2 motors for closing and opening the door, a LED, and an audible alarm. The digital outputs of the door are: a green button, a red button, a fully open door limit switch and a fully closed door limit switch. Connection PLC -gate is explained in Appendix 3.

The behaviour chosen for the door is as follows:

When the green button is activated, the door starts to move, in case the door is closed it will start to open, in case it is open, it will start to close. Once the door has opened or closed completely, the motor stops. The opening and closing movements can be alternated only by pressing the green button when the previous movement is finished.

The red button can stop the door movement indefinitely. Until the green button is pressed again, the movement will not continue. When pressing the green button to reactivate the movement, if the door was opening, it will continue to open, if it was closing, it will continue to close.

The LED shall be activated while the door is opening. As for the alarm, it can only be activated remotely via the SCADA, which will have an emergency stop button that will stop the system completely and activate the alarm until the emergency button is lifted. By industry standards, the system will not return to normal operation until the emergency button is lifted and a blue reset button is pressed. This emergency system is only found in the final SCADA.

An important clarification is that the door's physical system must function regardless of whether the door is in operation or not. The way in which the programme should be developed to achieve this objective will be explained in the section on programme creation.


8.2 PLC programming

Before the beginning of program creation, the steps explained in section 6.1 (*PLC connection and basic programming*) must have been followed. In that section, it is explained how to install CodeSys. Also, how to configure the equipment to be able to connect to the PLC model AXC F and start loading functional programs. Once these steps have been completed, the following explanation can be continued>

The first step to start programming our PLC is to know the variables to work with. Due to the need of control at least the 4 inputs and the 4 outputs that the industrial gate has physically, at least, it will be needed to add these 8 variables to the program. In *Figure 50*, it can be seen the variable list of CodeSys with the 4 inputs and outputs listed orderly.

| | Scope | Name | Address | Data type | Initialization |
|---|-------|------------------|---------|-----------|----------------|
| 1 | VAR | I1_GRbtn | %IX0.1 | BOOL | |
| 2 | VAR | I2_RDbtn | %IX0.2 | BOOL | |
| 3 | VAR | I3_DRopn | %IX0.3 | BOOL | |
| 4 | VAR | I4_DRcls | %IX0.4 | BOOL | |
| 5 | VAR | Q1_MTRopn | %QX0.1 | BOOL | |
| 6 | VAR | Q2_MTRcls | %QX0.2 | BOOL | |
| 7 | VAR | Q3_LED | %QX0.3 | BOOL | |
| 8 | VAR | Q4_ALARM | %QX0.4 | BOOL | |
| 9 | VAR | Q5_STATE | %QX0.5 | BOOL | |

Figure 50: Variable list table in CodeSys. (Source: Author's own)

To add variables and configure which is the memory address in the PLC, click on the "Tabular View" icon () located in the upper right area of the PLC_PRG window. The style of the window will change from code format to variable list format. Once in there, new variables can be added by right clicking and selecting "Insert". Also, an extra signal (Q5) has been added to let the user know in which direction the door is moving.

In this case, the name chosen for the variables is completely indicative and has been considered intuitive for programming. For example, the first variable seen in *Figure 50*, the variable green button corresponding to input 1, has been written as (input1) + (GREEN) + (button) = I1+GR+btn = I1_GRbtn . In no case does the name of the variable affect the development of the program. Due to the nature of our system, all the variables used in the SCADA are Boolean variables that can only have a value of 0 or 1.

Other signals to be added are those that will later be used to configure the operation of the emergency stop and reset that will be found in the SCADA. To do this, 3 different signals will be necessary, the one that indicates the value of the alarm button, the one that indicates the value of the reset button and an alarm status that will block the system until the reset is pressed. As shown in *Figure 51*, the variables are set as Boolean type.

| | | | | |
|----|-----|------------------|--------|------|
| 12 | VAR | EMERGENCY | %QX1.0 | BOOL |
| 13 | VAR | BTN_ALARM | %QX1.1 | BOOL |
| 14 | VAR | REARM | %QX1.2 | BOOL |

Figure 51: Emergency Stop signals in CodeSys. (Source: Author's own)

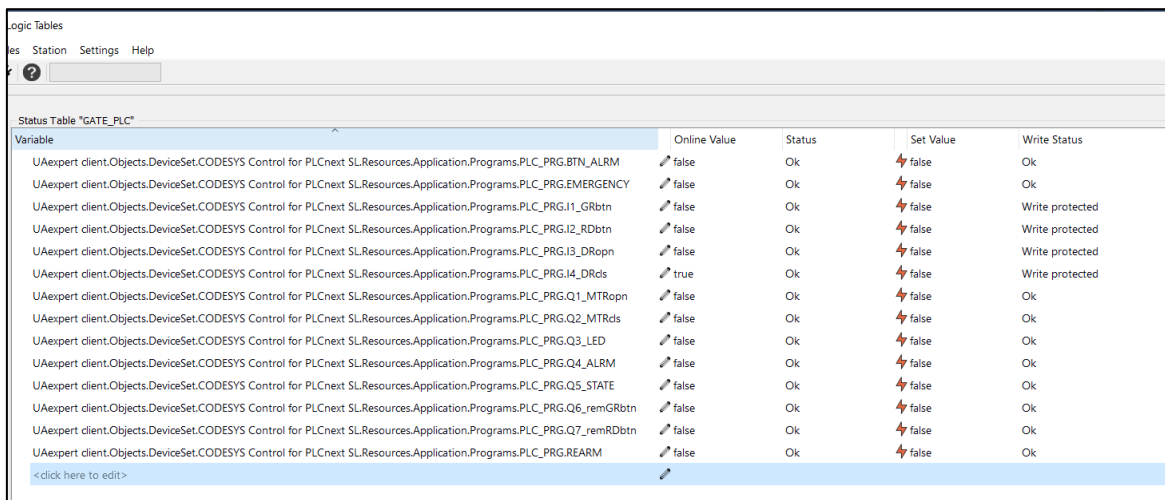
These three signals seen in *Figure 51* will be added as outputs for the reasons explained below. Later, in order to avoid confusions with the variables names and functions, a description of each signal associated with its exact name is given in Table 1.

Once the behaviour of our system is clear and the initial signals have been added, it is necessary to take into account that extra variables will be necessary for our system to work properly. The need for extra variables is mainly because two problems need to be taken into account. This is because the system must work even if the connection to SCADA is offline. And that it is not possible to edit the variables that are inputs to the PLC using SCADA.

Although these problems are discussed at the end of section 6.2.4 of this thesis. Next, it is shown how they will be solved and the reasoning behind the choice of the different variables that will be added to our system.

Issue 1: Inputs cannot be edited remotely:

After working on the Atvise and CodeSys software, it has been seen that Atvise Connect has no rights to set write rights on variables that are inputs, this can be seen in *Figure 52*, where the outputs are on write status "Ok" but the inputs are in "Write protected" state.



| Variable | Online Value | Status | Set Value | Write Status |
|---|--------------|--------|-----------|-----------------|
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.BTN_ALARM | false | Ok | false | Ok |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.EMERGENCY | false | Ok | false | Ok |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.I1_GRbtn | false | Ok | false | Write protected |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.I2_RDbtn | false | Ok | false | Write protected |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.I3_DRopn | false | Ok | false | Write protected |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.I4_DRds | true | Ok | false | Write protected |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.Q1_MTRopn | false | Ok | false | Ok |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.Q2_MTRds | false | Ok | false | Ok |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.Q3_LED | false | Ok | false | Ok |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.Q4_ALARM | false | Ok | false | Ok |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.Q5_STATE | false | Ok | false | Ok |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.Q6_remGRbtn | false | Ok | false | Ok |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.Q7_remRDbtn | false | Ok | false | Ok |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL.Resources.Application.Programs.PLC_PRG.REARM | false | Ok | false | Ok |

Figure 52: Variable list in Atvise Connect. (Source: Author's own)

The inputs of our system that are operated by the user are the two red and green buttons located next to the industrial gate. Unlike the limit switches, which must not be activated by the user, it must be possible to write the value of the buttons remotely in case it is

desired to operate the system with a SCADA. To remedy this problem, some new variables which are outputs have been added.

These new variables will act as inputs for the SCADA red and green buttons and will work in the same way as the physical buttons of the hardware. However, these variables will be configured as outputs within the CodeSys software. In this way, when exporting and connecting to the Atvise server, write permissions on these variables can be obtained.

In CodeSys, these variables, as can be seen in *Figure 53* will be entered as green button remote (Q6_remGRbtn) and red button remote (Q7_remRDbtn) and will be programmed in the same way as the green and red buttons, by placing a contact in parallel where the physical button contacts are located.

| | | | | |
|----|-----|-------------|--------|------|
| 10 | VAR | Q6_remGRbtn | %QX0.6 | BOOL |
| 11 | VAR | Q7_remRDbtn | %QX0.7 | BOOL |

Figure 53: Variables for SCADA buttons in CodeSys. (Source: Author's own)

With this, all the necessary variables have already been added to the program.

Issue 2: System must work with SCADA offline:

This problem is explained with examples and details in section 6.2.4 of this paper. However, it is discussed again here to explain the reasoning behind how the PLC has been programmed. It would be possible to use a lower number of variables in case of directly configuring the actions in the SCADA. This means that, if in the SCADA program it is configured that pressing the button activates the door, it would be possible to save a line of the contact diagram in CodeSys.

However, if this is done, when the SCADA is disconnected, the industrial gate would be blocked and would not respond to the physical green button, as there is no program that relates the action of pressing the green button with opening the gate. This is a great risk in industry, where breakdowns can occur, and the stoppage of a machine or gate can suppose a human or economic danger. For this reason, the complete program that is loaded into the PLC must be fully functional even if the OPCUA server is not operational.









For this reason, the code will contain the complete functionality that the door must have. The SCADA will act on the input values of information such as the buttons, and it will be the CodeSys program loaded in the PLC that will execute the program and perform the actuation of the outputs by means of the imposed conditions.

Once the variables that will be needed have been clearly defined, in *Table 1* shown below, all the variables can be identified together with their memory address in the PLC and the type of signal they are with respect to the PLC (if its input or output).

In *Table 1*, in the "Description" column, there is an explanation of what represents each signal and how the change in the value of the variable works. In the "Action" column, it is explained which is the action that the variable will perform in our system, if it activates some mechanism, or if it represents some state that will help the programming.


In addition, the "Image" column shows where the signals come from or what they are associated with. In case it is an external physical element of the hardware, this element is shown. In the opposite case, if it is a button or SCADA element, the button associated with the SCADA is shown. In the last case in which they are states or internal elements of the hardware, no image is shown in the column and only the word "internal" appears.


Table 1: General characteristics signal list from PLC program.

| Variable name | Address | Type | Description | Action | Image |
|---------------|---------|--------|--|--|---|
| I1_GRbtn | %IX0.1 | Input | Physical green button, returns 1 when is pressed. | Activates gate movement |  |
| I2_RDbtn | %IX0.2 | Input | Physical red button, returns 1 when is pressed. | Stops door movement |  |
| I3_DRopn | %IX0.3 | Input | Door open limit switch, sends a 1 if the door is fully open | Indicates door state | internal |
| I4_DRcls | %IX0.4 | Input | Door close limit switch, sends a 1 if door is fully close | Indicates door state | internal |
| Q1_MTRopn | %QX0.1 | Output | Open gate motor, if value is 1, activates opening door motor | Activated by green button, opens the door | internal |
| Q2_MTRcls | %QX0.2 | Output | Close gate motor, if value is 1, activates closing door motor | Activated by green button, closes the door | internal |
| Q3_LED | %QX0.3 | Output | LED, if value is 1 turns on the LED of the gate | Activated when door is moving |  |
| Q4_ALRM | %QX0.4 | Output | Sound Alarm, if value is 1 turns on the alarm placed in the gate | Sounds only while Alarm button is pressed |  |
| Q5_STATE | %QX0.5 | Output | Direction State, if door is opening value will be 1, if door is closing value will be 0. | Indicates door behaviour | internal |
| Q6_remGRbtn | %QX0.6 | Output | SCADA green button, returns 1 when is pressed | Activates door movement |  |
| Q7_remRDbtn | %QX0.7 | Output | SCADA red button, returns 1 when is pressed | Stops door movement |  |
| EMERGENCY | %QX1.0 | Output | Emergency state variable, if 1, system is in emergency state | Block the system until rearm button is pressed | internal |
| BTN_ALRM | %QX1.1 | Output | Emergency Stop Alarm button, returns 1 when pressed | Activates emergency state (until rearm) and alarm (until released) |  |
| REARM | %QX1.2 | Output | Restart button, when pressed returns 1, deactivates emergency state | Deactivates emergency state |  |

(Source: Author's own)

Once the signals involved in the system and their actions are known. The PLC programming can be started through the contact and coil diagram (LD). Another programming method could have been chosen, but this is the most common in the industry and the most intuitive for the programmer.

As a summary, for the programming of the PLC, an LD diagram has been created. In this diagram, there is a total of 5 different elements used. The most basic element is the contact () , when a bit is set in the contact, it will allow current to flow through it and the trigger flow to continue. In case there is a "/" inside the contact, it is the reverse condition, only the trigger flow can flow when the bit in the contact is zero.

The other 3 elements that can be found in the created LD are coils () . In case they are empty, they only place a 1 in the corresponding bit while they are receiving current. In case there is an "S" even if they only receive current for an instant, a 1 will be placed in the bit to which they are linked until a coil referenced to the same bit is activated but with the letter "R" inside it.

Starting with the programme, in the first network seen in *Figure 54* it can be seen that, both the physical and the remote green buttons, simply by being pressed and not held, will activate the action of the door opener motor. The motor will change its value to 1 (set) until it is reset to zero. As seen, activation of the opening motor will only occur if the door is not fully open, the Q5 status is not activated, and the emergency status is not activated.

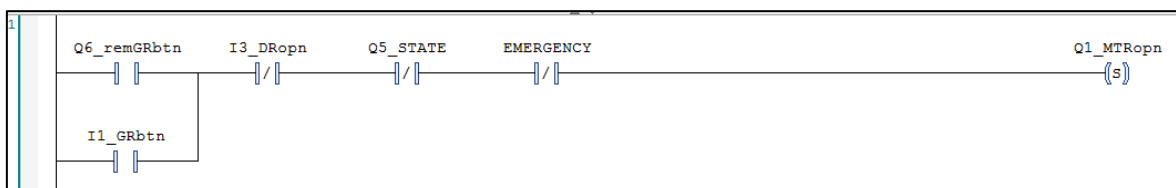


Figure 54: Network 1 of the PLC program. (Source: Author's own)

Next, in the second network seen in *Figure 55*, it can be seen that both the physical and the remote red buttons perform the same action. Simply by being pressed and not held, will reset the opening motor and change its value to 0. By doing this the door will remain stopped. As in network 1, this action cannot happen if the emergency status is activated.



Figure 55: Network 2 of the PLC program. (Source: Author's own)

On the other hand, networks 3 and 4 shown in *Figure 56* and *Figure 57* show the same behaviour but closing instead of opening. In network 3 it is also activated by the action of the green buttons, but in this case, the restriction of the state of Q5 is the opposite, this action can only happen in case Q5 is 1. Same as in the opening, if the emergency status is active or the door is closed, the action will not be able to pass.

