NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

LEONARDO ANTÓNIO LUÍS LUCAS

BSc in Electrical and Computer Engineering

WEB SUPERVISION SYSTEM OF A FREIGHT ELEVATOR

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon

2022, September

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL
AND COMPUTER ENGINEERING

WEB SUPERVISION SYSTEM OF A FREIGHT ELEVATOR

LEONARDO ANTÓNIO LUÍS LUCAS

BSc in Electrical and Computer Engineering

Adviser: Luís Filipe Figueira Brito Palma
Tenured Professor, NOVA University Lisbon

Co-adviser: Vasco da Silva Brito
Researcher, NOVA University Lisbon

Examination Committee

Chair: Nuno Filipe Silva Veríssimo Paulino
Associate Professor, NOVA University Lisbon

Rapporteurs: Luís Filipe Figueira Brito Palma
Tenured Professor, NOVA University Lisbon

João Almeida das Rosas
Assistant Professor, NOVA University Lisbon

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
2022, September

©

To my parents.

Acknowledgements

First of all, I would like to express my gratitude to Professor Luís Brito Palma, as my adviser, for being always available and interested in helping me during this journey and for all the great lessons and expertise transmitted. To Vasco Brito, as my co-adviser, for all the support that was crucial for the results of this work.

I would also like to thank all my friends from university and work that were with me during this time and who have always encouraged me during the rough times.

I would like to thank the NOVA University and its community that provided me the best experiences and teachings that I will carry with me forever.

Then, and most importantly, I want to thank my family and especially my parents for all the patience, understanding and support that made these last years much easier.

Abstract

Nowadays, automation and industrial control is an area in which there are innovations every day in terms of process digitalization, equipment interconnection and human-machine interaction, which results in a constant learning and adaptation to new technologies and methodologies developed. With this comes the responsibility to keep systems robust and prepared for eventual failures, while moving towards an increasing dependence on remote communication between different controllers and different processes. This fact leads to the need to create supervision and monitoring tools capable of detecting and transmitting existing failures, while ensuring that the system continues to operate with the same stability and performance.

Therefore, in this work it is proposed the development of a supervisory tool based on industrial automation that has a fault detection component and a human-machine interface in order to incorporate all the essential features of an industrial supervisor. Using industrial programming languages for Programmable Logic Controllers, it was possible to develop an algorithm that is based on inference mechanisms to identify potential faults in the system, which are then transmitted to the user in an interface that can be accessed either locally or remotely via the Web.

Keywords: Supervision, Human-Machine Interface, Fault Detection, Industrial Communication

Resumo

Nos dias de hoje, a automação e controlo industrial é uma área onde existe todos os dias inovações ao nível da digitalização de processos, da interconexão de equipamentos e na interação Homem-máquina, o que resulta numa constante aprendizagem e adaptação às novas tecnologias e metodologias desenvolvidas. Com isto, vem a responsabilidade de manter os sistemas robustos e preparados para eventuais falhas, ao mesmo tempo que se avança no sentido da cada vez maior dependência da comunicação remota entre diferentes controladores e diferentes processos. Este facto leva a que tenham de ser criadas ferramentas de supervisão e monitorização capazes de detetar e transmitir as falhas existentes, enquanto se garante que o sistema continua em funcionamento garantindo a mesma estabilidade e performance.

Assim, neste trabalho é proposto o desenvolvimento de uma ferramenta de supervisão baseada em automação industrial que possua uma componente de deteção de falhas e uma interface Homem-máquina de forma a incorporar todas as funcionalidades essenciais de um supervisor industrial. Recorrendo a linguagens de programação industrial para controladores lógicos programáveis, foi possível desenvolver um algoritmo que se baseia em mecanismos de inferência para identificar potenciais avarias no sistema que são posteriormente transmitidas ao utilizador numa interface que pode ser acedida quer localmente, quer remotamente via Web.

Palavras-chave: Supervisão, Interface Homem-Máquina, Deteção de Falhas, Comunicação Industrial

Contents

List of Figures

List of Tables

Acronyms

AC Alternating Current *(pp. 17, 28)*
AFTCS Active Fault Tolerant Control Systems *(pp. 4, 7)*
ASCII American Standard Code for Information Interchange *(pp. xv, 22, 23)*

CAN Controller Area Network *(pp. 21, 22, 24)*
CIP Common Industrial Protocol *(p. 24)*
CPU Central Process Unit *(pp. xvi, 16, 17, 32, 45)*
CRC Cyclic Redundancy Check *(p. 23)*

DC Direct Current *(p. 17)*
DRAM Dynamic Random-Access Memory *(p. 41)*

FBD Function Block Diagram *(pp. xv, 18–20, 40)*
FDD Fault Detection and Diagnosis *(pp. xv, 1–3, 6–9, 11)*
FL Fuzzy Logic *(pp. xv, 14)*
FRAM Ferroelectric Random-Access Memory *(p. 41)*
FTC Fault Tolerant Control *(pp. xv, 1, 2, 5–7, 35)*
FTCS Fault Tolerant Control Systems *(pp. 3, 6, 7)*

GB Gigabyte *(p. 40)*
GM General Motors Company *(p. 16)*

HLA High Level Architecture *(pp. xiv, xvi, 37–39)*
HMI Human-Machine Interface *(pp. xiv, xvi, xvii, 32, 35, 37, 38, 41, 43, 44, 47–49, 51–57, 65, 70–72, 77–79)*

I/O Input/Output *(pp. xvi, 16, 17, 47, 48)*
IEC International Electrotechnical Commission *(p. 17)*
IL Instruction List *(pp. xv, 18–20, 40)*

IP Internet Protocol *(pp. xix, 22–24, 41, 43–45, 48–50, 56, 57, 70, 77)*
IP Industrial Protocol *(p. 24)*
ISO International Organization for Standardization *(p. 21)*

kB Kilobyte *(p. 41)*

LD Ladder Diagram *(pp. xv, 17, 18, 20, 21, 40)*
LED Light-Emitting Diode *(p. 33)*
LRC Longitudinal Redundancy Check *(p. 23)*

MB Megabyte *(pp. 40, 41)*
MBAP Modbus Application Protocol *(p. 24)*
MQTT Message Queuing Telemetry Transport *(p. 79)*

NN Neural Networks *(p. 9)*

OSI Open System Interconnection *(pp. xv, 21, 22)*

PCA Principal Component Analysis *(pp. xv, 9, 10)*
PFTCS Passive Fault Tolerant Control Systems *(p. 4)*
PID Proportional–Integral–Derivative *(pp. 16, 32, 38, 63, 66, 67, 69, 70, 78)*
PLC Programmable Logic Controller *(pp. xiii, xv, xvi, 1–3, 16–18, 20, 22, 27, 32, 33, 40, 43–51, 56–59, 63, 65, 77)*

RAM Random-Access Memory *(p. 40)*
RJ Registered Jack *(pp. 40, 41)*
RTU Remote Terminal Unit *(pp. xv, 22, 23, 45, 49)*

SCADA Supervisory Control and Data Acquisition *(p. 2)*
SFC Sequential Function Chart *(pp. xv, 18, 19, 40)*
ST Structured Text *(pp. xv, 18, 20, 21, 40)*

TCP Transmission Control Protocol *(pp. xv, xvi, 22–24, 41, 45, 47–49, 70, 77)*

UDP User Datagram Protocol *(p. 24)*
USB Universal Serial Bus *(pp. 40, 41)*

VFD Variable-Frequency Drive *(pp. xiv, xvi, xvii, 28, 31, 32, 35, 38, 43, 45, 54, 63, 65, 66, 68, 71, 72, 74, 78)*

1 Introduction

1.1 Motivations

Nowadays, most industrial processes are controlled with automation systems, e.g., Programmable Logic Controller (PLC) devices, which are a major advance from the old hard-wired relay systems. These new automated systems are more compact, smarter and capable of controlling the behavior of many devices at the same time. Given this, PLC still need supervision, because in case of a fault the performance must not be the same when comparing to the nominal and fault-free performance. These named faults can cause serious damage to machinery or even fatal damage to humans, which means that these faults need to be detected as soon as possible to prevent these damages and to maintain a stable and safe environment.

In order to detect and overcome potential faults, Fault Tolerant Control (FTC) and Fault Detection and Diagnosis (FDD) systems are implemented in industrial control and supervision processes, which will prevent systems from collapsing and also improve reliability and efficiency, notions that are so important today from the perspective of industrial systems.

The research field of automatic supervision is an interesting and evolving area where a lot of progress has been made and much more is yet to be made. Given this fact, another important notion is the remote supervision, which nowadays is also becoming a more researched area. The digitalization of processes is becoming much more popular with the emergence of Industry 4.0 and, with this, the possibility of supervising physical processes remotely through WEB applications is an important feature to implement.

1.2 Objectives and Contributions

The objective of this thesis is to implement a WEB-based supervision system which will be tested and implemented on an elevator prototype used for educational purposes. This elevator is established in a laboratory in the Department of Electrical and Computer

Engineering - NOVA School of Science and Technology (NOVA SST). The work proposed has the main goal of creating a way of supervising the operation of the elevator and allowing this supervision to be made locally and remotely via WEB, which can be a major improvement on the system. This improvement will help students and professors to interact with this process remotely, which will contribute to the implementation of a remote laboratory and also to the digitalization of the processes present in the laboratory.