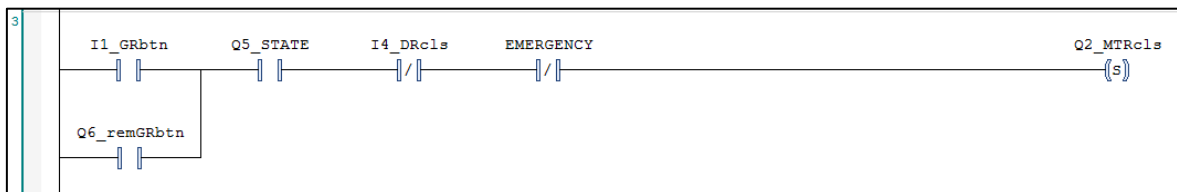


Figure 56: Network 3 of the PLC program. (Source: Author's own)

As in the stop function for the opening function (Network 2) in Network 4 seen in *Figure 57* either the remote or the physical red button can turn the motor to 0 and stop the gate, in this case, the closing motor. The motor will also stop if the door is completely closed and activates the limit switch. This network can only function if the emergency signal is not activated.

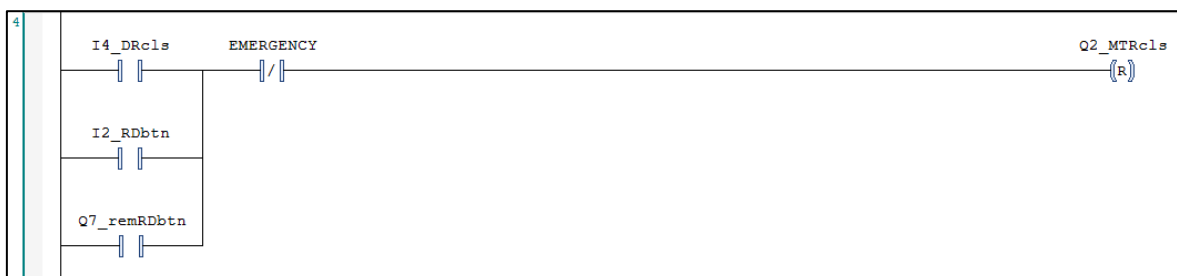


Figure 57: Network 4 of the PLC program. (Source: Author's own)

Next, in *Figure 58*, It can be seen networks 5 and 6. These networks are concerned with changing the variable Q5, this “state” indicates whether the door is closing or opening. To do this, when the limit switch is activated, the state of the door is alternately changed from 1 to 0, and when opposite, from 0 to 1. So that, after a door closing there will always be an opening, this will perform an infinite cycle. When the Q5 value is 0 the door is in the opening process, when it is in the closing process the value is 1. As in the previous networks, this can only occur when emergency state is not activated.

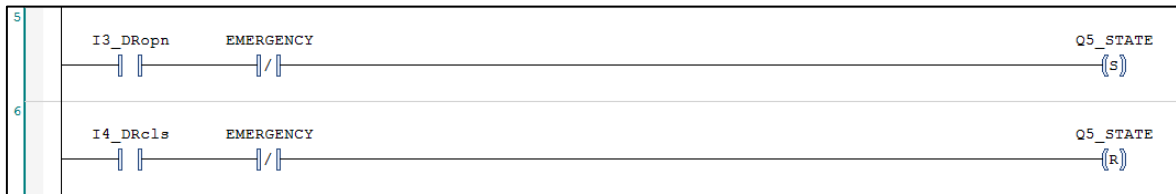


Figure 58: Network 5 and 6 of the PLC program. (Source: Author’s own)

To control the operation of the LED, network 7 shown in *Figure 59* is used. In this case, whenever a motor is running, the LED will be activated. If the system is in emergency mode, this network will not work.

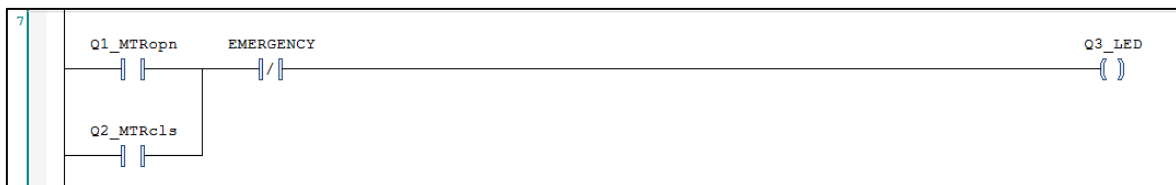


Figure 59: Network 7 of the PLC program. (Source: Author’s own)

Finally, *Figure 60* shows the 2 networks that control the operation of the emergency system, these are networks 8 and 9. In network 8 it can be seen how when the emergency button variable is activated, the emergency status variable is set to 1 indefinitely, which will block all the other networks apart from 8 and 9. As seen, while the alarm button is pressed, the audible alarm will be working. When the emergency button is pressed, the engines are also stopped indefinitely.

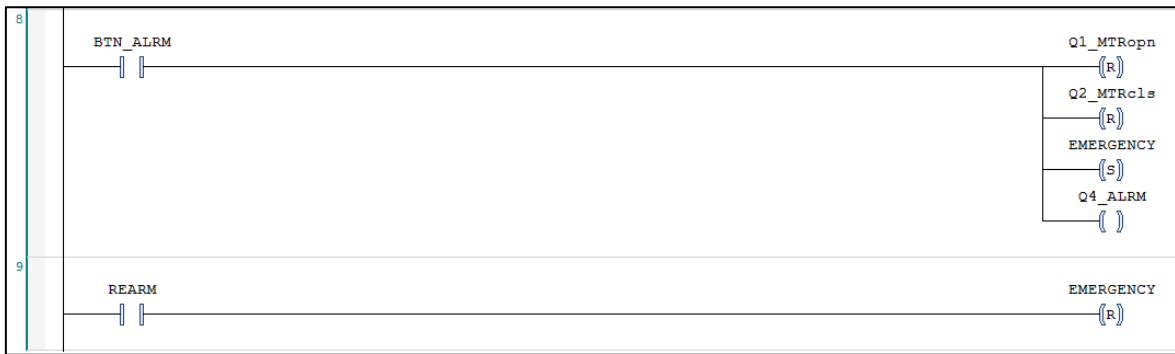


Figure 60: Network 8 and 9 of the PLC program. (Source: Author's own)

Once the above 9 networks have been created, the program that will control the behavior of the PLC is complete. Next, to continue with the development of the SCADA system, the user is required to follow the steps explained in section 6.2 of this thesis (OPC UA Server PLC installation and client connection). This section explains how, after having created a program in CodeSys, to install an OPC UA server on the PLC. Once created, it also explains how to connect it to a local OPC client through the UAexpert application.

8.3 Client to SCADA communication

Once the user has properly followed the steps in the previous section, it is assumed that the variables created in section 8.1 (Hardware behaviour) above are displayed in real time in the UAexpert client in the same way it is seen in *Figure 61*. At this point the server has been successfully created, and locally, the door - PLC system works perfectly (opening and closing system with the red and green buttons).

| # | Server | Node Id | Display Name | Value | Datatype | Source Timestamp | Server Timestamp | Statuscode |
|----|---------------|--|--------------|-------|----------|------------------|------------------|------------|
| 1 | OPCUAServe... | NS4[String]var[CODESYS Control for PLCnext SLApplication.PLC_PRG.I1_GRbtn | I1_GRbtn | false | Boolean | 10:44:07.989 | 10:44:07.989 | Good |
| 2 | OPCUAServe... | NS4[String]var[CODESYS Control for PLCnext SLApplication.PLC_PRG.I2_RDbtn | I2_RDbtn | false | Boolean | 09:59:54.790 | 09:59:54.790 | Good |
| 3 | OPCUAServe... | NS4[String]var[CODESYS Control for PLCnext SLApplication.PLC_PRG.I3_DRopn | I3_DRopn | false | Boolean | 11:13:40.087 | 11:13:40.088 | Good |
| 4 | OPCUAServe... | NS4[String]var[CODESYS Control for PLCnext SLApplication.PLC_PRG.I4_DRcls | I4_DRcls | true | Boolean | 11:13:41.891 | 11:13:41.891 | Good |
| 5 | OPCUAServe... | NS4[String]var[CODESYS Control for PLCnext SLApplication.PLC_PRG.Q1_MTRopn | Q1_MTRopn | false | Boolean | 11:04:10.195 | 11:04:10.195 | Good |
| 6 | OPCUAServe... | NS4[String]var[CODESYS Control for PLCnext SLApplication.PLC_PRG.Q2_MTRcls | Q2_MTRcls | false | Boolean | 11:13:49.089 | 11:13:49.089 | Good |
| 7 | OPCUAServe... | NS4[String]var[CODESYS Control for PLCnext SLApplication.PLC_PRG.Q3_LED | Q3_LED | false | Boolean | 11:13:40.690 | 11:13:40.690 | Good |
| 8 | OPCUAServe... | NS4[String]var[CODESYS Control for PLCnext SLApplication.PLC_PRG.Q4_ALARM | Q4_ALARM | false | Boolean | 11:27:06.490 | 11:27:06.490 | Good |
| 9 | OPCUAServe... | NS4[String]var[CODESYS Control for PLCnext SLApplication.PLC_PRG.Q5_STATE | Q5_STATE | false | Boolean | 11:13:49.089 | 11:13:49.089 | Good |
| 10 | OPCUAServe... | NS4[String]var[CODESYS Control for PLCnext SLApplication.PLC_PRG.Q6_remGRbtn | Q6_remGRbtn | false | Boolean | 11:03:57.189 | 11:03:57.189 | Good |
| 11 | OPCUAServe... | NS4[String]var[CODESYS Control for PLCnext SLApplication.PLC_PRG.Q7_remRDbtn | Q7_remRDbtn | false | Boolean | 11:03:18.789 | 11:03:18.789 | Good |
| 12 | OPCUAServe... | NS4[String]var[CODESYS Control for PLCnext SLApplication.PLC_PRG.REARM | REARM | false | Boolean | 11:13:49.280 | 11:13:49.280 | Good |
| 13 | OPCUAServe... | NS4[String]var[CODESYS Control for PLCnext SLApplication.PLC_PRG.EMERGENCY | EMERGENCY | false | Boolean | 11:28:12.679 | 11:28:12.679 | Good |
| 14 | OPCUAServe... | NS4[String]var[CODESYS Control for PLCnext SLApplication.PLC_PRG.BTN_ALARM | BTN_ALARM | false | Boolean | 11:27:06.557 | 11:27:06.557 | Good |

Figure 61: SCADA variable list in UAexpert client. (Source: Author's own)

Next, it is explained the process of exporting the UAexpert real-time variables to the Atvise Builder software. To do so, we will mainly make use of Atvise Connect and in the last steps of Atvise Builder. For this reason, for the user to be able to carry out the steps correctly and

without any problems, it is recommended to read section 7.3 of this thesis beforehand. Where the basic operation of Atvise Connect is explained.

Once the atvise Builder and Atvise Connect softwares have been installed and licensed, the process can be started by launching Atvise Connect. Once in the initial window, choose the option "Configure product over network". In the new screen, select the first station of the list, that corresponds to the local host of Atvise running on the computer. The local host server in Atvise Connect is always written in green letters in order to differentiate it from the other ones. Do double click on the station to connect and access its settings, once connected to the localhost server, a window like the one seen in Figure 62 will be seen.

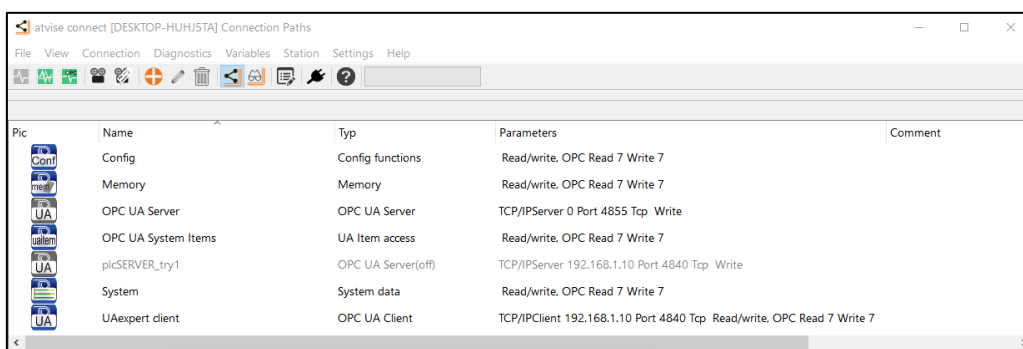


Figure 62: Localhost station in Atvise Connect. (Source: Author's own)

Then, by clicking on the symbol () in the upper area of the window shown in Figure 62, the process of adding an OPC UA client of UAexpert is going to be started. In the new window that has opened, enter the desired name for the client and then select "Active data request" and tick the option "OPC UA client". Then click on "Next" and a new window will open. In this new window, in the "Connection URL" section, I will be needed to enter the IP address of the client that has already been used in UAexpert (192.168.1.10) : (4840) as seen in Figure 63. Nothing else needs to be changed from this window, click on "Next".

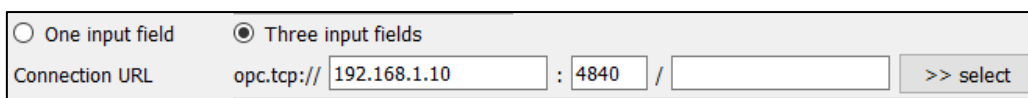




Figure 63: IP and gate configuration in Atvise Connect. (Source: Author's own)

In the last window that will open, make sure that the "Connection Active" and "write allowed" boxes are checked, then click "Save" and the client will have been successfully added.

Afterwards, it is going to be checked if the client is working correctly, also, how to configure the read and write rights on the variables exported to Atvise Builder.

To do so, go to the variable list window by clicking on the icon () located on the top toolbar. The display of the window will have changed, add a new list of variables by right clicking on the space on the left of the screen and clicking on the option "Add variable list". After creating a new list, add one by one the variables wished to export to Atvise Builder. To do this, click on the icon () located in the variables area. Then, in the new display that will be opened follow the path shown in *Figure 64*. Following that path will open the directory of the client where the variables are stored.

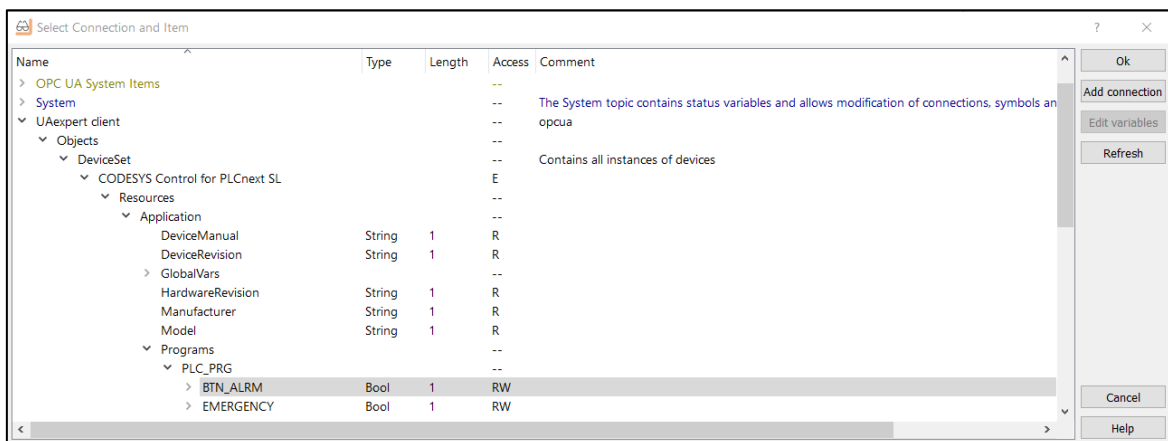


Figure 64: Variables directory path in CodeSys. (Source: Author's own)

Once in the variable's directory, click on the first variable the user desires in the list, then, the window will close automatically, and the variable will be added to the list. This process must be repeated for each one of the variables until the variable list looks like *Figure 65* as seen below. In case a variable has not been added correctly, it will be written in red, and the type of issue will be indicated in the status column.