This work has the objective of proposing an alternative to Supervision systems used in the industrial context. In high-level processes, the more used supervision application is the Supervisory Control and Data Acquisition (SCADA) system. This system is very expensive and complex to implement, but it has a lot of useful functionalities. In the context of a laboratory, it is almost impossible to implement and interact with this type of systems, so the supervision system proposed can be viewed as a "Mini SCADA", since that in one hand it has fewer functionalities and it lacks on graphical options, but in the other hand it will implement a Fault Detection and Diagnosis (FDD) instrument, which typically is not implement in systems of this kind but is indeed very helpful.

1.3 Document Structure

This document is divided in three chapters:

- The first chapter introduces briefly the document. In this chapter the motivations, objectives, contributions and the document structure are presented.
- The second chapter presents the state of the art regarding three main subjects: Fault Tolerant Control (FTC), Fault Detection and Diagnosis (FDD) and Programmable Logic Controller (PLC). This chapter presents the research done in these specific areas which are essential for the development of the work proposed.
- The third chapter presents the proposed methodologies, the technologies used in the project, the ideas that were developed and the main functioning of the system proposed.
- The fourth chapter presents the experimental results and the tests performed during this work as well as some conclusions about these results.
- The fifth chapter is destined to conclusions and thoughts about possible future work.

2

State of the Art

2.1 Introduction

The focus of this chapter is to present and provide an overview of the main concepts on which this work is based. This chapter is divided into 3 sections where, firstly, the Fault Tolerant Control Systems (FTCS) are presented, going over the different approaches. In the next section there is an overview of the Fault Detection and Diagnosis (FDD) systems, describing the various methods and the last section presents the Programmable Logic Controller (PLC), their architectures, and their capabilities.

2.2 Fault Tolerant Control

Automated processes are always vulnerable to faults caused by humans, hardware or software or even any exterior factor, therefore some of them can be predicted while others may be unexpected. These faults can have major consequences in terms of performance or even critical damages to machinery or people which means that, given the inevitability of failures in a process, this must be taken into account when projecting control systems. Sensors, actuators, controllers or even the process itself can suffer from failures and these can turn into much bigger faults when spread to the whole system. Given this, faults need to be detected as soon as possible and, in some cases, corrected. Designing a control system which is able to manage different levels of failure importance, while still maintaining the stability of the system, has been one of the biggest challenges amongst experts in the field.

These systems are now called Fault Tolerant Control Systems (FTCS). FTCS are closed-loop control systems capable of detecting faults and accommodating them automatically. The main objective is to keep the integrity of a system in the event of an unexpected fault and still maintaining an acceptable performance. As the name states, FTCS are capable of tolerating faults or malfunctions in the system in order to maintain its availability (Aström et al. 2011; Y. Zhang and Jiang 2008).

2.2.1 Faults

So, first of all, what is a fault? In manufacturing, and in the point of view of the supervision, the term fault defines a situation when a variable is presenting values deviated from an accepted or nominal range, which can result, after a period of time, in production losses or damage to machines or humans (Blanke et al. 2006; Cardoso 2006).

There are three major classifications or sources of faults: sensor faults, actuator faults and plant faults. In fig. 2.1 it is shown a model with the input faults.

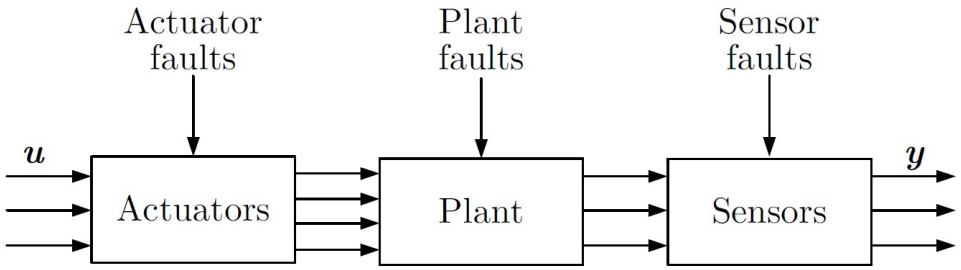


Figure 2.1: Different classifications of faults (Blanke et al. 2006).

- Sensor faults represent the values, associated to variables, that were misread by the sensors; those faults may happen as a result of unconnected wires, accumulated dirt on the sensor or damage on the sensor itself.
- Actuator faults concern the total or partial loss of action power by the actuators, which means that whether the actuators do not execute the functions they were supposed to, or the intended execution is defective; these faults can happen because of any mechanical or electrical defects.
- Plant faults are the faults not directly related to sensors or actuators, which are basically any modification in the parameters of the system which will result in a different behavior from the one expected.

Sensor and actuator faults are generally represented with additive faults in the system model, parameters of which are not changed. Plant faults are commonly represented with multiplicative faults and these are the ones that change the properties and dynamics of the system (Blanke et al. 2006; Cardoso 2006).

2.2.2 Approaches

In (Bartoszewicz 2011), (Jiang 2005), (Y. Zhang and Jiang 2008), (Mahmoud and Xia 2013) and (Patton 1997) two different approaches of Fault-Tolerant Control Systems are suggested: Passive Fault Tolerant Control Systems (PFTCS) and Active Fault Tolerant Control Systems (AFTCS). Passive approaches focus on having a robust controller to overcome

previously known faults while active approaches present a controller which is reconfigured after detecting faults in order to keep the desired performance and stability. In fig. 2.2 it is shown a typical structure of a FTC system with a block of fault diagnosis and another for controller reconfiguration.

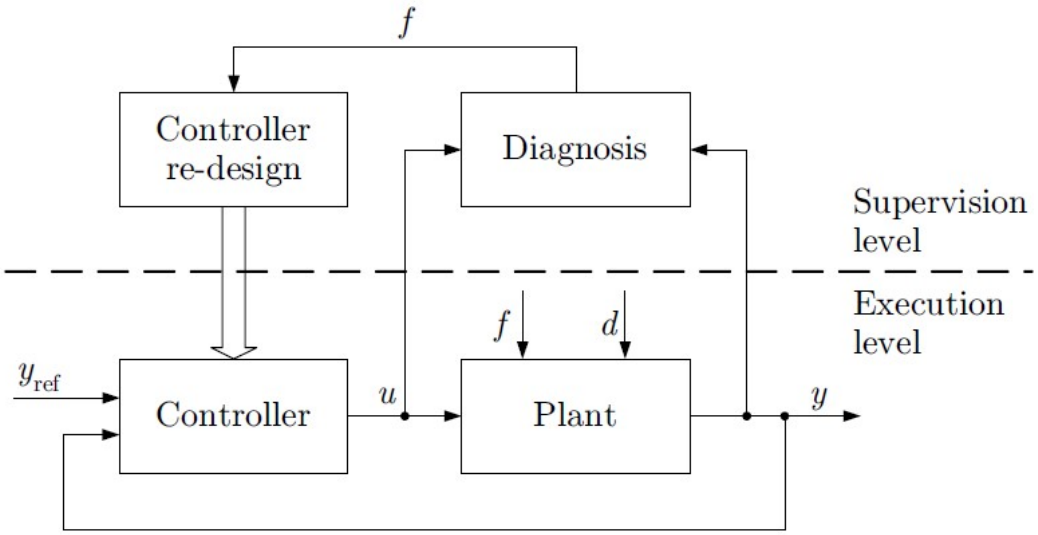


Figure 2.2: Structure of a FTC system (Blanke et al. 2006).

2.2.2.1 Passive FTC systems

As stated before, passive methods focus on maintaining a robust controller against expected faults. By having a fixed controller, these control systems can tolerate faults on the system between an acceptable range, which gives these systems a limited capability of overcoming faults, but in certain cases, when faults can be tolerated, this approach is sufficient. Given the fact that the controller is fixed, and therefore never change its parameters during the process, passive FTCS do not have a fault detection or controller reconfiguration mechanisms, as shown in fig. 2.3. These robust controllers rely only on the tolerance to certain faults and that explains the "passive" description, i.e., the controller's parameters are fixed, and the known faults are taken into consideration for its design, so it is previously designed to have the desired performance and stability in the occurrence of failures on the system. However, the controller can only guarantee the desired performance under the faults considered, so all the unexpected failures and uncertainties, depending on their severity, may destabilize the system (Jiang 2005; Y. Zhang and Jiang 2008).