The screenshot shows the 'Atvise connect [DESKTOP-HUHJ5TA] Logic Tables' interface. The main window displays a table titled 'Status Table "GATE_PLC"'. The table has five columns: 'Name', 'Variable', 'Online Value', 'Status', 'Set Value', and 'Write Status'. The 'Name' column contains 'GATE_PLC'. The 'Variable' column lists 16 variables, each with a long path starting with 'UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG...'. The 'Online Value' column shows values like 'false' or 'true'. The 'Status' column shows 'Ok'. The 'Set Value' column shows 'false' with a lightning bolt icon. The 'Write Status' column is currently blank.

| Name | Variable | Online Value | Status | Set Value | Write Status |
|----------|---|--------------|--------|-----------|--------------|
| GATE_PLC | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.BTN_ALARM | false | Ok | false | |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.EMERGENCY | false | Ok | false | |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.I1_GRbtn | false | Ok | false | |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.I2_RDbtn | false | Ok | false | |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.I3_DRopn | false | Ok | false | |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.I4_DRds | true | Ok | false | |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.Q1_MTRopn | false | Ok | false | |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.Q2_MTRds | false | Ok | false | |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.Q3_LED | false | Ok | false | |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.Q4_ALARM | false | Ok | false | |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.Q5_STATE | false | Ok | false | |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.Q6_remGRbtn | false | Ok | false | |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.Q7_remRDbtn | false | Ok | false | |
| | UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.REARM | false | Ok | false | |

Figure 65: Variable list added successfully in Atvise Connect. (Source: Author's own)

Then, click on the "write values" (⚡) symbol located in the toolbar of Figure 65, this will ask the client for read and write rights for the variables that have been added to the list. After doing so, a load will be started and the "write status" column, which was previously blank, will show the write rights for each variable. As can be seen in Figure 66, not all rights granted will be equal.

The screenshot shows the same table as Figure 65, but now the 'Write Status' column is populated. The values are 'Ok' for some variables and 'Write protected' for others. A yellow box highlights the 'Write Status' column, and a blue box highlights the 'Write protected' entries. Arrows labeled 'Outputs' and 'Inputs' point to the 'Ok' and 'Write protected' entries respectively.

| Variable | Online Value | Status | Set Value | Write Status |
|---|--------------|--------|-----------|-----------------|
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.BTN_ALARM | false | Ok | false | Ok |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.EMERGENCY | false | Ok | false | Ok |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.I1_GRbtn | false | Ok | false | Write protected |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.I2_RDbtn | false | Ok | false | Write protected |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.I3_DRopn | false | Ok | false | Write protected |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.I4_DRds | true | Ok | false | Ok |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.Q1_MTRopn | false | Ok | false | Ok |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.Q2_MTRds | false | Ok | false | Ok |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.Q3_LED | false | Ok | false | Ok |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.Q4_ALARM | false | Ok | false | Ok |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.Q5_STATE | false | Ok | false | Ok |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.Q6_remGRbtn | false | Ok | false | Ok |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.Q7_remRDbtn | false | Ok | false | Ok |
| UAexpert client.Objects.DeviceSet.CODESYS Control for PLCnext SL_Resources.Application.Programs.PLC_PRG.REARM | false | Ok | false | Ok |

Figure 66: Variable list with write status column updated. (Source: Author's own)

As indicated by the color coding in Figure 66, only variables that are outputs can obtain read and write rights. Inputs can only be read, but not written. Write or read refers to being able to edit the value remotely or just to see what the value is. In case the user wants to check that the signals with "Ok" of "write status" are outputs, please refer to Table 1.

Once this last step has been completed, these variables can be exported to the Atvise Builder project in which the SCADA will be created.

As previously mentioned at the beginning of this subsection, it is necessary to have followed the steps explained in the section 6.1 (Atvise PC initial configuration) of this thesis. This section explains how to install, activate the licenses, and start Atvise Builder correctly. In case these steps have not followed properly, it is possible that the following explanation done in Atvise Builder could not be done as desired.

Once inside Atvise Builder, go to the local server via the left branch menu and right click on the "Data sources" option. From the options that appear, select "Add Data Source" > "atvise connect". A new menu will open as shown in Figure 67, in which it is required to enter the name (could be anything) and the address to which the local atvise connect server is associated. The address to enter is: "opc.tcp://localhost:4855", this is the address associated with Atvise Connect server port.

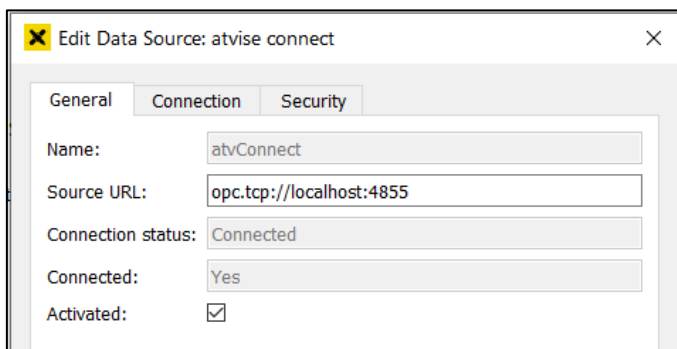


Figure 67: Data source setting in Atvise Builder. (Source: Author's own)

After, go to the "Connection" window and set the "Licensing method" option to "internal". The other settings do not need to be edited or changed, confirm with "Ok" and the Atvise Connect data source will be successfully added.

In order to browse the directory where the variables are located, right click on the new source of information that will have been added in the branch menu and click on the "Browse" option.

After this, a new tab will open in which it is possible to browse through a large number of directories of the client, the licenses, and settings of the local OPC UA server. It is recommended to use Figure 68 to get to the directory in which the project figures are stored.

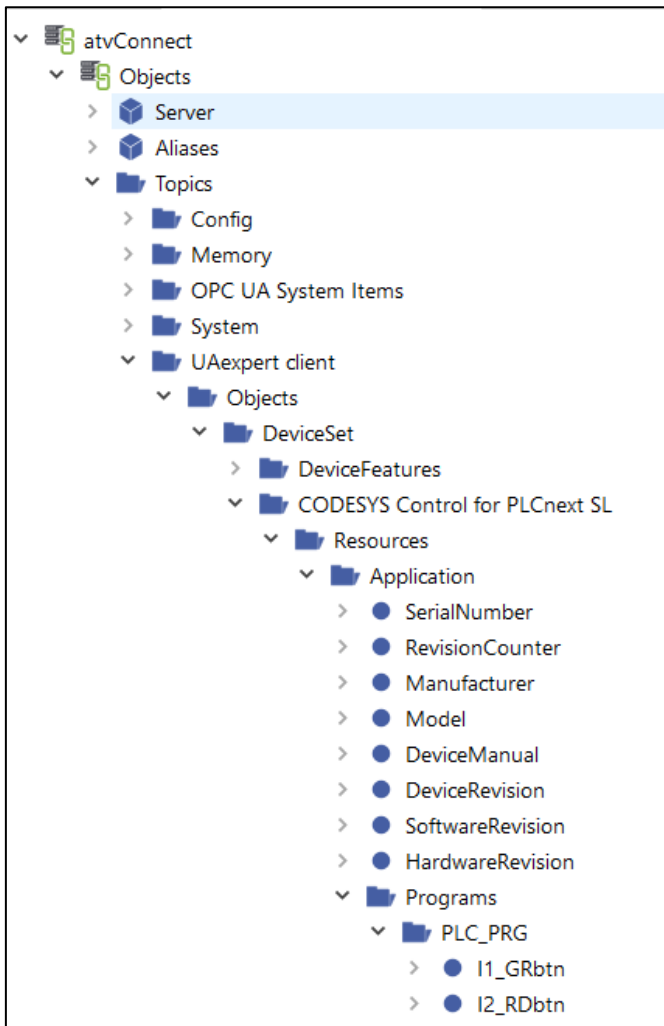


Figure 68: Variable directory in the Atvise Builder browse option. (Source: Author's own)

Once the directory is open, select all variables and copy them to the clipboard. Then go to the left branch menu and in the "OBJECTS" folder inside the local server right click and, in the same way as in *Figure 69*, copy the variables by mirroring them.

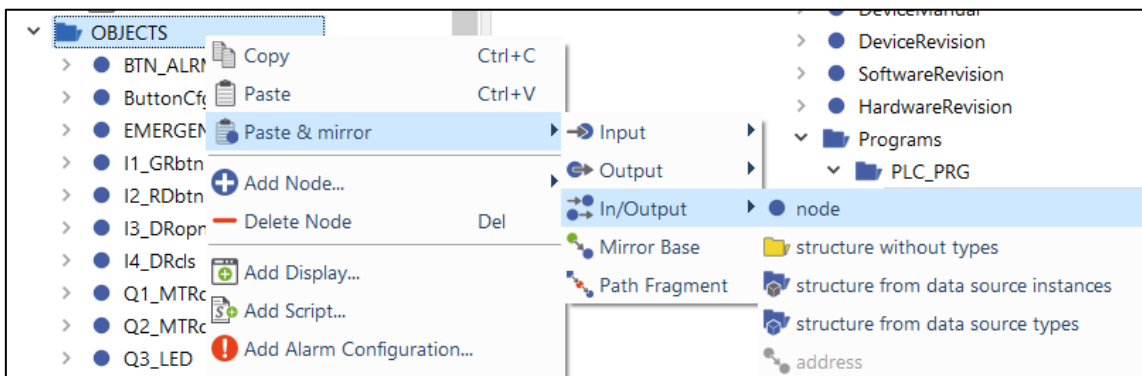


Figure 69: Copying and mirroring method in Atvise Builder project. (Source: Author's own)

The procedure of adding these variables by mirroring is done so that in case there is a change in the server, the variables that are in the SCADA are at all times linked to their original location and change at their current real value. The steps of how to mirror a variable in atvise has been done by following the YouTube official atvise video “atvise Data Aquisition” (atvise, 2020) . The In/Out option is used to be able to write and receive information from the variables at the same time. With this procedure done, the variables from the OPC UA server are now available for use in the SCADA elements and can be passed to the design part of the SCADA.

8.4 SCADA design and programming

In this last section of the thesis, it is intended to explain in detail the steps followed to achieve the final visualization of the created SCADA. To do so, the final visualization created will be presented and the different SCADA components will be explained individually, as well as their function, behavior, and action on the system, if applicable. The SCADA design presented in the following section has been based on the tutorial set provided by atvise on the company's official YouTube channel “atvise video Tutorials (English)” (atvise Youtube, 2015).

It is necessary to say that some of the general elements that do not have a function are considered not necessary to explain. These non-explanatory elements are background color, NOVIA University header, outline color, etc. In order to know how the tools of the display creation toolbars work and to be able to replicate these features, it is recommended to read section 7.2.3 (Display creation) of this thesis.

From the various default screens available in atvise, the "Main" screen at the top of the displayed directory has been chosen to set up the SCADA. This display is located in the branch menu of the server, as shown in *Figure 70*. This is because in this way the SCADA will be the first thing to be seen when initializing the display in the user's browser.

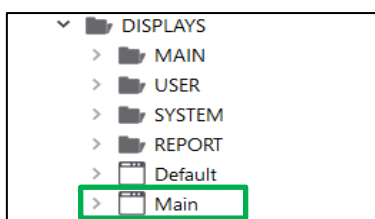


Figure 70: Main display tree branch menu location. (Source: Author's own)

Having said this, the final visualization of the SCADA in operation is shown below in *Figure 71*. This is the model that can be seen inside Atvise Builder with all the elements added, when accessing the final visualization, some parts that may look off or overlapping will work correctly in synchronization when the SCADA is running.

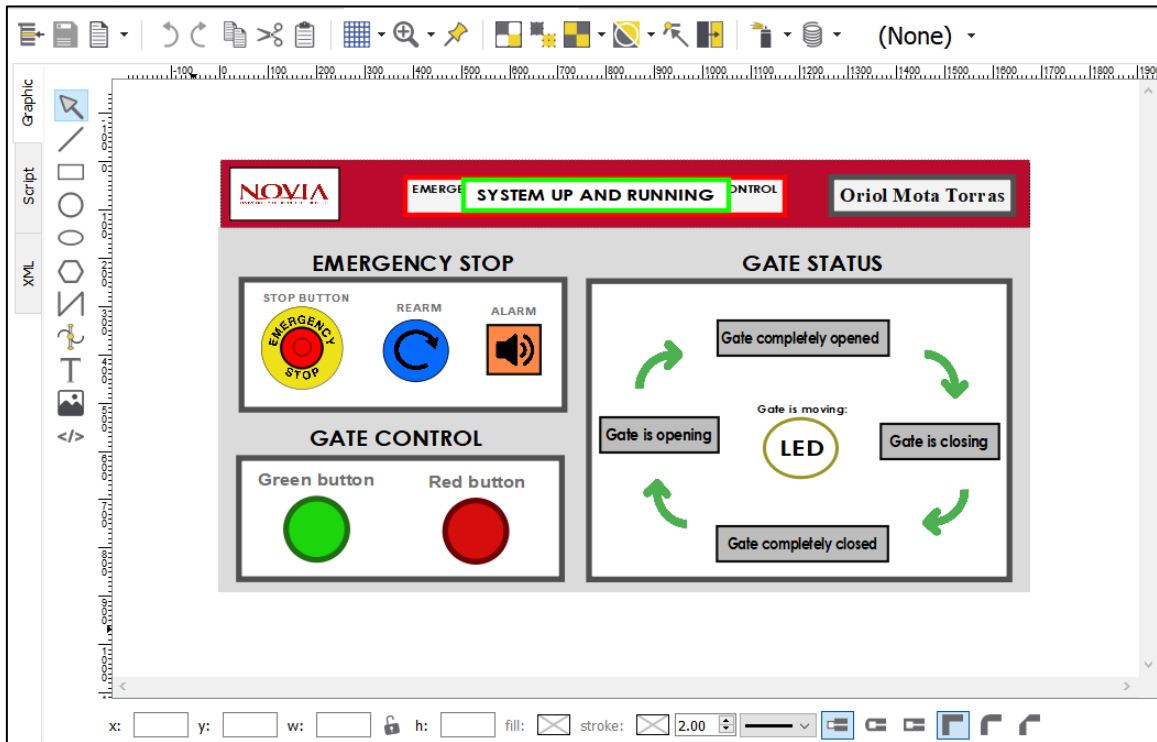


Figure 71: Final elements visualization in Atvise Builder. (Source: Author's own)

Before starting with the component-by-component explanation of SCADA, it is considered necessary to show *Figure 72*. It shows 4 orderly phases of the SCADA graphical creation process. In order to create the visualization shown, mainly rectangles, circles, image insertion and floating text have been used. Once these four tools have been mastered by means of resizing and color changes, it is possible to replicate the displayed SCADA on a graphical level.

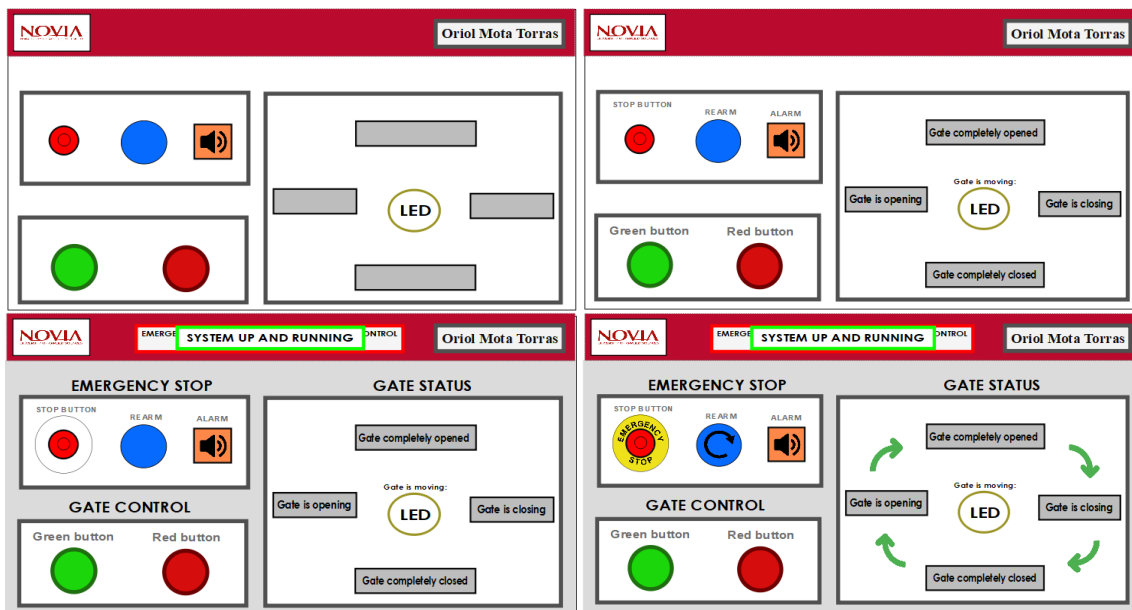




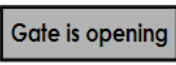
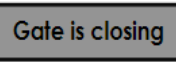




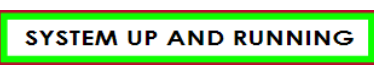
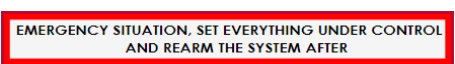


Figure 72: SCADA design evolution in 4 steps. (Source: Author's own)

To link these elements to the variables, use has been made of the "Add simple dynamic", this function adds animations and the ability to act on the variables to the SCADA elements. This function, as explained in the section of this thesis, offers a wide range of possibilities for the SCADA elements to influence or be influenced by the system variables, as well as to add animation, image, and sound changes.

In order to sort the different elements, present on the screen and to know what their function is, *Table 2* has been created. In which, all the SCADA elements can be identified, their name, what they are, the dynamic functions they have, and with respect to which variable they act. In this way it is possible to see all the elements in an orderly way and not to forget any component.

Table 2: SCADA elements configuration table.

| Name | SCADA element | Function | Dynamic | Variable associated |
|---------------------------|---|---|--|---------------------|
| Remote green button |  | Activates door movement | <ul style="list-style-type: none"> Color change when value is 1 Change variable value to 1 | Q6_remGRbtn |
| Remote red button |  | Stops door movement | <ul style="list-style-type: none"> Color change when value is 1 Change variable value to 1 | Q7_remRDbtn |
| Full opened indicator |  | Turns yellow if door is opened | <ul style="list-style-type: none"> Color change when value is 1 | I3_DRopn |
| Full closed indicator |  | Turns yellow if door is closed | <ul style="list-style-type: none"> Color change when value is 1 | I4_DRcls |
| Opening indicator |  | Turns yellow if door is opening | <ul style="list-style-type: none"> Color change when value is 1 | Q5_STATE |
| Closing indicator |  | Turns yellow if door is closing | <ul style="list-style-type: none"> Color change when value is 1 | Q5_STATE |
| LED indicator |  | Turns yellow if the LED is on | <ul style="list-style-type: none"> Color change when value is 1 | Q3_LED |
| Sound alarm indicator |  | Turns orange if the sound alarm is ongoing | <ul style="list-style-type: none"> Color change when value is 1 | Q4_ALARM |
| Emergency stop button |  | Activates the emergency state | <ul style="list-style-type: none"> Color change when value is 1 Change variable value to 1 | BTN_ALARM |
| Restart button |  | Rearms the system after the emergency state | <ul style="list-style-type: none"> Color change when value is 1 Change variable value to 1 | REARM |
| Normal function indicator |  | Is displayed if there is no emergency state | <ul style="list-style-type: none"> Visible when value is 0 | EMERGENCY |
| Emergency state indicator |  | Blinks and is displayed in case of an emergency state | <ul style="list-style-type: none"> Visible when value is 1 Blink while value is 1 | EMERGENCY |

(Source: Author's own)

Next, the configuration made for each of the elements in *Table 2* above will be explained. In particular, the dynamics used for each element and the reasoning behind each one will be described.

Firstly, the operation of the gate control area shown in *Figure 73* is explained, in this section are located the red and green buttons that will control the movement of the gate.

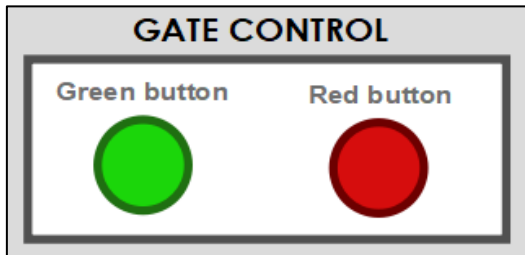


Figure 73: Gate Control section of the final SCADA. (Source: Author's own)

Regarding to the green button, two dynamics have been realised, the exact configurations that have been implemented can be seen in *Figure 74* and *Figure 75*. The first dynamic shown consists of changing the variable when the button is pressed. To configure this dynamic, the event that will trigger the dynamics must be the cause of the mouse cursor, for this reason in the "event" section the option "mouse" is ticked and "Up" and "Down" has been selected.

In the result part, select the variable on which to act, in this case the green remote button (Q6_remGRbtn). To select variables from the list of atvise objects, click on the 3 dots at the top of the variable finder and click on "Objects on My Server", then all added variables will be available. Finally, in the "Action" part, the type of action to be performed is defined, in this case a change of the node value from 0 to 1 or from 1 to 0 depending on its current value. As the linked variable is Boolean type, the discrete condition option must be selected.

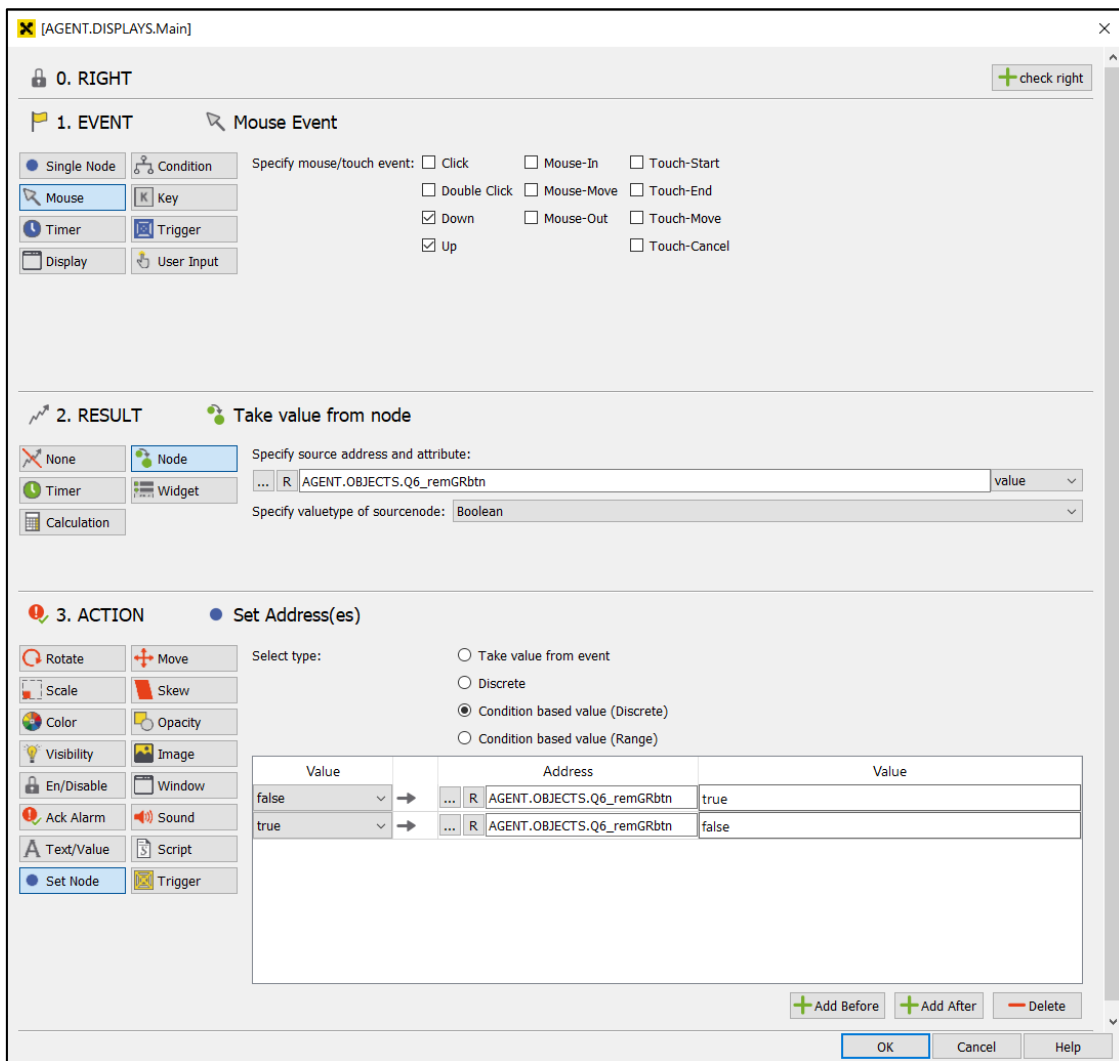


Figure 74: Green button SCADA dynamic n1. (Source: Author's own)

As an explanation of the configuration made in *Figure 74* in the "event" part. Depending on the type of button desired, the mouse event must be configured differently. In case of configuring a push button that returns 1 while it is held (the case of the green button) the actions that trigger the event must be "Up" and "Down". In this way, when the user starts clicking the dynamic will happen once, changing value from 0 to 1, then, when the click is released, the dynamic will be triggered again changing the value from 1 to 0.

As will be explained later for the emergency stop button, if it is intended to configure the button to act as a switch, the mouse action in the "event" section must be "click". This way after clicking once the button will show a 1 until it is not clicked a second time, thus acting as a latching button.

The second dynamic with which the green SCADA button is configured is the colour change function. As can be seen in *Figure 75*, in this case the event that will trigger the dynamic is the change of variable of a node (the remote green button node Q6_remGRbtn). The result part does not need to be configured, as no node change is created, only the object itself is affected.

Finally, in the action part, it can be seen how each of the two possible values of the button is assigned a different fill colour. This way when the button is pressed it becomes darker, giving an animation that makes it clear to the user when the button is being pressed. As the linked variable is Boolean type, the discrete condition option must be selected.

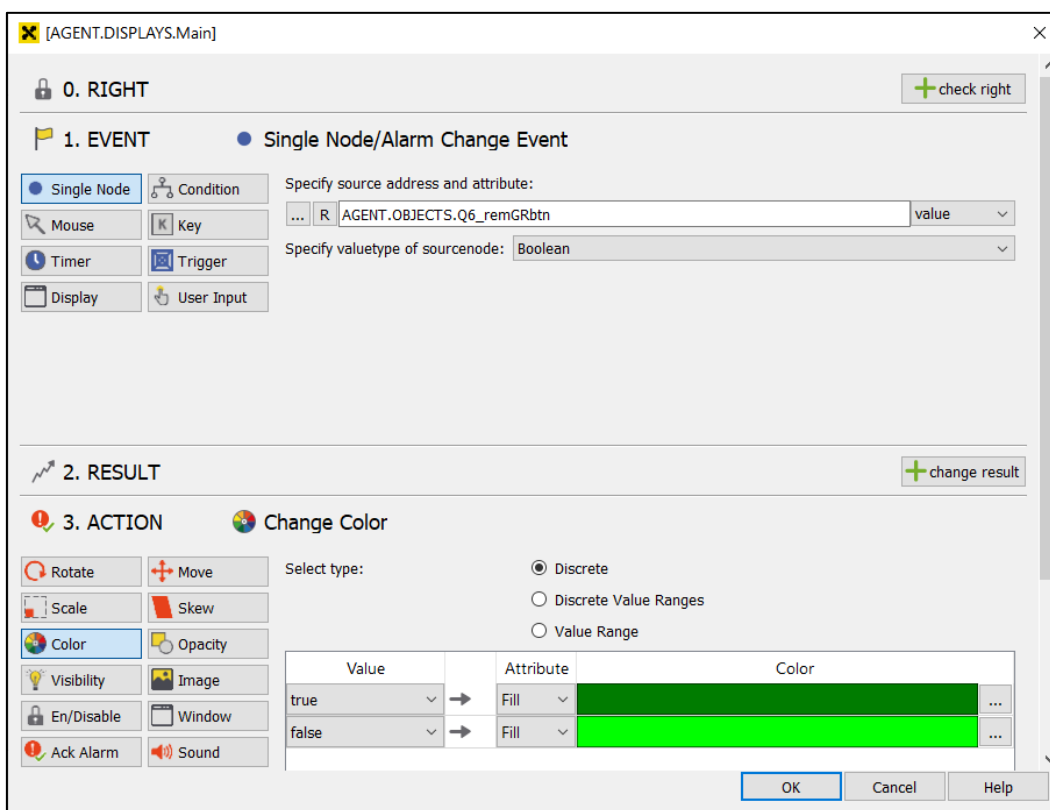


Figure 75: Green button SCADA dynamic n2. (Source: Author's own)

Regarding to the red button configuration, the configuration is the same as for the green button, with the difference that the variable to be acted upon is the remote red button variable (Q7_remRDbtn). For this reason, it is not considered necessary to add representative figures to show its configuration and it is taken as understood.

Next, it is explained how the "Gate status" zone shown in *Figure 76* has been created and configured. To connect the above 4 indicators, arrow image files have been used, which convey to the user what the next stage in the flow of operation will be.

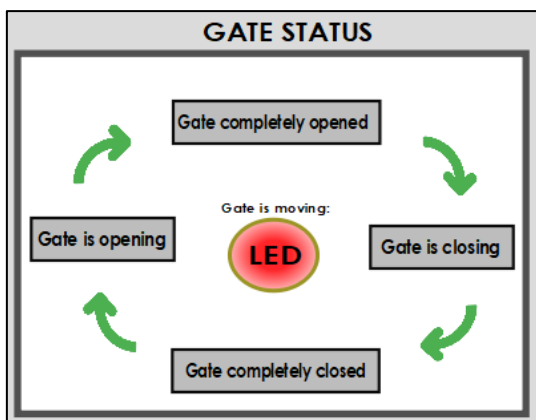


Figure 76: Gate Status section of the final SCADA. (Source: Author's own)

For the "Gate completely open" and "Gate completely closed" indicators, only one colour change dynamic has been performed, the configuration has been the same as for the green and red buttons. The difference is that, in this case, when the value of the variable "I3_DRopn" is zero, the fill colour will be grey, when its 1, the fill colour will be yellow, same as the closing indicator with "I4_DRcls". This gives the user the sensation that they are looking at a luminous panel. The dynamics must be applied to the rectangle behind the letters.

Regarding the other two gate status indicators, the "Gate is opening" and "Gate is closing" indicators, it is necessary to remember the system that has been created to operate the PLC. As explained in the PLC programming section, a variable Q5_STATE which defines whether the gate is closing or opening has been created, this, avoids errors of both motors being activated at the same time.

When this variable is set to 0, the gate is opening, when it is set to one, the gate is closing. Because of this, to make the SCADA indicators it is needed to create colour change animations that have opposite colour fills. The "opening" indicator will be yellow when Q5 is zero and grey when it is 1. The closing indicator will act in the opposite way, being grey when Q5 is 0 and yellow when Q5 is 1. Both dynamics are referenced to the variable "Q5_STATE".

As for the LED, a dynamic colour change has also been implemented, so that when the element is ON, the user can see it in the SCADA. In the case of the LED it will be coloured red. The LED is always active when the gate is moving, for this reason, it has been placed in the "GATE STATUS" section.