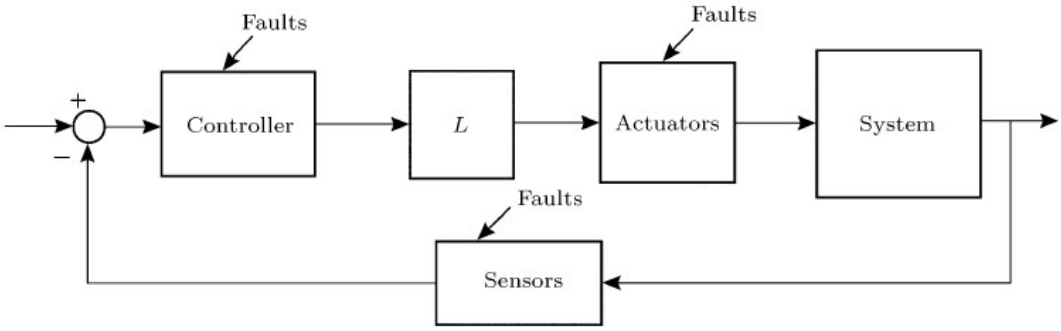


Figure 2.3: Structure of a passive FTC system (Jiang 2005).

2.2.2.2 Active FTC systems

On the other hand, active FTCS respond actively to faults, which means that the controller reacts to faults by constantly reconfiguring itself in order to maintain an acceptable performance. This approach is the best application on systems where any minor fault can have a major influence on the whole process. So, as opposed to the Passive FTCS applications, in these cases no fault can be tolerated, which means that the controller needs to be redundant and recalculate its parameters in order to overcome a fault and keep the system available at all times (Aström et al. 2011).

Active FTCS are mainly composed by a Fault Detection and Diagnosis (FDD) System, a reconfigurable controller and a reconfiguration tool (Fig. 2.4). So, comparing with Passive FTCS, active approaches include an additional layer of supervision where the FDD and the reconfiguration systems can be found.

These mechanisms working together continuously allow the system to be reconfigured as the faults are detected, while providing an adequate performance and stability. The

whole control system depends substantially on the FDD system to make the adjustment possible, given that the detection of a fault in real-time is what triggers the controller's reconfiguration.

One of the main challenges of the AFTCS is the amount of time needed to execute the fault detection and diagnosis and the system reconfiguration. The time available for reconfiguring the controller is very limited which means that the FDD system needs to be very fast and precise to maintain a functioning process even when the performance is not on the desired level (Baillieul and Samad 2015; Y. Zhang and Jiang 2008).

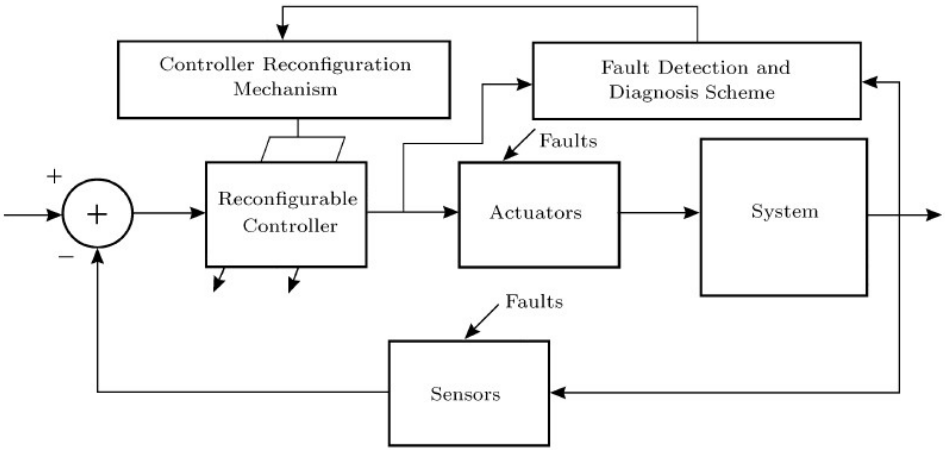


Figure 2.4: Structure of an active FTC system (Jiang 2005).

According to (Baillieul and Samad 2015) and (Patton 1997), Active FTCS can still be divided into two different approaches regarding the method of reconfiguring the controller. These two categories can be named as Projection Based and On-line Reconfiguration. The Projection Based approach proposes a finite number of pre-designed controllers where each one is projected to overcome a specific fault and when a fault or defect is detected, the reconfiguration tool will determine the best approach to overcome this fault. The On-line Reconfiguration approach calculates new controller parameters as faults are detected so that the controller is always up to date even when under faulty conditions.

2.3 Fault Detection and Diagnosis

Back in the day, traditional approaches to fault diagnosis were mainly hardware based, i.e., special sensors to measure specific values or multiple sensors to have a certain level of redundancy on the system. Besides hardware redundancy, the faults in the system were detected by human operators which made the systems more vulnerable and dependent on human skills.

As it was stated before, a big part of the recovery from errors by a system is made possible by the early detection of faults and failures. In critical situations such as nuclear,

chemical or power plants, aeronautics and astronautics the tolerance for faults is very limited, which means that the systems need to be available and reliable all the time not to cause any catastrophic failure or cause any risk to machinery or humans (Brito Palma 2007).

So, in order to detect and manage the faults that may occur in a manufacturing plant, a system of Fault Detection and Diagnosis (FDD) is implemented. FDD systems are incredibly important to maintain a safe environment for both machine and human. These systems have three main tasks (Frank 1996; Patton 1991): Fault Detection, Fault Isolation and Fault Identification, as shown in fig. 2.5. These steps are normally described as residual generation, residual evaluation and fault analysis, respectively.

1. **Fault Detection** registers that there is indeed a fault (residual generation, i.e., generates signals or symptoms reflecting the faults existent).

2. **Fault Isolation** locates the fault in terms of time and place (residual evaluation, i.e., logical decision-making on the time of occurrence and the location of a fault).

3. **Fault Identification** determines the sizes and type or nature of the fault (fault analysis, i.e., determination of the type of fault, size and cause).

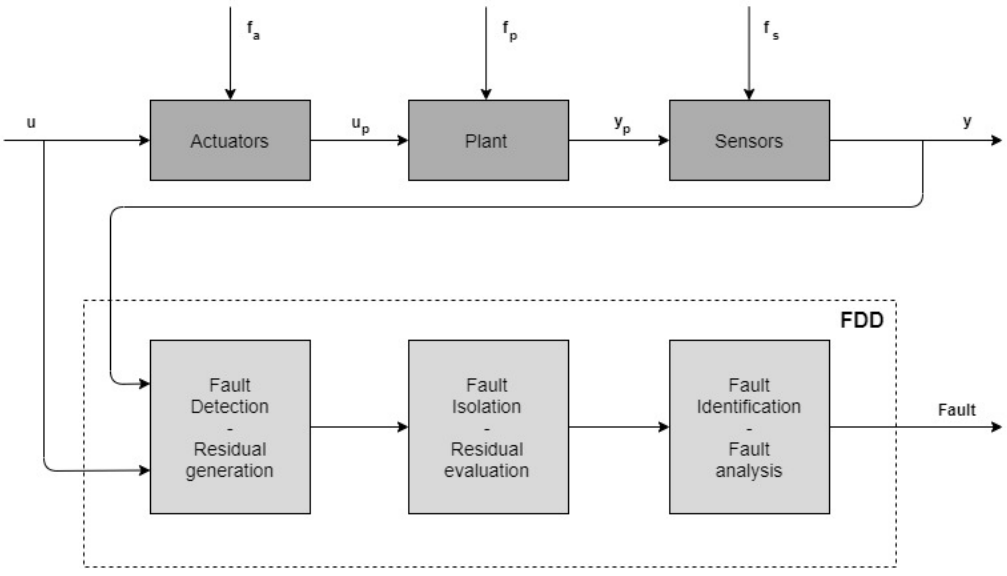


Figure 2.5: Structure of a FDD system.

2.3.1 Fault Detection Methods

In the literature, e.g., (Frank 1996; Khalastchi and Kalech 2018; Marzat et al. 2009; Miljković 2011), there are many in-depth studies about Fault Detection and its methods

and techniques. Given this, generally FDD systems can be divided into three main groups, depending on its methods of residuals generation: data-based, model-based and knowledge-based.

Data-Based methods compare online data with historically stored values in order to detect fault symptoms (e.g., quadratic mean values, threshold values, covariances, etc.). Usually, data-based approaches make use of a known dataset to train a model that will be able to classify new received data in real time and differentiate it as healthy or defective behavior. Given this, this trained model relies heavily on the quality of the training data, which means that the examples used on the training must embrace the most scenarios possible in favor of getting a model that is stable and accurate.

These approaches have the advantage of being model-free, i.e., data-based methods do not need to replicate the system analytically, which in some cases is too complex or nearly impossible to have the correct model of a system or environment (Frank 1996; Khalastchi and Kalech 2018).

Two examples of data-based approaches in Fault Detection and Diagnosis are Principal Component Analysis (PCA) and Pattern recognition with Neural Networks (NN), which are explained next.

- Principal Component Analysis (PCA):

  The objective of PCA is to reduce the dimension of a large dataset while keeping the larger amount of information possible, which will make possible the processing and monitoring of its data. PCA selects only the relevant eigenvectors of the data's covariance matrix in order to reduce the dimension of the system, and these eigenvectors that were selected will then be used to engineer a linear time-invariant statistical model used to generate residuals (Marzat et al. 2009). In fig. 2.6 it is possible to see the steps of the application of the PCA method.