Finally, the dynamics used for the SCADA emergency system will be explained, the section associated with this function in the final visualisation can be seen in *Figure 77*. A central sign associated with the emergency system is also shown in the SCADA. It indicates whether the system is operating normally or whether there is an emergency.

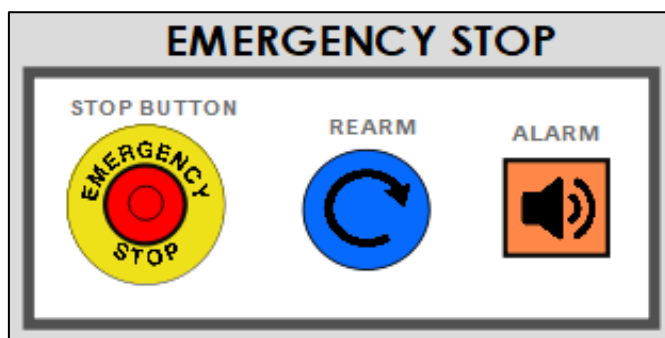


Figure 77: Emergency Stop section of the final SCADA. (Source: Author's own)

The normal operation of the emergency system works as follows: When the user detects a dangerous situation, he/she presses the emergency button, this button works in such a way that until the user does not lift it again, it will keep returning a 1. At the moment the emergency stop button is activated and until it is lifted, the audible alarm will be sounded in order to inform the hypothetical operators of the situation. After pressing and therefore lifting the emergency stop button, the alarm will stop sounding. But the system will still be in an emergency state. To reset the system and ensure that everything has been restored, press the blue reset button. The system will then return to normal operation. Although there are variations, this emergency system is the one commonly used at the industrial level for any type of machinery.

To configure the red emergency stop button, the same dynamics configuration has been used as for the previous buttons but linking it to (BTN_ALARM). However, in this case, in the value change dynamic, the "Event" that causes the dynamic is "mouse" > "click", in this way the button will remain pressed after clicking it for the first time and will remain activated until it is clicked again.

The blue restart button has been configured in exactly the same way as the red and green buttons, only the associated variable has been "REARM". The alarm indicator functions in the same way as the LED indicator, in case the alarm is ON, the background colour of the alarm indicator will turn orange.

Finally, a system has been created consisting of two information signs whose visibility alternates depending on whether there is an emergency situation or not. The first sign with the green outline is shown in *Figure 78* and is only visible if the system is operating normally. In the event of an emergency, the red outlined sign also shown in *Figure 78* will be flashing, warning the user. For the dynamics of these two informative texts, two blocks of individual elements have been created and a visibility dynamic has been added to them.

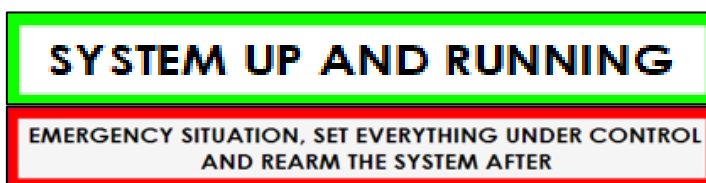


Figure 78: Information State signs of the final SCADA. (Source: Author's own)

Figure 79 below shows the visibility configuration made for the emergency signal highlighted in red. In the same way, as for color changes, the event that triggers the dynamic is the change of the value of a variable, in this case, "EMERGENCY". To make the banner visible, the visibility dynamic in "action" has to be selected, then, check the "Discrete" option and define the value for which the element is required to be visible (true visible and false not visible in this case). In this case, in addition to visible or invisible, it has been added that while visible it flashes every 750ms.

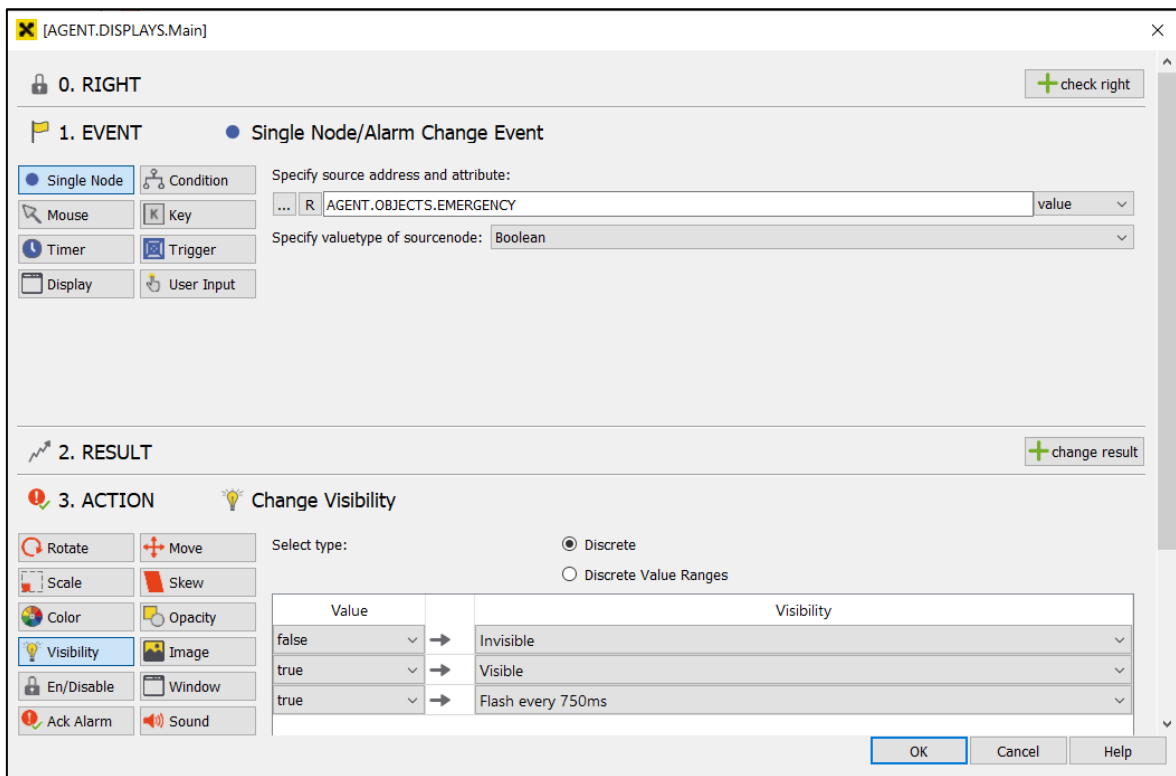


Figure 79: Emergency banner dynamic configuration of the final SCADA. (Source: Author's own)

Next, the same process has been carried out but with the opposite configuration for the sign bordered in green. It is visible when "EMERGENCY" is 0 and invisible when it is 1. Also, the green sign does not have any flashing.

Once all this process is completed, it is possible to execute the SCADA in the PC browser and make it work perfectly. In order to access the SCADA visualization in the computer's Internet browser, go to the upper options bar and select "Guided Actions" > "Open Web Browser for Serve" > "My Server (port)" > "http 80". After that, the default browser of the PC will open automatically, and it will show a visualization equal to the one shown in *Figure 80*. After entering the user "root" and clicking on the padlock in the upper area, it will be possible to operate with the created SCADA.

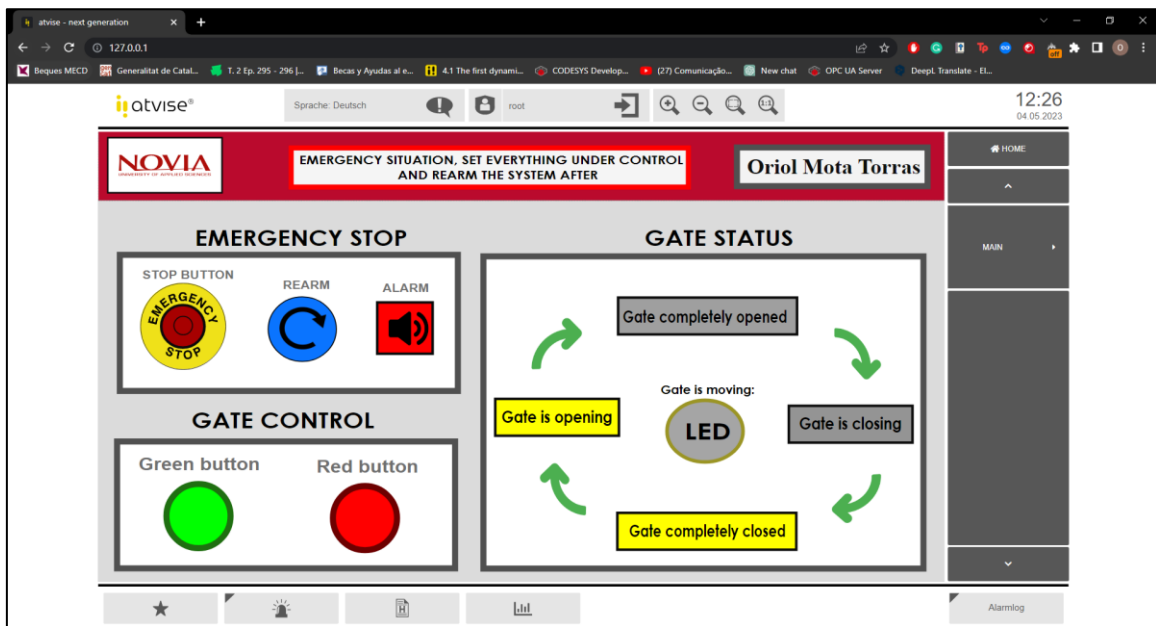


Figure 80: Final SCADA visualization. (Source: Author's own)

This SCADA is perfectly functional and able to control the hardware from our browser located on the PC. Unfortunately, it has not been possible to further develop the complete system and this SCADA could not be exported to the Bachmann HMI. It is considered that this task will belong to future stages of work that can be carried out. Below, in Figure 81, there is an image of the system that has been developed in full operation.

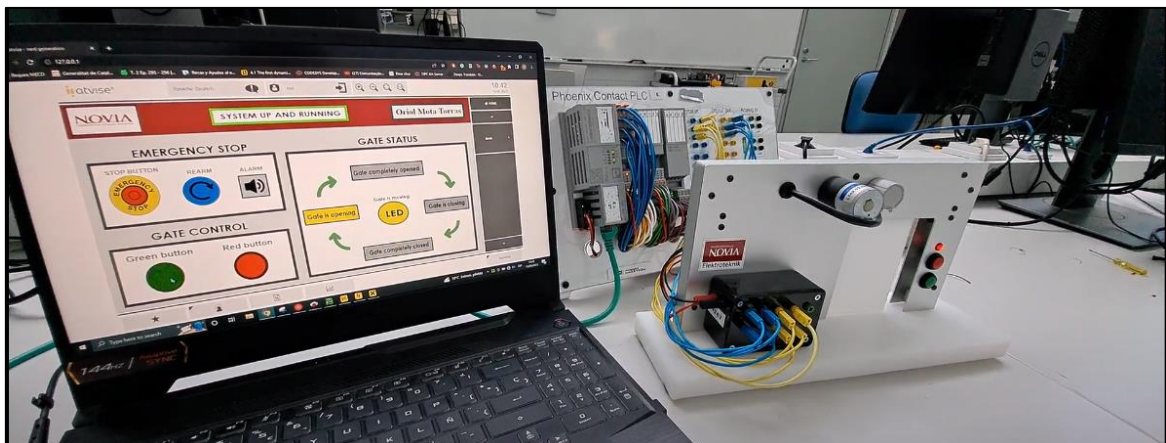


Figure 81: Final SCADA developed. (Source: Author's own)

In order to easily have the resources presented and exposed throughout this and the previous sections, an online directory of public access has been created in which the files

of the different programs used during the SCADA can be found. In this way, opening the files after starting each program will be able to obtain the same result as the one presented in this thesis. In the directory, there are also two videos. One that shows the explained operation of SCADA and another where the process is shown from zero on the computer through which the server is created (using the files in the directory).

The link to the directory is as follows: [click here](#)

9 Results

Over the course of the project, it has been possible to read the development of the different points in reference to the objectives. Next, in this chapter, it is intended to present the different results obtained during the thesis. One factor to take into account in this aspect is that it has not been possible to achieve all the objectives and goals that were set at the beginning of the project. So, this chapter will provide the reader with the exact limit and description of the extent of the work that has been possible to carry out.

The main and absolute objective of this thesis is to create a SCADA system consisting of a PLC, an industrial HMI and a hardware to be controlled. In order to achieve this goal, it has been necessary to split the process into different steps and deliverables.

First of all, one of the basic requirements in any industrial environment is to have all the devices that will be used up to date, fully functional and correctly configured. The configurations and installation carried out in each of the devices are easily located in the first steps shown in sections 4, 5 and 6 of this thesis for the HMI, the PLC and the PC respectively.

Among the various procedures released, the device that required by far the most time and resources to be upgraded correctly was the OT1210W industrial HMI. As explained in section 4 of this report, the update process is long and complex due to it being a slow and repetitive process. This process required the main use of the remote screen access configuration system (TSSW), explained in this thesis. Also, it required the use of the update and internal server options located on the HMIs. Regarding the HMIs updating process, a total of 6 Bachmann OT1210W industrial displays have been upgraded to the latest version. To do so, contact was established with the product technical service. Finally, after several interactions, the internal software of the displays was updated to "Linux 1.52", the Atvise version of OT1200 to "Atvise 3.9.1" and the licensing structure of the displays to the latest version, which promises to be compatible with future versions of Atvise for several years to come. Regarding the software version used in the personal computer, the latest version available from Atvise, version 3.9.1, has been chosen.

Regarding the PLCs, the drivers have been updated by using the CodeSys V3.5 S19 software, whose installation is explained in section 5. The controller version chosen for the correct

operation of the PLC is related to the internal version with which the software can work, which is 4.7.0.0. It is for this reason that with the help of the "CodeSys Control for PLCnext SL" library, the PLC drivers have been updated to version "4.7.0.0-b.trunk.39 (plcnext, armhf)". In the case of a different configuration, the PLC cannot operate since an older version of CodeSys would be needed and the commitment of this thesis involves using all the tools updated to the latest version.

The next step of the process has been to determine the location of the OPC UA server in the developed system. Due to the possibility of the different components used to work independently, it has been seen that the OPC UA server can be located in any of the different devices. It could be found in the PLC, in the industrial display, or in the PC (which finally has been used as a client). In order to mark some limits for the work and realisation of this procedure, it has been decided to continue only with the possibility of using the server that can be created in the PLC. This, as will be explained below, is one of the reasons why communication between all the devices could not be fully accomplished.

After make some research on CodeSys V35 S19 software it has been possible to create and install an OPCUA server on the Phoenix Contact AXC F PLC. With the server created and operational, the PLC has been programmed using a ladder diagram (LD). The installed program controls successfully a miniature industrial gate provided by NOVIA University in the desired way. With the server installed and the PLC controlling the hardware properly, an external OPC UA client has been simulated and connected to the server by using the UAexpert software.

With the purpose of establish a communication flow between the program where the SCADA is developed (Atvise Builder) and the UAexpert client, Atvise Connect has been used. This software has made it possible to browse the directories of the OPCUA client and to establish write and read rights on the various variables that control the system. Once the variables were configured in Atvise Connect, it has become possible to transfer this information to Atvise Builder.

After the variables are found in the Atvise Builder project, using the tools offered by the software, a functional SCADA capable of controlling the opening and closing actions of the industrial gate has been designed. This SCADA also incorporates the possibility of monitoring the current status of the gate (whether it is open, closed, fully closed or fully

open). Moreover, an extra function, which is not possible to simulate physically on the gate due to the lack of buttons, has been added to the SCADA. The function consists of the implementation of a classic emergency stop of any industrial machinery. The operation of this emergency stop and the other components of the SCADA is described in section 7 of this thesis.