- Neural Networks (Pattern recognition):

  Neural Networks (NN), in this context, are used to train a model to detect faults. Giving the network enough scenarios (i.e., in faulty and nominal conditions), the system is then able to recognize what are the faulty situations, and even when there are not enough training data with faults, the network can insert those faults artificially.

  The most common architecture of NN is a feedforward network (Fig. 2.7), which has an input layer, an output layer and, between, hidden layers. These hidden layers can have a single or a multi-layer layout. The number of layers, and neurons, will affect the efficiency of classification and the overall performance of the network, so it is a difficult choice and definitely a challenge (Khalastchi and Kalech 2018; Miljković 2011).

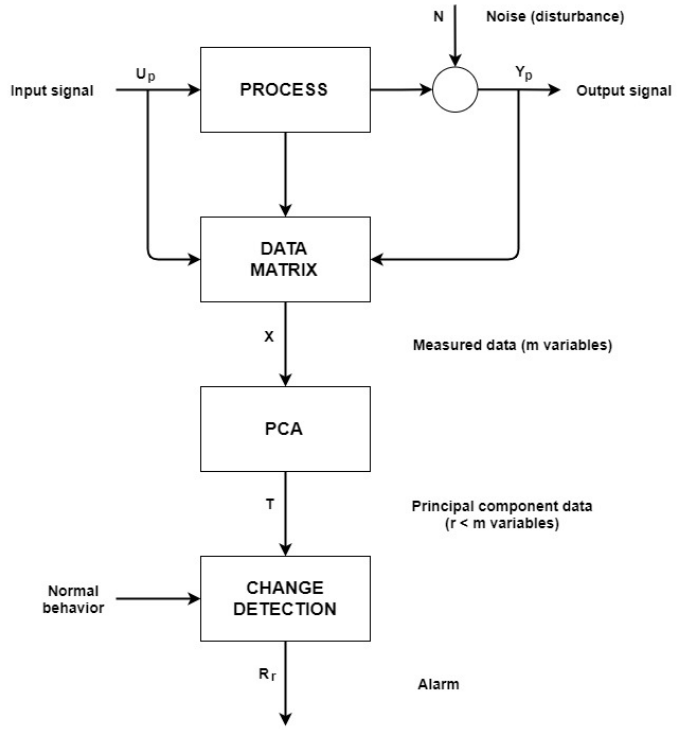


Figure 2.6: Fault detection system using PCA method (Miljković 2011).

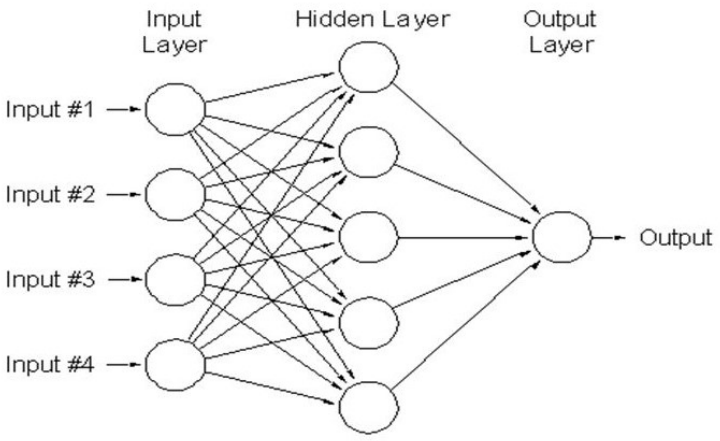


Figure 2.7: Feedforward Neural Network.

Model-based methods of fault detection are based on analytical redundancy, which implies that the outputs of the system supervised are constantly compared with the outputs of the system's analytical model. Given this, one can understand that model-based approaches are dependent on a mathematical model of the system, and also on the knowledge of its behavior, in order to process the fault detection. Any deviation from the expected values can be assumed to be a fault.

One main advantage of model-based approaches is how quick an unknown fault can be detected online. Nevertheless, the disadvantages are the difficulty to correctly replicate a system using analytical equations and formulas, as mentioned before, and obviously the cost of the whole process too (Khalastchi and Kalech 2018).

Three examples of model-based approaches in FDD systems are parity equations, State and output estimation and parameter estimation, which are explained next.

- Parity equations

  Fault Detection with parity equations is executed by comparing the actual behavior of the system with a modeled non-faulty behavior of the same system. This way it is possible to monitor the parity of the equations of the system. If the parity does not exist, so there is generation of residuals and this expresses a faulty behavior (Miljković 2011). In fig. 2.8 it is shown a fault detection system where the parity equations method is applied.

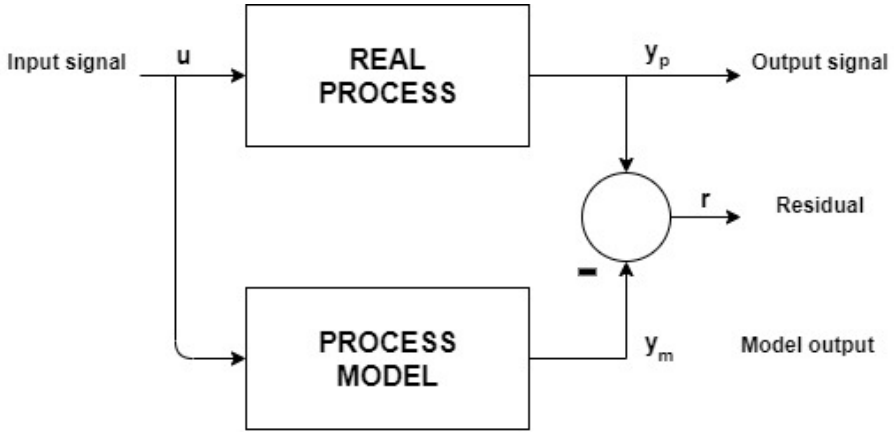


Figure 2.8: Fault detection system using parity equations (Miljković 2011).

- State estimation:

  The state estimation method, also called state observation, creates residuals by estimating, as the name states, the states of the system and then comparing them with the real state values. Differences between the values measured will indicate a fault in the system. Luenberger observers and Kalman filters are the most common when considering linear models. For nonlinear models, the solution is to linearize the model and then implement one of the techniques mentioned before (Marzat et al. 2009). In Fig. 2.9 it is possible to observe the architecture of state estimators.

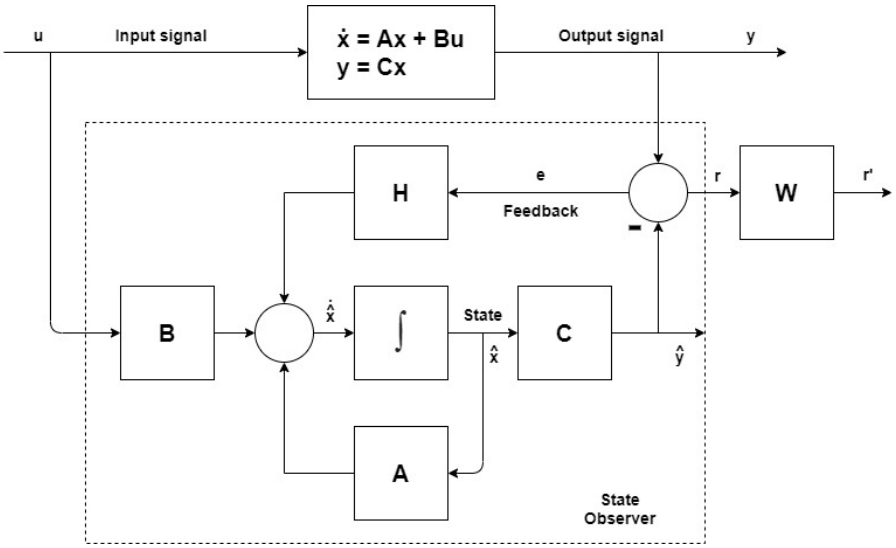


Figure 2.9: Fault detection system using state estimation (Miljković 2011).

- Parameter estimation:

  The method of parameter estimation is implemented when there are faults that may change the physical parameters or characteristics of the system (e.g., inductance, capacitance, resistance, mass, etc.), so any change on these parameters, out of accepted bounds, reflects a possible fault. If the model of the system is known, by measuring its inputs and outputs, it is possible to measure the parameters and weights involved in the process (Fig. 2.10). The residual generation will occur if there is a discrepancy between the parameters estimated and the parameters of the fault-free model. In linear models, one common method is the recursive least-square algorithm to estimate the parameters, if these are also linear, while in nonlinear models the computation may be much more complex and expensive (Marzat et al. 2009).