It is necessary to explain that the information flow could only be established between the PLC server, the client and the SCADA simulation software in the PC. It has not been possible to export the created SCADA to the Bachmann industrial screen. However, as explained in section 6.4, it was possible to visualise a SCADA on the screen by using the HMI internal atvise server, but the elements of that visualization could not be linked to the PLC server in any way. This is because the HMI internal server does not have the capacity to detect either the PLC server or the Atvise Connect program (in charge of identifying the external information sources in Atvise SCADA). Only the localhost Atvise PC server is able to detect Atvise Connect, the HMI server, although it can be modified via the PC, is not able to detect it.

Apart from trying to communicate with the HMI through the internal server that it has, it has also been tried to operate with the HMI by establishing a direct connection through the PLC server and the HMI. As any result has been obtained, attempts have been made to establish a direct connection between the local atvise server and the screen, also without any result.

In order to easily have the resources presented and exposed throughout this and the previous sections, an online directory of public access has been created in which the files of the different programs used during the SCADA can be found. In this way, opening the files after starting each program will be able to obtain the same result as the one presented in this thesis. In the directory there are also two videos. One that shows the explained operation of SCADA and another where the process is shown from zero on the computer through which the server is created (using the files in the directory).

The link to the directory is as follows: [click here](#)

In general terms, it has been possible to develop a functional SCADA system with an OPC UA server located in a Phoenix Contact AXC F PLC. The system is able to control the behaviour of a miniature industrial gate through a dynamic interface located in the user's PC browser. The final communication flow established was "Industrial Gate" > "PLC OPCUA server" > "OPCUA UAexpert client" > "Atvise Connect" > "Atvise Builder SCADA" > "Web visualisation in localhost".

10 Discussion

With respect to the initial aims and objectives

In order to achieve the aims of this project, a total of 6 objectives have been defined. The first of these objectives consisted of updating all the equipment available in the laboratory. This objective is set because one of the goals of the thesis is that future students of NOVIA University can have a functional and trouble-free material to perform an exercise similar to this thesis, by using the HMI and the PLC. With regard to this first objective, it was possible to successfully upgrade the 6 Bachmann HMIs to atvise 3.9.1 and Linux 1.52 and the 8 PLCs acquired by NOVIA University to 3.7.0.0 drivers.

With regard to the objectives of creating a server and designing a SCADA capable of controlling the server, as well as the chosen hardware, both have been adequately met. The final system presented has a fully functional SCADA that controls the variables of an OPC UA server located in the Phoenix Contact PLC.

Although most of the objectives have been successfully met, not all of them have been achieved. The objective of establishing an information flow between the hardware, PLC, PC and HMI has not been possible. The problem in this process has been the impossibility to export the SCADA created in the PLC to the HMI, despite being able to access the screen and the internal software. It has been possible to simulate a SCADA in the HMI, but the variables on which it operates only exist at the internal level of the display and are not connected to any communication network.

One of the reasons why it has not been possible to establish communication between all the devices could be because a system like the one intended to develop should only be operated with a main server. However, Atvise software works by creating its own server on each of their devices, this causes having the PLC server, the PC software Atvise server and the HMI internal Atvise server, making communication between the three stations much more difficult.

Regarding the objective of creating a manual, I consider that section 7 of this thesis, where all the steps followed until the final product are explained, adequately meets the objective. However, the manual will be complete until progress has been made, not being able to establish communication between the HMI and the PC.

The last of the objectives consisted of carrying out a class activity for future students, due to lack of time and having focused efforts on completing the previous objectives, it has not been possible to develop this point. An appropriate activity to carry out would be to follow the steps of the manual and understand what is done in each one.

With respect to other published work

If we refer to works developed previously by other authors, it is necessary to mention the lack of academic thesis that could have been helpful for this topic. Although it is true that there are similar studies, the process that has been done in this case has been done using specific software and products that have a certain configuration.

In other thesis and academic articles, devices used are much easier to configure and communicate with each other are used. The most common case is to use all the equipment of the same brand, normally Siemens.

With respect to the limitations of the work

The main limitation in developing this project has been the lack of available resources. On the internet, either in forums or in academic articles, there is practically no information on the aspects worked on. The biggest sources of information have been the tutorials and official pages of each of the software, the problem with these aids is that since they are private companies that seek to do business by installing and configuring their products, many times the tutorials were not really useful. Other times they were incomplete, so in this way the customer must hire the services of the company.

When communicating products of the same brand, a single software is already trained to carry out the entire process and has the ability to automatically detect the own company's products. However, trying to communicate such different and independent products has required a much more sophisticated configuration than in the usual cases. In addition, it has also been necessary to use 4 different software to be able to communicate only the PLC with the SCADA.

The last of the limitations has been the lack of help from the stakeholder. The technical service in charge of helping with the Atvise software has taken up to 15 days to answer. In addition, many times they have stopped answering for forcing another user to be created

Also, they have even refused to give free 1-month licenses to be able to continue with the work. In the same way, the contact had no idea what to do to complete the process, he limited himself to empty explanations not related to the process to be carried out and to answer the same thing that the technical service told him.

With respect to consistencies and inconsistencies

Because the document has been drafted with the objective of establishing a PLC - PC - HMI flow and this has not been fully achieved, some inconsistencies in the documentation may be found in some sections. Also, due to continuous improvement and the slow process of upgrading the equipment provided, it may be that some of the screenshots in certain sections are from an earlier version of the software than the final one, however, this does not affect the explanation and understanding of the process in any way.

One of the main problems encountered during the development of the thesis has been the issue of atvise licence management. In order to use atvise, free one-month trial accounts have been created, each time with a different e-mail address. As the e-mail had to be validated by an employee and the company had to be kept in contact for the process of updating the displays, that process was much more difficult. Finally, the technical service has suggested to the university that the next time a similar project is carried out, they should purchase the educational licence packs available.

11 Conclusions

One of the various jobs that an industrial engineer has to deal with is the assembly and development of SCADA systems. In these systems, there are often devices from different manufacturers which need a high level of configuration to be able to communicate with each other. This thesis focuses on establishing communication between the Bachmann industrial HMI model OT1210W and an OPC UA server located in a PLC model AXC F from Phoenix Contact.

After starting the process, it was seen that the hardware was outdated and did not work as desired or did not work directly. In order to make all the devices work properly, it has been necessary to update the industrial HMIs and the PLC drivers. Bachmann's HMIs have been updated to the latest Linux 1.52 software version, their internal version of Atvise has been updated to the latest version 3.9.1 and the internal licensing structure has been changed to the latest available update. The PLC drivers have been updated by CodeSys V35 SP19 to the latest existing version, 3.7.0.0. In addition, the latest software available of Atvise, CodeSys and UAexpert has been used on the personal computer.

Finally, after working to establish the desired communication flow, it has not been possible to communicate all the devices as originally intended. The SCADA system that has been developed communicates the PLC server with a SCADA visualized on the user's computer internet browser, however, it has not been possible to implement this SCADA on the Bachmann industrial HMI. The system that has been created monitorize a miniature industrial gate provided by NOVIA University as mechanical hardware. This door is controlled by the Phoenix Contact PLC on which an OPC UA server is installed using CodeSys software. Next, the information from this server is transmitted to the user's PC simulating an OPC UA client by using the UAExpert program. As the final step of the process, using Atvise Builder and Atvise Connect, the writing and reading rights of the client variables have been obtained and a web SCADA has been generated in the internet browser of the user's PC. In this way, it has been possible to control the desired hardware using a SCADA system connected to an OPC UA server.

Regarding the SCADA created, the system allows a real time monitoring of the door status. Moreover, it is possible to control its behaviour and it even has an emergency stop system with rearm, as indicated in the industrial standards.

To sum up, it has been possible to create a SCADA system able to control the OPC UA server. This thesis shows how to configure an automated system made up of industrial components of different brands, and how the need to configure these devices is one of the most complicated tasks for an industrial engineer.

11.1 Future work

Regarding future studies, the continuation of this thesis should be focused on fully completing the initial idea of being able to control the OPCUA server from the industrial display. It would only be necessary to replicate the process carried out until the end but managing to export the SCADA available in the personal computer to the Human Machine Interface. In order to realise this step, that future research should focus on how to link the variables from the OPC UA client to the internal Atvise server of the HMI. By achieving that aspect, the work would be completed, and the project would take on a much more industrial aspect.

Another aspect that can be considered is to try not to require a client located on the PC, as in industrial environments the type of structure present is not possible. An alternative model would be to try to set up a system where the information flow is directly from the PLC to the HMI, making the HMI behave as a client of the OPC UA server.

Other options could have been explored, such as only operating with the HMI internal server. To reduce as much as possible the inconsistencies in the process, we have tried to carry out the whole process in the way it would be done at the industrial level. Placing the server in the PLC and operating the other elements as OPC UA clients.

12 References

- 4.0 Solutions. (2021). *What is OPC? - Part I - What you need to know... [YouTube Video]*. Retrieved from https://www.youtube.com/watch?v=ApxtnOTVbzY&ab_channel=4.0Solutions
- Andrés, D. M. (2021, 5 28). *Youtube. ¿Quieres conectar tu industria? Usa OPC UA*. Retrieved from https://www.youtube.com/watch?v=gJ0vYAPhNcl&ab_channel=DiegoMart%C3%ADndeAndr%C3%A9s
- atvise - HMI + SCADA. (2023). *atvise® product portfolio*. Retrieved from <https://www.atvise.com/en/products/atvise-overview>
- atvise. (2015). *atvise® Aggregate (Englisch)*. Retrieved from https://www.youtube.com/watch?v=LxZZVwQqB3w&list=PLB4piw0c4tc0-qMi8nf8Tsq9cadQuOk0u&index=12&ab_channel=atvise
- atvise. (2015). *YouTube. atvise® Engineering 1 (English)*. Retrieved from https://www.youtube.com/watch?v=n55oQYfEWLI&list=PLB4piw0c4tc0-qMi8nf8Tsq9cadQuOk0u&index=1&ab_channel=atvise
- atvise. (2015). *YouTube. atvise® Engineering 2 (English)*. Retrieved from https://www.youtube.com/watch?v=256zkbwN6Qo&list=PLB4piw0c4tc0-qMi8nf8Tsq9cadQuOk0u&index=2&ab_channel=atvise
- atvise. (2015). *YouTube. atvise® Engineering 3 (English)*. Retrieved from https://www.youtube.com/watch?v=oBhu_fvUZyo&list=PLB4piw0c4tc0-qMi8nf8Tsq9cadQuOk0u&index=3&ab_channel=atvise
- atvise. (2015). *YouTube. atvise® live (English)*. Retrieved from https://www.youtube.com/watch?v=mGdVLw7JrKg&list=PLB4piw0c4tc0-qMi8nf8Tsq9cadQuOk0u&index=11&ab_channel=atvise
- atvise. (2015). *YouTube. atvise® ObjectDisplay - Custom Bar Graph (English)*. Retrieved from https://www.youtube.com/watch?v=B2L2-dfsa00&list=PLB4piw0c4tc0-qMi8nf8Tsq9cadQuOk0u&index=6&ab_channel=atvise
- atvise. (2015). *YouTube. atvise® ObjectDisplay - Custom Gauge (English)*. Retrieved from https://www.youtube.com/watch?v=folh4P0nj0s&list=PLB4piw0c4tc0-qMi8nf8Tsq9cadQuOk0u&index=7&ab_channel=atvise
- atvise. (2015). *YouTube. atvise® User and Rights Management (English)*. Retrieved from https://www.youtube.com/watch?v=q12isb7ay4s&list=PLB4piw0c4tc0-qMi8nf8Tsq9cadQuOk0u&index=5&ab_channel=atvise
- atvise. (2015). *YouTube. atvise® Vertical Engineering (English)*. Retrieved from https://www.youtube.com/watch?v=-DWilqNFgd8&list=PLB4piw0c4tc0-qMi8nf8Tsq9cadQuOk0u&index=4&ab_channel=atvise
- atvise. (2017). *Youtube. atvise® Builder Login (Englisch)*. Retrieved from https://www.youtube.com/watch?v=Cdfvu-7mzPk&list=PLB4piw0c4tc0-qMi8nf8Tsq9cadQuOk0u&index=13&ab_channel=atvise

- atvise. (2020). *YouTube. atvise® connect - Installation, Licensing & First Steps (English)*. Retrieved from https://www.youtube.com/watch?v=IQ-yEMdvw1U&list=PLB4piw0c4tc0-qMi8nf8Tsq9cadQuOk0u&index=14&ab_channel=atvise
- atvise. (2020). *Youtube. atvise® Data Aquisition (English)*. Retrieved from https://www.youtube.com/watch?v=bCdZCV2BXnU&ab_channel=atvise
- atvise. (2021). *Youtube. atvise® Product Licensing (English)*. Retrieved from https://www.youtube.com/watch?v=hUTXMOEqJ1w&t=100s&ab_channel=atvise
- atvise. (2021). *YouTube. atvise® Product Licensing (English)*. Retrieved from https://www.youtube.com/watch?v=hUTXMOEqJ1w&list=PLB4piw0c4tc0-qMi8nf8Tsq9cadQuOk0u&index=15&ab_channel=atvise
- atvise. (2023). *Youtube. First steps in atvise*. Retrieved from https://www.youtube.com/watch?v=u4SsQlI4gLc&ab_channel=atvise
- atvise Youtube. (2015). *Youtube. atvise® Video Tutorials (English)*. Retrieved from https://www.youtube.com/watch?v=n55oQYfEWLI&list=PLB4piw0c4tc0-qMi8nf8Tsq9cadQuOk0u&ab_channel=atvise
- AtviseCustomerService. (2023). License issues. (O. Mota, Interviewer)
- AVEVA. (2021). *What is SCADA - Making Use of Operational Data*. Retrieved from <https://www.aveva.com/es-es/solutions/operations/scada/#:~:text=SCADA%20es%20una%20combinaci%C3%B3n%20de,controlar%20y%20optimizar%20dicho%20proceso.>
- Bachmann electronic GmbH. (2023). *Operator Terminal System Software*. Retrieved from <https://www.bachmann.info/en/products/operator-terminal-systemsoftware>
- Bachmann Visutec GmbH. (2019). *Release Notes Atvise® 3.4. Atvise.com*. Retrieved from <https://customer.atvise.com/en/component/phocadownload/category/108-atvise-3-4>
- Bachmann Visutec GmbH. (2021, February 28). *User manual System software Linux V1.52. Document version number 9*. Retrieved from <https://customer.atvise.com/en/customer-area-downloads-en/category/120-atvise-3-9>
- Bachmann Visutec GmbH. (2023). *Atvise. Reference manual Atvise Builder.* . Retrieved from <https://customer.atvise.com/en/customer-area-downloads-en/category/120-atvise-3-9>
- Bachmann Visutec GmbH. (2023). *atvise® connect. High-performance data acquisition for industrial applications*. Retrieved from <https://www.atvise.com/en/products/expansions/atvise-connect>
- CODESYS. (2023). *CODESYS Development System V3*. Retrieved from CodeSys website: https://store.codesys.com/engineering/codesys.html?__store=en

- CodeSys Group. (2022). *CODESYS Communication OPC UA OPC UA Server*. Retrieved from CodeSys Online Help: https://content.helpme-codesys.com/en/CODESYS%20Communication/_cds_runtime_opc_ua_server.html
- CodeSys Help. (2023). *CodeSys Online help*. Retrieved from CodeSys 3.5.17.0 official manual: <https://help.codesys.com/>
- Electrónica Edimar. (2020, April 14). *Programación en PLC: controladores programables / Edimar*. Retrieved from <https://edimar.com/programacion-en-plc/#:~:text=No%20obstante%2C%20sigue%20siendo%20un,m%C3%A1quina%20o%20proceso%20que%20controlan.>
- Harb Mecatrónica. (2020). *¿Which is the difference between SCADA y HMI? [YouTube Video]*. Retrieved from https://www.youtube.com/watch?v=BNwmOGLt9jc&t=253s&ab_channel=HarbMecatr%C3%B3nica
- Inductive Automation. (2020). *What is a PLC?* Retrieved from <https://www.inductiveautomation.com/resources/article/what-is-a-PLC#:~:text=PLC%20stands%20for%20Programmable%20Logic,in%20size%20and%20form%20factors.>
- Martín, D. (2021). *¿Quieres conectar tu industria? Usa OPC UA [YouTube Video]*. Retrieved from https://www.youtube.com/watch?v=gJ0vYAPhNcl&ab_channel=DiegoMart%C3%ADndeAndr%C3%A9s
- OPC Foundation. (2015). *Members - OPC Foundation. The Industrial Interoperability Standard™*. Retrieved from <https://opcfoundation.org/members>
- OPC Foundation. (2019, September 26). *Unified Architecture - OPC technologies - OPC UA*. Retrieved from <https://opcfoundation.org/about/opc-technologies/opc-ua/>
- OPC Foundation. (2022). *History of OPC Foundation*. Retrieved from OPC Foundation website: <https://opcfoundation.org/about/opc-foundation/history/>
- OPCconnect. (2023). *History of OPC*. Retrieved from OPCconnect website: <https://www.opcconnect.com/history.php>
- Phoenix Contact. (2013). *AXC F 2152 - Controller 2404267*. Retrieved from Phoenix Contact website: <https://www.phoenixcontact.com/fi-fi/tuotteet/controller-axc-f-2152-2404267>
- SCADA international. (2023). *SCADA intrenation website*. Retrieved from What is SCADA? SCADA systems explained: https://scada-international.com/what-is-scada/?utm_medium=cpc&utm_source=google.com&utm_campaign=scada-explained&gad=1&gclid=Cj0KCQjwslejBhDOARIsANYqkD02-dhLKvusnnrSel3iRGeHTEcAXbJ89jMck9F3NdudElr1O7f-_YEaAnU7EALw_wcB
- Software AG. (2022, February 11). *OPC UA integration - Software AG*. Retrieved from [https://www.softwareag.com/en_corporate/resources/iot/article/opc-ua.html#:~:text=OPC%20Unified%20Architecture%20\(OPC%20UA\)%20is%20a%20machine%2Dto,specifications%20into%20an%20extensible%20framework](https://www.softwareag.com/en_corporate/resources/iot/article/opc-ua.html#:~:text=OPC%20Unified%20Architecture%20(OPC%20UA)%20is%20a%20machine%2Dto,specifications%20into%20an%20extensible%20framework)

Technopedia. (2011, December 16). *OLE for Process Control (OPC)*. Retrieved from <https://www.techopedia.com/definition/1307/ole-for-process-control-opc>

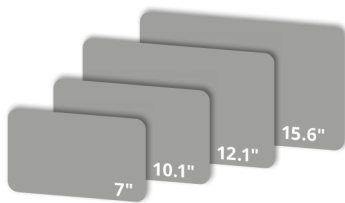
TESLA measurement control systems environmental technologies. (2023). *Atvise SCADA Bachmann - TESLA MEASURING CONTROL SYSTEMS | DATALOGGER | HUMIDITY TEMPERATURE SENSORS | GAS SENSORS | DATA COLLECTION SYSTEMS*. Retrieved from <https://www.teslakontrol.com/en/atvise-scada-bachmann>

Vester Business. (2021, 11 24). *SCADA Software for Industry 4.0 - The first HMI SCADA software | atvise®*. Retrieved from atvise® - Sistema SCADA Web: <https://atvise.vesterbusiness.com/en/>

Vester Business. (2022). *Youtube. Cómo configurar el cliente MQTT de atvise® Connect*. Retrieved from [https://www.youtube.com/watch?v=625GOKVmx1k&t=630s&ab_channel=Vester Business](https://www.youtube.com/watch?v=625GOKVmx1k&t=630s&ab_channel=Vester+Business)

Appendix 1: OT1200 Bachmann Web Terminal datasheet

Visualization and Operating / Operating and Monitoring Devices / Operator Terminals



OT1200 Web Terminal

The specifications and interfaces of the OT1200 web terminal make it the ideal choice for small and medium web visualizations, such as atvise®.

Brilliant displays with high brightness, strong colors, very good viewing angles and a long service life make a lasting impression. The modern widescreen diagonals (7", 10.1", 12.1", 15.6") offer high resolutions, e.g. 15.6" with Full-HD. The display lifetime (half brightness) of over 50,000 hours lies within the premium segment.

User input via projected-capacitive (PCAP) multi-touchscreens allows the use of multi-touch-gestures (e.g. pinch-to-zoom, swipe, two-hand operation). The visually appealing, hardened glass surface is easy to clean. The exclusive use of industrial components guarantees a long service lifespan and availability. Permanently soldered RAM and eMMC memories, as well as the absence of any moving parts, ensure maximum resistance to shock and vibration. High temperature ranges of up to 60 °C allow operation under harsh environments.

The pre-installed browser is designed for industrial applications. An integrated onscreen-keyboard enables convenient user input and reduces development efforts. Unwanted, automatic updates are prohibited, ensuring that only released software is used in the field, reducing downtime to a minimum.

The Linux system (based on Debian) provides an easy to use, web-based configuration tool. Configuration settings can be changed directly on the device or remote via browser. The system software includes further useful services, e.g. SSH, NTP, VNC, as well as a Web-API for secure hardware access. The API can be used to react to hardware keys, control LEDs and display brightness, and much more.

We will gladly supply your required software configuration direct from the factory: unpack, connect, done.

Features:

- Pre-installed and easy to configure web-browser
- Gesture control with PCAP Multi-Touch
- Ideal for all kinds of web-visualization
- Visualize data from OPC-UA sources in combination with atvise®
- Brilliant display with high resolutions in 7", 10.1", 12.1" and 15.6" widescreen
- Cortex®-A9 Dual-Core CPU, 2x ETH 10/100, 2x USB, CFast-Slot, 4 GB eMMC onboard
- Robust design and extended temperature range up to +60 °C

| OT1200 Series | OT1207 |
|-----------------------------------|---|
| Display | |
| Diagonals | 7" |
| Resolution | 800x480 |
| Brightness | 450 cd/m ² |
| Contrast | 800:1 |
| Viewing angle (L, R, U, D) | 89/89/89/89 ° |
| Life expectancy (half brightness) | 50,000 h |
| Processor/RAM | |
| Processor | ARM® Cortex®-A9 i.MX6Dual |
| RAM | 1 GB DDR3 RAM |
| Mass Storage | |
| Mass storage integrated | 4 GB eMMC ¹⁾ |
| Control/Display Elements | |
| Touch screen type | Analog resistive touch |
| Design | Custom marking on request |
| LEDs and control keys | Custom design on request |
| Interfaces | |
| Ethernet 10/100 MBit | 1x RJ45 |
| USB | 2x USB 2.0 |
| Power Supply | |
| Power supply | 24 V DC (18 to 36 V) |
| Mating plug | MC1, 5/3-STF-3,5 (2-pol.) |
| Reverse polarity protection | Yes |
| Galvanic isolation | No |
| Electrical Safety | |
| Protection class (DIN EN 61140) | III |
| Degree of protection (IEC 60529) | IP65 in front IP20 at rear |
| Ambient Conditions | |
| Operating temperature | 0 to +60 °C fanless |
| Rel. humidity operation | 5 to 95% without condensation |
| Storage temperature | -20 to +70 °C |
| Rel. humidity storage | 5 to 95% without condensation |
| Installation altitude | 2,000 to 4,500 m above sea level: Temperature reduction of 0.5 Kelvin (K) per 100 m height |
| Pollution degree (IEC 60664-1) | 2 |
| Approvals / Certificates | |
| Product safety | CE, cULus, CCC |

1) At delivery, approx. 1.2 GB occupied by system software

| OT1200 Series | | OT1207 |
|------------------------|--|--------|
| Software | | |
| Operating system | Linux Embedded (Debian) | |
| Visualization software | Industrial Browser Visualization server with atvise® available | |
| Other Software | Bachmann Terminal Setup Software (TSSW) SSH Server VNC Server NTP Client Firewall Screen saver ... and many more useful software packages. Additional software packages can be installed subsequently. | |

| Order Code | | |
|---|-------------|---|
| Item-No. | Item | Description |
| OT1207W/BE1/MX6/ 1G/EMC/04/LX | 00025995-00 | Operator Terminal: 7" (800x480); Resistive Touch; Freescale ARM® Cortex®-A9 i.MX6Dual; 1 GB DDR3 RAM; 1x Eth 10/100; 2x USB2.0; 4GB eMMC onboard; OS: Linux Embedded; Operating Temp. 0 .. +60 °C |
| OT1207W/BE1/MX6/ 1G/EMC04/LX/atvise® | 00033189-00 | Operator Terminal: 7" (800x480); Resistive Touch; Freescale ARM® Cortex®-A9 i.MX6Dual; 1 GB DDR3 RAM; 1x Eth 10/100; 2x USB2.0; 4GB eMMC onboard; OS: Linux Embedded inkl. atvise® hmi; Operating Temp. 0 .. +60 °C |

Visualization and Operating / Operating and Monitoring Devices / Operator Terminals

| OT1200+ Series | OT1207+ | OT1210+ | OT1212+ | OT1215+ |
|-----------------------------------|---|-----------------------------------|-----------------------|-----------------------|
| Display | | | | |
| Diagonals | 7" | 10.1" | 12.1" | 15.6" |
| Resolution | 800x480 | 1280x800 | 1280x800 | 1920x1080 |
| Brightness | 450 cd/m ² | 400 cd/m ² | 400 cd/m ² | 400 cd/m ² |
| Contrast | 800:1 | 800:1 | 1.000:1 | 1500:1 |
| Viewing angle (L, R, U, D) | 89/89/89/89 ° | 85/85/85/85 ° | 89/89/89/89 ° | 85/85/85/85 ° |
| Life expectancy (half brightness) | 50,000 h | 100,000 h | 50,000 h | 70,000 h |
| Processor/RAM | | | | |
| Processor | ARM® Cortex®-A9 i.MX6Dual+ | | | |
| RAM | 2 GB DDR RAM | | | |
| Mass Storage | | | | |
| Mass storage integrated | 4 GB eMMC ¹⁾ | | | |
| Mass storage selectable | Optional CFast (min. 4 GB) ²⁾ | | | |
| Control/Display Elements | | | | |
| Touch screen type | Resistive | Projective capacitive multi touch | | |
| Design | Custom marking on request | | | |
| LEDs and control keys | Custom design on request | | | |
| Status indication | 1x LED for supply voltage (green) 2x LED for status and speed per ETH socket | | | |
| Interfaces | | | | |
| Ethernet | 2x 10/100 (RJ45, separate) | | | |
| USB | 2x USB 2.0 | | | |
| Power Supply | | | | |
| Power supply | 24 V DC (18 to 36 V) | | | |
| Mating plug | MC1, 5/3-STF-3,5 (2-pol.) | | | |
| Reverse polarity protection | Yes | | | |
| Galvanic isolation | No | | | |
| Electrical Safety | | | | |
| Protection class (DIN EN 61140) | III | | | |
| Degree of protection (IEC 60529) | IP65 in front IP20 at rear | | | |

1) At delivery, approx. 1.2 GB occupied by system software

2) Storage medium not included unless stated in the order description

Visualization and Operating / Operating and Monitoring Devices / Operator Terminals

| OT1200+ Series | OT1207+ | OT1210+ | OT1212+ | OT1215+ |
|-----------------------------------|--|---------|---------|---------|
| Ambient Conditions | | | | |
| Operating temperature | 0 to +60 °C fanless | | | |
| Rel. humidity operation | 5 to 95% without condensation | | | |
| Storage temperature | -20 to +70 °C | | | |
| Rel. humidity storage | 5 to 95% without condensation | | | |
| Installation altitude | 2,000 to 4,500 m above sea level: Temperature reduction of 0.5 Kelvin (K) per 100 m height | | | |
| Pollution degree (IEC 60664-1) | 2 | | | |
| Approvals / Certificates | | | | |
| Product safety | CE, UL, cUL, cULus, CCC | | | |
| Software | | | | |
| Operating system | Linux Embedded (Debian) | | | |
| Visualization software | Industrial Browser Visualization server with atvise® available | | | |
| Other Software | Bachmann Terminal Setup Software (TSSW) SSH Server VNC Server NTP Client Firewall Screen saver ... and many more useful software packages. Additional software packages can be installed subsequently. | | | |

Visualization and Operating / Operating and Monitoring Devices / Operator Terminals

| Order Code | | |
|---|-------------|--|
| Item-No. | Item | Description |
| OT1207WM/BE1/ MX6+/2G/EMC04/LX | 00033340-00 | Operator Terminal: 7" (800x480); Resistive Touch; Freescale ARM® Cortex®-A9 i.MX6Dual+; 2 GB DDR3 RAM; 2x Eth 10/100; 2x USB2.0; 4GB eMMC onboard; CFast-Card-Slot; OS: Linux Embedded; Operating Temp. 0 .. +60 °C |
| OT1210WM/BE1/ MX6+/2G/EMC04/LX | 00033341-00 | Operator Terminal: 10.1" (1280x800); Projected-Capacitive Multi-Touch; Freescale ARM® Cortex®-A9 i.MX6Dual+; 2 GB DDR3 RAM; 2x Eth 10/100; 2x USB2.0; 4GB eMMC onboard; CFast-Card-Slot; OS: Linux Embedded; Operating Temp. 0 .. +60 °C |
| OT1212WM/BE1/ MX6+/2G/EMC04/LX | 00033342-00 | Operator Terminal: 12.1" (1280x800); Projected-Capacitive Multi-Touch; Freescale ARM® Cortex®-A9 i.MX6Dual+; 2 GB DDR3 RAM; 2x Eth 10/100; 2x USB2.0; 4GB eMMC onboard; CFast-Card-Slot; OS: Linux Embedded; Operating Temp. 0 .. +60 °C |
| OT1215WM/BE1/ MX6+/2G/EMC04/LX | 00033343-00 | Operator Terminal: 15.6" (1920x1080); Projected-Capacitive Multi-Touch; Freescale ARM® Cortex®-A9 i.MX6Dual+; 2 GB DDR3 RAM; 2x Eth 10/100; 2x USB2.0; 4GB eMMC onboard; CFast-Card-Slot; OS: Linux Embedded; Operating Temp. 0 .. +60 °C |
| OT1207WM/BE1/ MX6+/2G/EMC04/LX/atvise® | 00033344-00 | Operator Terminal: 7" (800x480); Resistive Touch; Freescale ARM® Cortex®-A9 i.MX6Dual+; 2 GB DDR3 RAM; 2x Eth 10/100; 2x USB2.0; 4GB eMMC onboard; CFast-Card-Slot; OS: Linux Embedded incl. atvise® hmi license; Operating Temp. 0 .. +60 °C |
| OT1210WM/BE1/ MX6+/2G/EMC04/LX/atvise® | 00033345-00 | Operator Terminal: 10.1" (1280x800); Projected-Capacitive Multi-Touch; Freescale ARM® Cortex®-A9 i.MX6Dual+; 2 GB DDR3 RAM; 2x Eth 10/100; 2x USB2.0; 4GB eMMC onboard; CFast-Card-Slot; OS: Linux Embedded incl. atvise® hmi license; Operating Temp. 0 .. +60 °C |
| OT1212WM/BE1/ MX6+/2G/EMC04/LX/atvise® | 00033346-00 | Operator Terminal: 12.1" (1280x800); Projected-Capacitive Multi-Touch; Freescale ARM® Cortex®-A9 i.MX6Dual+; 2 GB DDR3 RAM; 2x Eth 10/100; 2x USB2.0; 4GB eMMC onboard; CFast-Card-Slot; OS: Linux Embedded incl. atvise® hmi license; Operating Temp. 0 .. +60 °C |
| OT1215WM/BE1/ MX6+/2G/EMC04/LX/atvise® | 00033347-00 | Operator Terminal: 15.6" (1920x1080); Projected-Capacitive Multi-Touch; Freescale ARM® Cortex®-A9 i.MX6Dual+; 2 GB DDR3 RAM; 2x Eth 10/100; 2x USB2.0; 4GB eMMC onboard; CFast-Card-Slot; OS: Linux Embedded incl. atvise® hmi; Operating Temp. 0 .. +60 °C |

Bachmann electronic GmbH · 09/2021 · Specification subject to change – the product's characteristics are exclusively governed by the data of the respective user manual.