Knowledge-based approaches try to combine model-based and data-based methods which results in a hybrid approach where the data-based approach detects a fault, and the

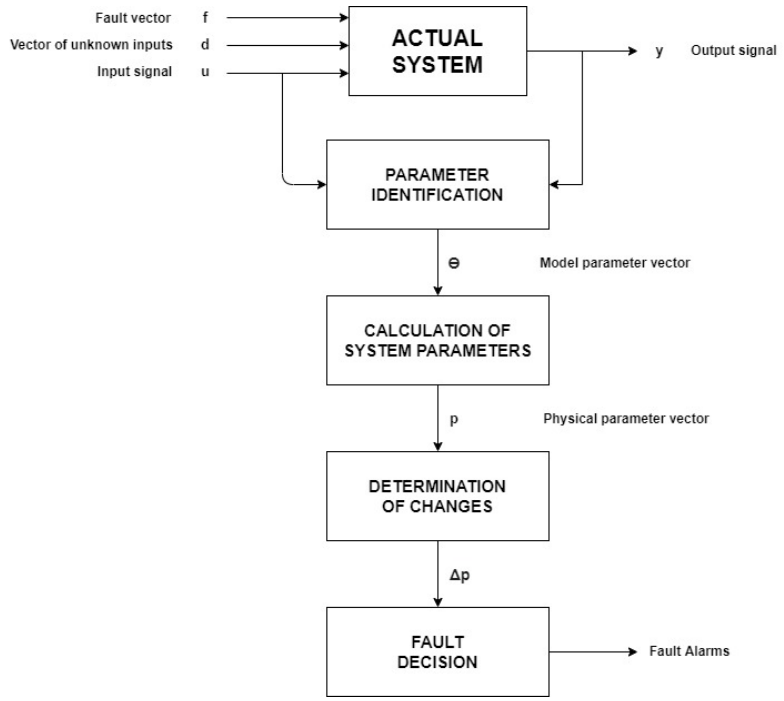


Figure 2.10: Fault detection system using parameter estimation (Miljković 2011).

model-based approach correlates that fault to a diagnosis. The knowledge behind these methods is retrieved from human experts, so that knowledge-based methods replicate human reasoning. Recently, knowledge-based methods have been a major object of study and, with all the innovations, these can also be associated with artificial intelligence methods (Khalastchi and Kalech 2018).

- Expert systems:

  Expert systems try to mimic the reasoning and behavior of a human operator. The knowledge of an expert-system is represented by rules written (and easily added or removed) by experts, and these rules are usually IF-THEN algorithms. Given this fact, this method relies on Boolean results, which makes these systems vulnerable to ambiguities. This can be sufficient for known faults but, when handling with the unknown, the systems can be unable to act properly and this can be explained by the lack of generality and inability to learn by itself (Miljković 2011). It is shown in fig. 2.11 the interactions between an expert system and the humans involved.

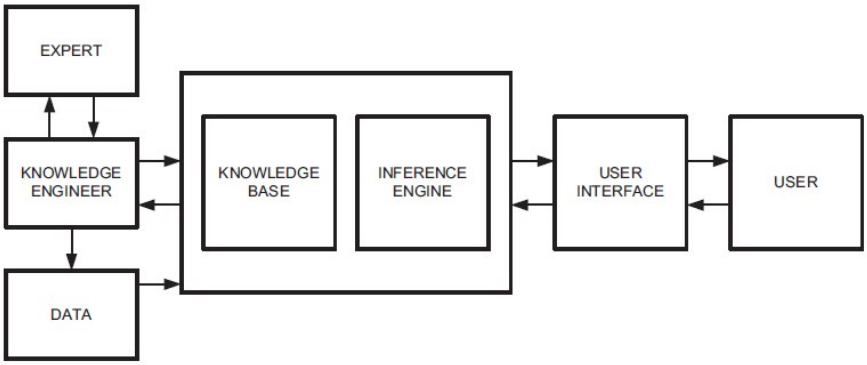


Figure 2.11: Fault detection system using expert systems (Miljković 2011).

- Fuzzy Logic (FL):

  A method to overcome the previous problem of binary decisions is FL controllers. The output of a FL system is interpreted by an expert in order to grant the right decision and not only a true/false decision. The main architecture of the fuzzy inference is presented in the following picture. To complete the process, the following steps are taken: Fuzzification, where the inputs, which are raw values, are represented graphically depending on the magnitude of their relevance (membership function); Rule based inference, where the IF-THEN rules are evaluated by a method of fuzzy reasoning, simulating the human reasoning, and the control actions to execute are decided depending on the inputs; Defuzzification, the last step where the fuzzy information is converted to output values. The steps of Fuzzy Logic are shown in fig. 2.12.

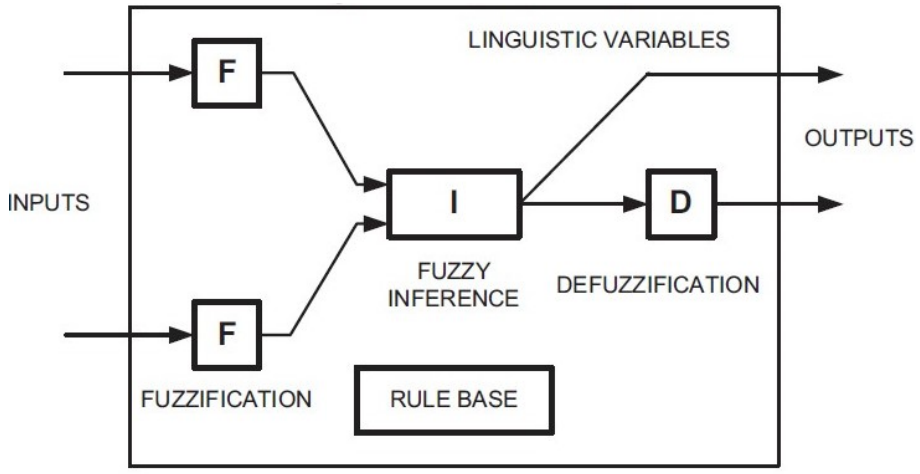


Figure 2.12: Structure of a FL controller (Miljković 2011).

Figure 2.13 shows an overview of the approaches to Fault Detection and the examples of these approaches mentioned before.

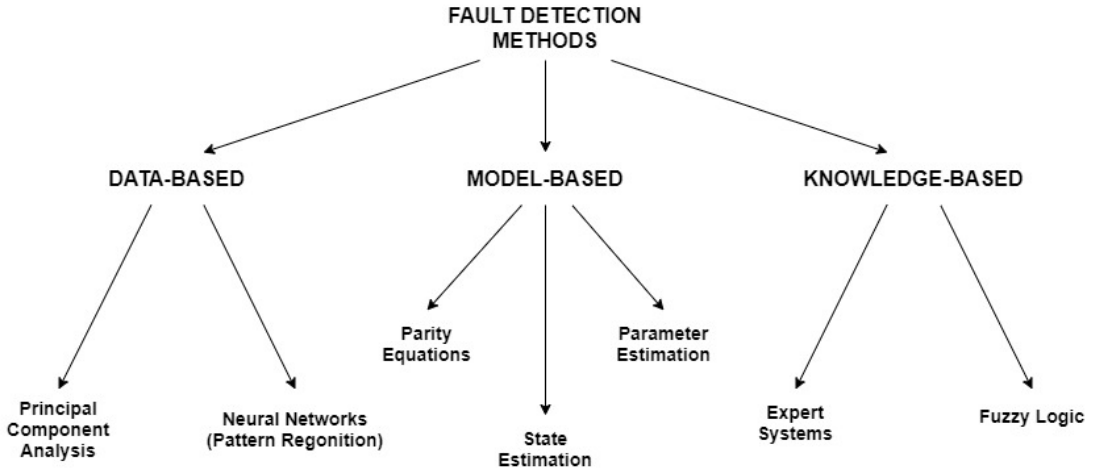


Figure 2.13: Overview of categories and techniques of Fault Detection.

2.3.2 Fault Diagnosis Methods

Fault Diagnosis occurs after Fault Detection. This task will determine the size, location, and time of the Fault Detection, which includes Fault Isolation and Fault Identification, as discussed earlier. Fault Diagnosis approaches can be divided into two categories (Brito Palma 2007): classification methods and reasoning methods.

The classification methods are used to indicate a change in symptom vectors in order to detect a fault. Some examples of classification methods are:

- geometrical distance and probabilistic methods;
- artificial neural networks;
- fuzzy clustering.

If it is possible to relate symptoms and faults and obtain diagnostic models, then it is possible to apply reasoning methods, based on causalities and inferences. Some examples of reasoning methods are:

- probabilistic reasoning;
- possibilistic reasoning with fuzzy logic;
- reasoning with artificial neural networks.

2.4 Programmable Logic Controllers

In the past, industrial machinery was controlled using mechanical systems (for example hydraulic cylinders) or even manually. This fact resulted in much larger infrastructures and a greater need for human power. With the development of discrete electronics, these mechanical control systems were after replaced by electronic control loops composed by transducers, relays and hard-wired control circuits.

In the 1960s, there was a huge development of microprocessors and integrated circuits, which was a major breakthrough to the emergence of the Programmable Logic Controller (PLC), first developed by General Motors Company (GM) in the late 1960s, with the following specifications (Erickson 1996; Galloway and Hancke 2012):

- Easily programmed and reprogrammed.
- Easily maintained and repaired.
- Capable of operation in a plant environment.
- Smaller than relay equipment.
- Capable of communicating with central data collection system.
- Cost-competitive with the relay systems then in use.

Hereupon, PLC came to revolutionize the control of manufacturing lines, since before it was done by large panels of hard-wired relay panels, which were hard to troubleshoot in the event of a problem, and now the same functions could be achieved by a compact and computer-based system that was simple to program and resistant to all the inherent adversities of an industrial plant such as dust, extremely high or low temperatures, collisions and vibrations. The modularity of these systems reflected their flexibility and ease of maintenance which, therefore, proved to be much more practical than relay-based systems used until then.

So, by definition, PLC are solid-state control systems that have a user-programmable memory for storing instructions for the purpose of implementing specific functions such as Input/Output (I/O) control, logic, timing, counting, three mode PID control, communication, arithmetic, and data and file processing (Stouffer, Falco, and Scarfone 2011).

2.4.1 PLC Architecture

In Fig. 2.14 it is possible to see the main architecture and the components of PLC. The architecture consists of five main components (Bolton 2015; P. Zhang 2010):

1. **Central Process Unit (CPU)**: basically the "brain" of PLC. It has the capability of calculating the outputs according to the inputs received and the program stored in

its memory. The CPU will save anything saved in its memory even when the PLC suffered a power loss.

2. **Memory**: needed for storing the program that contains all the control decisions to be posteriorly executed by the CPU and also all the data regarding inputs and outputs handled by the program.
3. **Power supply**: converts the usual AC voltage from the grid to the low DC voltage needed for the internal circuits (normally 5 or 15 V DC).
4. **Input/Output (I/O) boards**: the input board receives input signals, which can be analog or digital, from the external devices connected to the PLC (e.g., proximity sensors, magnetic field sensors and mechanical switches), and these signals are managed by the program in order to send control signals through the output board, which can also be analog or digital, to the desired machines (e.g., relays, motors and frequency inverters).
5. **Communication module**: the communication board enables the communication with other PLC or with a computer, where normally the program for the PLC is developed and then transferred using a communication cable connected to this module.

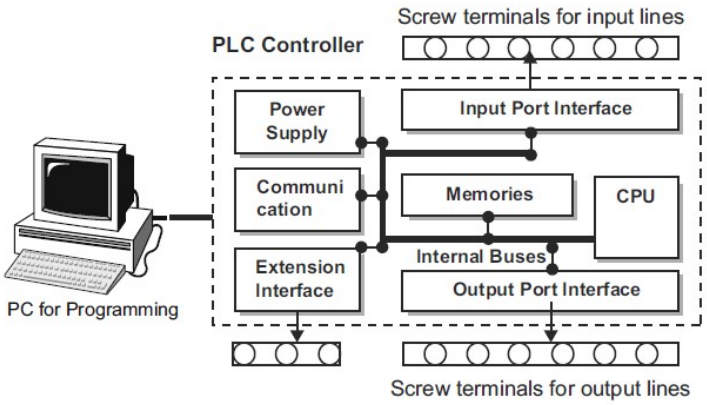


Figure 2.14: Architecture of PLC (P. Zhang 2010).

2.4.2 Programming languages

During the development of more PLC models and with the growth of more manufacturers, the interconnection between all these new devices began to seem impossible given the fact that every brand had their own set of proprietary software and hardware. So, to combat this lack of integration between systems and to create a standardized programming language, which could be easily understood by any operator, the Ladder Diagram (LD) was created and most manufacturers adopted this language in their programs. Nevertheless, standards about PLC programming languages were published, where International Electrotechnical Commission (IEC) 61131-3 stands out. IEC 61131-3 defines five main

PLC programming languages: Ladder Diagram (LD), Sequential Function Chart (SFC), Function Block Diagram (FBD), Instruction List (IL) and Structured Text (ST) (Bolton 2015). In fig. 2.15 it is possible to see the division of these five languages between textural and graphical languages.

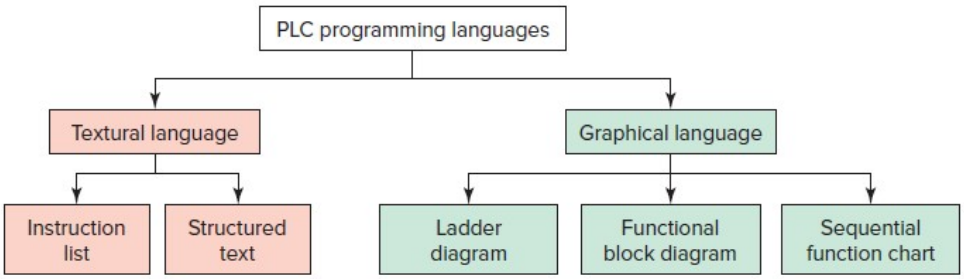


Figure 2.15: Division of PLC programming languages (Petruzella 2017).

2.4.2.1 Ladder Diagram (LD)

Ladder Diagram is a graphical language created to resemble the relay logic operations, which became the most popular among programmers and operators as a result of its similarity with physical relay systems. This language replicates all the elements of a common circuit, such as switches, timers, counters, coils and so on. Even when LD is implemented in complex systems, its graphical simplicity makes a program much easier to understand and, therefore, to debug.

One problem related to LD language concerns the fact that occasionally LD programs are not compatible between PLC since the manufacturers, despite following the standards, have different methods and structures to develop LD programs.

In Fig. 2.16, it is evident that the purpose of the LD language is mirroring a relay logic circuit.

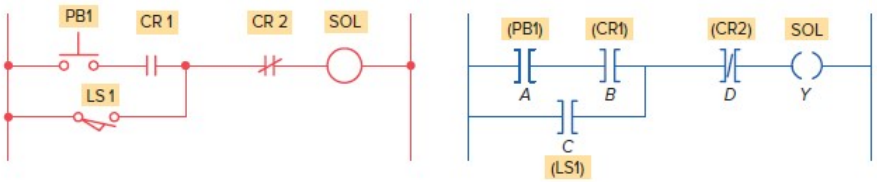


Figure 2.16: Comparison between relay logic, on the left, and LD language, on the right (Petruzella 2017).

2.4.2.2 Sequential Function Chart (SFC)

Sequential Function Chart is a language based on Grafcet and Petri nets which implements a sequence of states and transitions between states. The states are associated with actions to be executed while the transitions represent logical conditions. When designing a SFC

program, the programmer must take into account that in order to execute an action, the condition associated with the preceding transition needs to be verified. If the condition is false, the program remains in the same state and keeps executing its function until it returns true. These states can have associated more than one execution order, which means that multiple operations can be performed in parallel.

In Fig. 2.17, it is possible to observe one implementation example of a simple SFC program.

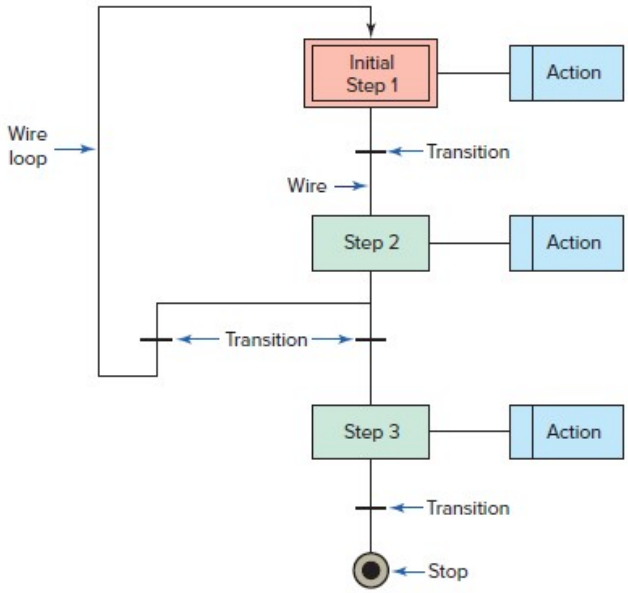


Figure 2.17: Example of SFC program (Petruzella 2017).

2.4.2.3 Function Block Diagram (FBD)

Function Block Diagram is another graphical language that is mainly based on blocks representing certain. These blocks can represent multiple functions as Boolean (e.g., AND, OR, XOR, NOT, etc.), arithmetic (e.g., adder, multiplier, subtractor, etc.), selection, timer and comparison. FBD programs follow a certain sequence given that an output from a block can be an input on another block, which creates a hierarchy between inputs and outputs. In fig. 2.18 there is an example of an FBD program.

2.4.2.4 Instruction List (IL)

IL is a low-level programming language resembling Assembly. On one hand it has a simple implementation, but on the other hand its methods have limited capabilities which only solves problems of low complexity. Given its simplicity, other advantage lies on the fact that IL programs, as they are text-based, take up less memory and therefore its execution is much faster when compared to graphical languages. The main objective of this language is to load, store and move values on memory and jump between instructions.

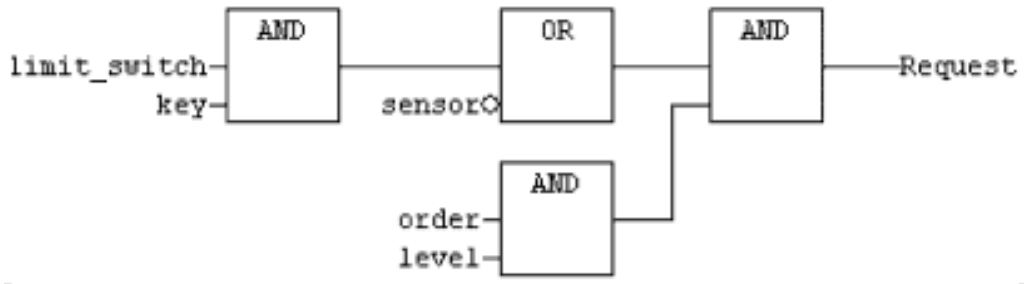


Figure 2.18: Example of FBD program.

In fig. 2.19, it is possible to observe a comparison between IL and ST languages.

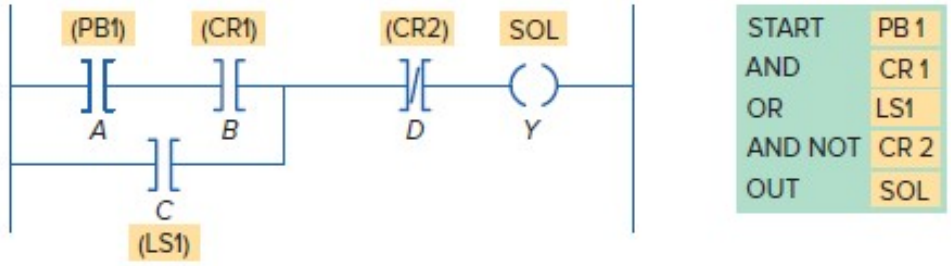


Figure 2.19: Comparison between LD language, on the left, and IL language, on the right (Petruzella 2017).

2.4.2.5 Structured Text (ST)

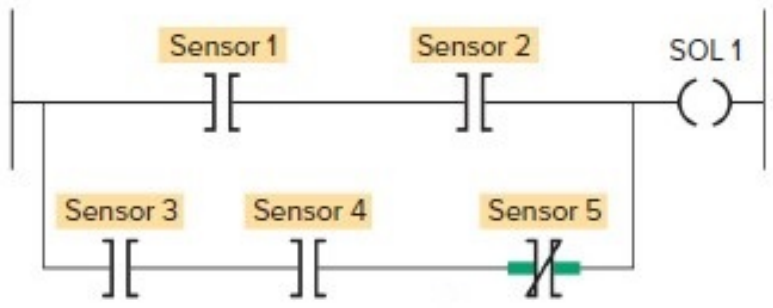
Structured Text is a high-level text language which syntax is based on another language named Pascal. ST allows the declaration of variables and the creation of function blocks using, for example, arithmetic, Boolean, comparison, conditional or loop functions.

This language is chosen over other languages in more complex operations, where graphical languages would become much harder to understand. The main advantages when comparing ST, as a textural language, with graphical languages is the small memory allocation that these ST programs need and the possibility of implementing complex yet compact procedures.

In Fig. 2.20, it is possible to observe a comparison between LD and ST languages.

2.4.3 Industrial communication protocols

With the increasing popularity of the PLC, there was the need to connect this device to the rest of the components of the system, such as sensors, actuators and computers. This was fundamental for a greater compatibility between devices and, with this, the transmission of data improved and became more useful for human operators. These proposed communication protocols came to substitute the traditional wiring methods




```
IF Sensor_1 AND Sensor_2 THEN
        SOL_1 := 1;
ELSEIF Sensor_3 AND Sensor_4 AND NOT Sensor_5 THEN
        SOL_1 := 1;
END_IF;
```

Figure 2.20: Comparison between LD language, above, and ST language, below (Petruzella 2017).

between relays, which were used in the plant level and typically had signals of 4-20 mA or 0-10 V (Galloway and Hancke 2012).

Industrial communication protocols followed however certain references for the correct implementations of the networks. One major reference for the configuration of these protocols is the Open System Interconnection (OSI) seven layer reference model, proposed by International Organization for Standardization (ISO) in the 1980s. The OSI network stack has the following layers (from the lower to the higher level layer): physical, data link, network, transport, session, presentation and application, as shown in Fig. 2.21.

2.4.3.1 Controller Area Network (CAN)

Controller Area Network (CAN) was initially developed by Robert Bosch GmbH in the 1980s and later in the 1990s CAN was recognized as a standard. Its design was originally destined for the automotive industry and then its use was spread to other areas such as robotics and medical equipment. CAN depended on two layers: data link layer and physical layer. Its transmission rate had a maximum of 1 Mbps at short distances (Irwin 1997). Over the time, given that this was a lower level protocol, higher level protocols were created which were based on CAN, e.g., CANopen, DeviceNet and ControlNet, which will be addressed next.

- CANopen

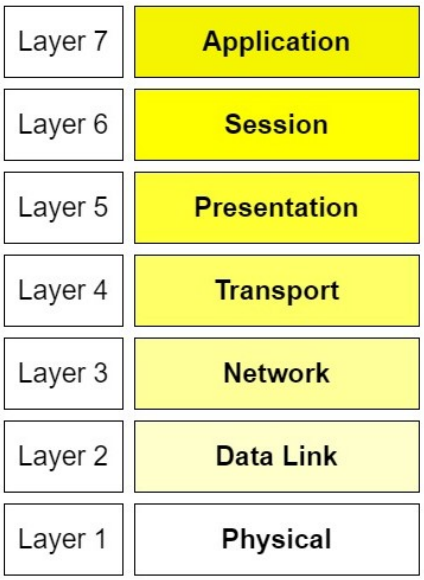


Figure 2.21: OSI reference model for network configuration.

CANopen is based on CAN protocol but it presents an application layer on its model, besides data link and physical layers, which makes CANopen an higher level protocol. This protocol is now widely implemented in building automation, machinery control and medical instrument. CANopen has three types of communication betwen devices: master-slave, client-server and producer-consumer, which covers the main systems where this protocol is used (Irwin 1997).

- ControlNet and DeviceNet

  As it was previously mentioned, ControlNet and DeviceNet are also higher layer protocols based on CAN, developed by Allen-Bradley (now owned by Rockwell Automation) and its use is more destined to industrial automation applications. ControlNet's objective is the trasnmission of control information and it is more used in redundant systems with scheduled processes and communications. DeviceNet is an adaptation from ControlNet, but more intended to communications between devices, e.g., PLC, sensors and actuators (Galloway and Hancke 2012).

2.4.3.2 Modbus

The Modbus protocol was proposed by Modicon (now owned by Schneider Electric) in the late 1970s, and its main goal was to transmit data between Programmable Logic Controller (PLC) and all kinds of industrial instruments. Since then, this protocol has been widely used between a variety of proprietary devices. Modbus communications can be divided into two types: Modbus serial, which includes Remote Terminal Unit (RTU) and ASCII modes, depends on a serial bus where all the devices are connected in a master-slave model; and Modbus TCP that depends on a TCP/IP network where all the devices are

connected in a client-server type of interaction (Irwin 1997). Modbus RTU, Modbus ASCII and Modbus TCP/IP are examined next.

- Modbus RTU

  As it is shown in fig. 2.22, Modbus RTU has four different types of information: address, function, data and Cyclic Redundancy Check (CRC). Each byte has the following 11 bit character configuration: 1 initial bit (ST) for the synchronization, 8 bits with the binary codification of the data, 1 bit of parity (PT) used for detecting errors and 1 final bit (SP) to separate different characters (Irwin 1997).

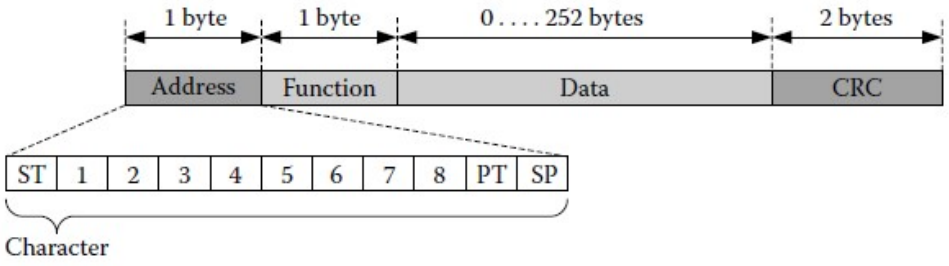


Figure 2.22: RTU transmission frame (Irwin 1997).

- Modbus ASCII

  Similar to Modbus RTU, Modbus ASCII also has a defined transmission frame, as it is shown in fig. 2.23. This frame is initiated by the char ":", that defines the beggining of a frame, and then the address, function, data and Longitudinal Redundancy Check (LRC). Finally, the end of the frame is occurs when the sequence "CF" (Carriage Return) and "LF" (Line Feed) is detected (Irwin 1997).

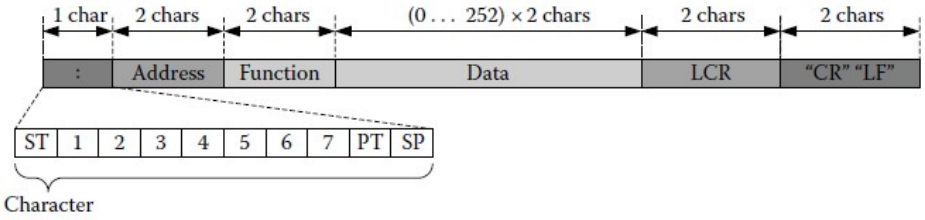


Figure 2.23: ASCII transmission frame (Irwin 1997).

- Modbus TCP/IP

  The frame of Modbus TCP/IP data transmission is shown in fig. 2.24. It has a Modbus Application Protocol (MBAP) header with the following information: transaction identifier that identifies to which transaction the frame belongs, protocol identifier, the length of the unit identifier expressed in bytes and finally the unit identifier which identifies the device that will receive the frame. After the MBAP header there are still present in the frame the function code and the data (Irwin 1997).

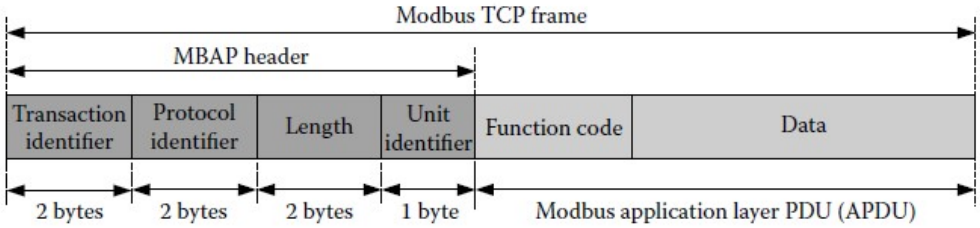


Figure 2.24: TCP transmission frame (Irwin 1997).

2.4.3.3 Industrial Ethernet

Industrial Ethernet, apart from the normal Ethernet, was more robust and designed to be applied in industrial environments. Given the fact that, in industrial plants, the data is required to be transmitted in real time, this was a problem with the standard Ethernet previously mentioned, and that was another reason for the creation of Industrial Ethernet.

- Ethernet/IP

  Ethernet/IP means Ethernet Industrial Protocol, which is different from Ethernet Internet Protocol. Ethernet/IP is an implementation of Common Industrial Protocol (CIP)'s application layer over Ethernet, similar to ControlNet and DeviceNet that use CIP over CAN. The messaging in this protocol is made over TCP/IP and UDP connections.

2.4.3.4 PROFIBUS

PROFIBUS is one of the most implemented communication protocols, being typically implemented in Siemens devices. PROFIBUS is based on a master-slave implementation, where the information from all the connected devices flow through a main bus and then reach a central device that manages all the data. The low level communication is made through a process called PROFIBUS Distributed Periphery (DP) and all the other communication use PROFIBUS Process Automation (PA). The physical layer of this protocol is based on RS-485 serial communication (Galloway and Hancke 2012).

- PROFINET

PROFINET is an adaptation of PROFIBUS but with certain differences. One main difference is that this protocol uses Ethernet as the physical layer. Nevertheless, the logic is the same as PROFIBUS with one main bus connected to all the devices in order to retrieve data to a main controller.

3

Proposed Approaches

3.1 Introduction

The focus of this chapter is to present and provide an overview of the solution proposed for the supervision system of a freight elevator model. This chapter is divided into 3 sections where, firstly, the freight elevator model is presented, going over the structure, the hardware, and its specifications. In the next section there will be an overview of the proposed solution, describing the high-level architectures and technologies adopted in this process. After establishing and explaining the general architecture of the proposed system, the following section will present the implementation of the project and all the steps taken, i.e., hardware and software configuration, communication protocols and all the system functionalities.

3.2 Freight Elevator System

As stated before, the supervision system proposed will be implemented in a freight elevator model present in the automation laboratory of the Department of Electrical and Computer Engineering, which means that, before describing the work that has been done, it is necessary to understand the system in which it was applied, its components and, most importantly, how it operates. This model was developed within the scope of the Master Thesis of Nuno Ângelo titled "Projecto e Concepção de um Sistema Elevador Monta-cargas Industrial" (Ângelo 2016). After this first project, there were some modifications in the following years, the last of which was implemented by Duarte Santos with the Master Thesis "Protótipo Ciber-Físico de Elevador Monta-Cargas" (Santos 2022), in which the electrical schematic was redone, updating the PLC and the algorithm behind the control of the process.

3.2.1 Hardware Description

This model of a freight elevator, or a cargo lift, has a structure that simulates a real elevator with a cabin that travels between three floors.

The cabin has two mono stable ampoules to indicate when it is in the region of a floor and two bi stable ampoules to indicate which floor it is.

In three different heights, along the travel route, there are inductive sensors to indicate the correct position of each floor and in each end of the course, there are two end-of-stroke detectors, to guarantee that the cabin does not descend or ascend further than allowed.

In order to know the precise position of the cabin, an ultrasonic sensor was implemented at the top of the structure which allows the controller to have the feedback needed about the operating status of the system.

Finally, at the top of the structure there is also a three-phased AC motor, controlled by a Variable-Frequency Drive (VFD), with a brake that, when energized, allows the motor to ascend or descend the elevator.

- **Process Hardware**

  - **Structure:**

    The model has a structure with 2.30 meters of height, 1.08 meters of width and 0.88 meters of depth as represented in figure 3.1.

Figure 3.1: Elevator Structure.

The left side of the structure is constituted by the elevator system where it has the cabin, the counterweight, multiple sensors, shock absorbers and the

traction machine, while on the right side is where the electrical cabinet is placed, containing both the power and control circuits.

- **Mechanical end-of-stroke detectors:**

  In figure 3.2 is represented the end-of-stroke detector used at the top and bottom of the structure to guarantee that, for safety reasons, the elevator stays within the range of movement and does not travel above or below the floors. In this case, the elevator should stop immediately to not compromise the integrity of the system.

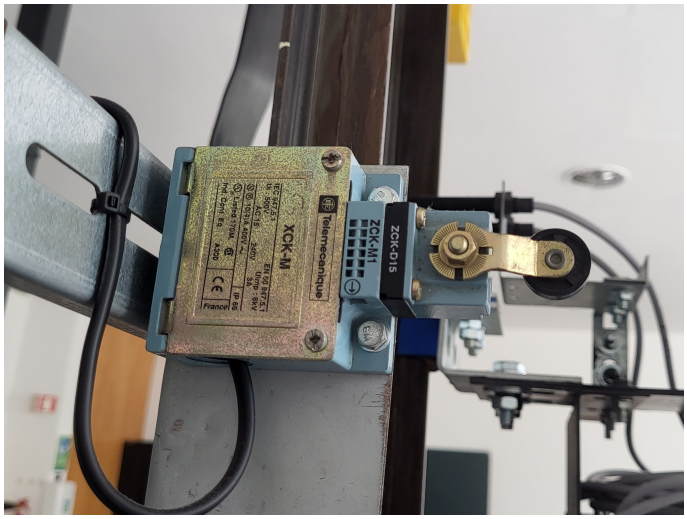


Figure 3.2: End-of-stroke detector.

- **Mono-stable and bi-stable ampoules:**

  At the top of the elevator cabin, there are four reed-switches, two of them mono-stable and the other two bi-stable, shown in figure 3.3. This means that the mono-stable reed-switches only have one contact that closes once it detects a magnetic field, while the bi-stable, or latching, reed-switches keep their contacts closed or opened until they are magnetized again.

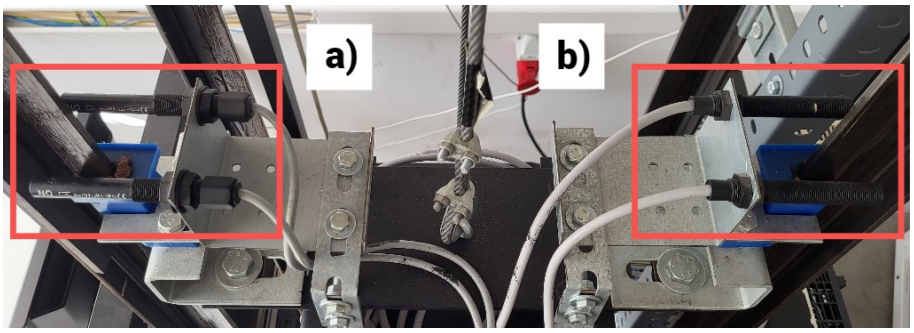


Figure 3.3: Reed-switch sensors: a) Mono-stable reed-switches, b) Bi-stable reed-switches.

The sensors on the left are mono-stable, dictating whether the cabin is currently

in a floor or not. The sensors on the right are bi-stable and depending on the combination of the two sensors, it is possible to determine specifically which floor the cabin is passing. The digital signal of each ampoule is then transmitted to the controller.

- **Inductive sensors**

  Each floor, three in total, has an inductive sensor, as illustrated in figure 3.4, that detects a metallic strip of the cabin, and transmits this digital feedback to the controller indicating the correct position