Source: <https://www.bachmann.info/en/products/ot1200>

Appendix 2: Phoenix Contact AXC F 2521 datasheet

Controller - AXC F 2152

2404267

<https://www.phoenixcontact.com/gb/products/2404267>


Please be informed that the data shown in this PDF document is generated from our Online Catalog. Please find the complete data in the user documentation. Our General Terms of Use for Downloads are valid.



PLCnext Control for the direct control of Axioline F I/Os. With two Ethernet interfaces. Complete with connector and bus base module.

Product Description

The PLCnext AXC F 2152 controller for the Axioline I/O system is fast, robust and easy to use. It has been consistently designed for maximum performance, easy handling and use in harsh industrial environments.

Your advantages

- Linux operating system
- Supports high-level languages
- Up to 63 AXIO I/O modules can be mounted side by side
- 2x Ethernet interfaces (integrated switch)
- Increased resistance to EMI
- Extended temperature range of -25 °C ... 60 °C
- PROFINET support
- Connection to PROFICLOUD
- Numerous protocols supported such as: http, https, FTP, OPC UA, SNMP, SMTP, SQL, MySQL, DCP, etc.
- Connection to the PLCnext Store

Commercial Data

| | |
|--------------------------------------|--------------------|
| Item number | 2404267 |
| Packing unit | 1 pc |
| Minimum order quantity | 1 pc |
| Sales Key | DRADAC |
| Product Key | DRADAC |
| Catalog Page | Page 10 (C-6-2019) |
| GTIN | 4055626356280 |
| Weight per Piece (including packing) | 286.3 g |
| Weight per Piece (excluding packing) | 223 g |
| Customs tariff number | 85371091 |
| Country of origin | DE |

Controller - AXC F 2152

2404267

<https://www.phoenixcontact.com/gb/products/2404267>


Technical Data

Notes

| | |
|-------------------------|---|
| Utilization restriction | |
| CCCEX note | Use in potentially explosive areas is not permitted in China. |

Product properties

| | |
|----------------|-----------------|
| Type | modular |
| Product type | Controller |
| Product family | PLCnext Control |

Insulation characteristics

| | |
|----------------------|---------------------------------------|
| Protection class | III (IEC 61140, EN 61140, VDE 0140-1) |
| Overvoltage category | II |

Display

| | |
|---------------------|----|
| Diagnostics display | No |
|---------------------|----|

Electrical properties

Real-time clock

| | |
|----------------------------|------------------------------|
| Realtime clock | Yes |
| Description realtime clock | 1.73 s/day = 20 ppm at 25 °C |

Potentials: Communications power U_L feed-in (the supply of the Axioline F local bus U_{BUS} is generated from U_L)

| | |
|----------------------|---|
| Supply voltage | 24 V DC |
| Supply voltage range | 19.2 V DC ... 30 V DC () |
| Current draw | max. 442 mA (with 1 A at U_{BUS} for the I/Os and $U_L = 24$ V) |
| Power consumption | max. 10.6 W (with 1 A at U_{BUS} for the I/Os and $U_L = 24$ V) |
| Protective circuit | Surge protection; electronic Reverse polarity protection; electronic |

Potentials: Axioline F local bus supply (U_{BUS})

| | |
|-------------------|------------------------------|
| Supply voltage | 5 V DC (via bus base module) |
| Power supply unit | 1 A |

Connection data

Axioline F connector

| | |
|-----------------------------------|---|
| Connection method | Push-in connection |
| Conductor cross section, rigid | 0.2 mm ² ... 1.5 mm ² |
| Conductor cross section, flexible | 0.2 mm ² ... 1.5 mm ² |
| Conductor cross section AWG | 24 ... 16 |
| Stripping length | 8 mm |

Interfaces

Controller - AXC F 2152

2404267

<https://www.phoenixcontact.com/gb/products/2404267>


| | |
|------------|-----|
| Web server | yes |
|------------|-----|

Axioline F local bus

| | |
|----------------------|-----------------|
| Number of interfaces | 1 |
| Connection method | Bus base module |
| Transmission speed | 100 Mbps |

Ethernet

| | |
|-------------------------------|--|
| Bus system | RJ45 |
| Number of interfaces | 2 |
| Connection method | RJ45 jack |
| Note on the connection method | Auto negotiation and autocrossing |
| Transmission speed | 10/100 Mbps (full duplex) |
| Transmission physics | Ethernet in RJ45 twisted pair |
| Transmission length | max. 100 m |
| No. of channels | 2 |
| Protocols supported | HTTP HTTPS SFTP SNTP IPsec syslog OPC UA |

System properties

| | |
|------------------------|--|
| Processor | Arm® Cortex®-A9 2x 800 MHz |
| Flash memory | 512 Mbyte (internal flash memory) SD card from Phoenix Contact (for external flash memory, see accessories) |
| Retentive data storage | 48 kByte (NVRAM) |
| RAM | 512 Mbyte DDR3 SDRAM |

IEC 61131 runtime system

| | |
|----------------|----------|
| Program memory | 8 Mbyte |
| Data storage | 12 Mbyte |

Axioline

| | |
|---|--|
| Amount of process data | max. 1482 Byte (per station (total input and output data)) |
| | max. 1024 Byte (Axioline F local bus (input)) |
| | max. 1024 Byte (Axioline F local bus (output)) |
| Number of supported devices | max. 63 (per station) |
| Number of local bus devices that can be connected | max. 63 (observe current consumption) |

PROFINET

| | |
|-----------------|--------------------------------------|
| Device function | PROFINET controller, PROFINET device |
| Update rate | min. 1 ms (4 devices) |
| | min. 16 ms (64 devices) |

Controller - AXC F 2152

2404267

<https://www.phoenixcontact.com/gb/products/2404267>


| | |
|-----------------------------|----------------------------------|
| Conformance Class | B |
| Number of supported devices | max. 64 (at PROFINET controller) |

Function

| | |
|-----------------------|-----|
| Diagnostics display | No |
| Controller redundancy | yes |
| Safety function | No |

Software

Functionality

| | |
|---------------------------------|------------------------------|
| Programming languages supported | Instruction list (IL) |
| | Symbolic flowchart (SFC) |
| | Ladder diagram (LD) |
| | Function block diagram (FBD) |
| | Structured text (ST) |
| | C++ |
| | C# |
| | Java |
| | Python |
| Simulink | |

System requirements

| | |
|-----------------------|-------------------|
| Engineering tool | PLCnext Engineer |
| | Eclipse |
| | Visual Studio |
| | MATLAB® Simulink® |
| Application interface | OPC UA |

IoT capabilities

Platform: PROFICLOUD

| | |
|--------------------------|-----|
| Supports cloud computing | yes |
|--------------------------|-----|

Dimensions

| | |
|---------------------|--|
| Dimensional drawing | |
| Width | 45 mm |
| Height | 126.93 mm |
| Depth | 75 mm |
| Note on dimensions | The depth applies when a TH 35-7.5 DIN rail is used (in accordance with EN 60715). |

Controller - AXC F 2152

2404267

<https://www.phoenixcontact.com/gb/products/2404267>



Environmental and real-life conditions

Ambient conditions

| | |
|--|---|
| Degree of protection | IP20 |
| Ambient temperature (operation) | -25 °C ... 60 °C up to 2000 m above mean sea level (observe derating) |
| | -25 °C ... 55 °C up to 3000 m above mean sea level (observe derating) |
| | ≤ 55 °C (with max. 1 A on U _{Bus}) |
| | > 55 °C ... 60 °C (only in conjunction with an Axioline F power module AXL F PWR 1H (Item no. 2688297)) |
| Ambient temperature (storage/transport) | -40 °C ... 85 °C |
| Permissible humidity (operation) | 5 % ... 95 % (according to DIN EN 61131-2) |
| Permissible humidity (storage/transport) | 5 % ... 95 % (according to DIN EN 61131-2) |
| Shock (operation) | 30g (in accordance with EN 60068-2-27/IEC 60068-2-27) |
| Vibration (operation) | 5g (in accordance with EN 60068-2-6/IEC 60068-2-6) |
| Air pressure (operation) | 70 kPa ... 106 kPa (up to 3000 m above sea level) |
| Air pressure (storage/transport) | 58 kPa ... 106 kPa (up to 4500 m above mean sea level) |

Approval data

ATEX

| | |
|----------------|---|
| Identification | <input type="checkbox"/> II 3 G Ex ec IIC T4 Gc |
| Certificate | TÜV 19 ATEX 8356 X |

IECEX

| | |
|----------------|--------------------|
| Identification | Ex ec IIC T4 Gc |
| Certificate | IECEX TUR 19.0031X |

UL, USA / Canada

| | |
|----------------|---------|
| Identification | cULus |
| Certificate | E238705 |

UL Ex, USA / Canada

| | |
|----------------|------------------------------------|
| Identification | Class I, Zone 2, AEx nA IIC T4 |
| | Class I, Div. 2, Groups A, B, C, D |
| | Ex nA IIC Gc T4 |
| Certificate | E366272 |

Corrosive gas test

| | |
|----------------|---|
| Identification | ISA S71.04.2013 G3 Harsh Group A, DIN EN 60068-2-60:2016-06 Method 4 |
|----------------|---|

EMC data

| | |
|---------------------------------|---|
| Conformance with EMC directives | Immunity test in accordance with EN 61000-6-2/IEC 61000-6-2 Electrostatic discharge (ESD)EN 61000-4-2/IEC 61000-4-2 Criterion B, 6 kV contact discharge, 8 kV air discharge |
|---------------------------------|---|

Controller - AXC F 2152



2404267

<https://www.phoenixcontact.com/gb/products/2404267>

| | |
|-------------------------------|---|
| | Immunity test in accordance with EN 61000-6-2/IEC 61000-6-2 Electromagnetic fieldsEN 61000-4-3/IEC 61000-4-3 Criterion A, Field intensity: 10 V/m |
| | Immunity test in accordance with EN 61000-6-2/IEC 61000-6-2 Fast transients (burst)EN 61000-4-4/IEC 61000-4-4 Criterion B, 2 kV |
| | Immunity test in accordance with EN 61000-6-2/IEC 61000-6-2 Transient overvoltage (surge)EN 61000-4-5/IEC 61000-4-5 Criterion B, DC supply lines: ±0.5 kV/±0.5 kV (symmetrical/asymmetrical), fieldbus cable shield: ±1 kV |
| | Immunity test in accordance with EN 61000-6-2/IEC 61000-6-2 Conducted interferenceEN 61000-4-6/IEC 61000-4-6 Criterion A, Test voltage 10 V |
| | Noise emission test in accordance with EN 61000-6-4/IEC 61000-6-4 |
| Electromagnetic compatibility | Conformance with EMC Directive 2014/30/EU |

Mounting

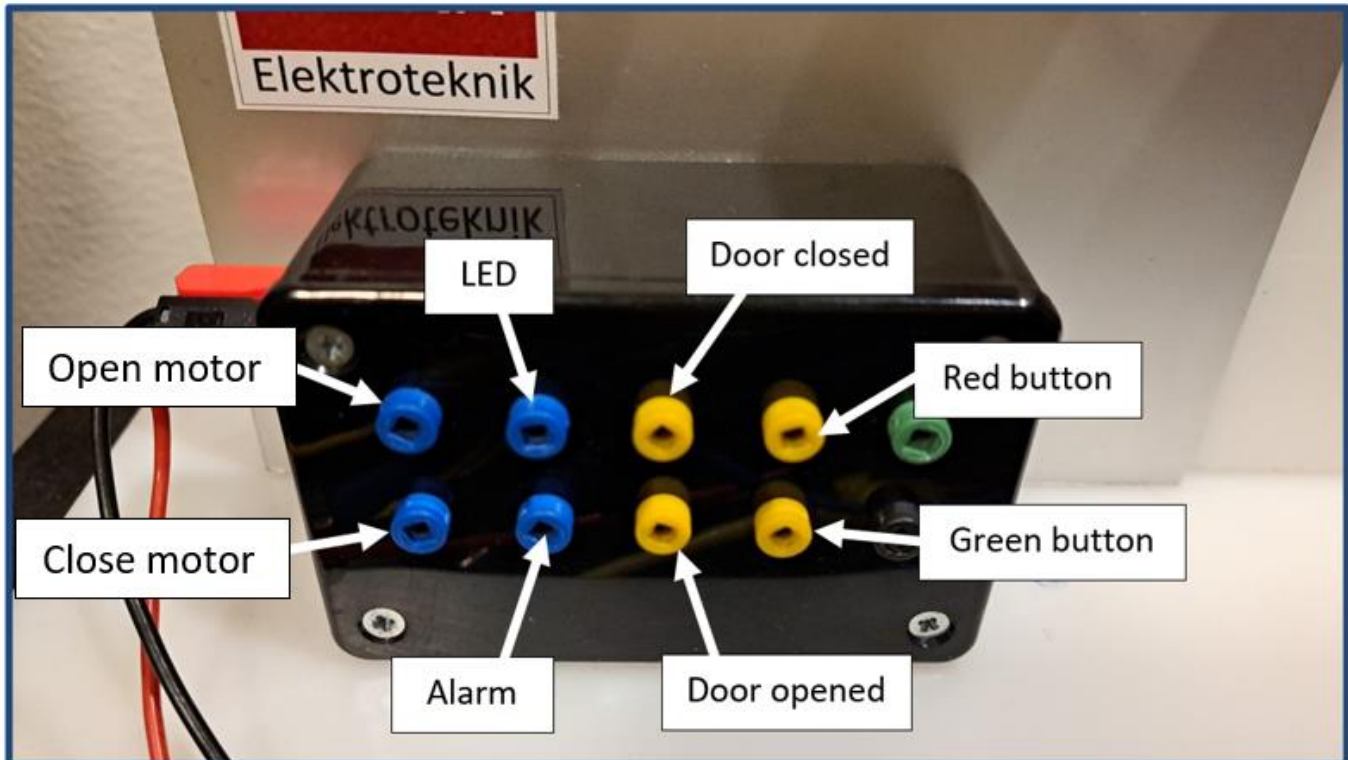
| | |
|---------------|-------------------|
| Mounting type | DIN rail mounting |
|---------------|-------------------|

Phoenix Contact 2023 © - all rights reserved
<https://www.phoenixcontact.com>

PHOENIX CONTACT Ltd
 Halesfield 13, Telford
 Shropshire, TF7 4PG
 01952 681700
info@phoenixcontact.co.uk

Appendix 3: PLC to Gate connections

Throughout the thesis, the connections made between the PLC and the industrial gate are not mentioned because it is considered out of scope. In the following, it is shown with informative signs to which corresponds each one of the inputs and outputs present in the industrial gate.



“Open motor” connected to “Digital out” 1

“Close motor” connected to “Digital out” 2

“LED” connected to “Digital out” 3

“Alarm” connected to “Digital out” 4

“Green button” connected to “Digital in” 1

“Red button” connected to “Digital in” 2

“Door close” connected to “Digital in” 3

“Door open” connected to “Digital in” 4

The connection in the PLC will look as follows:

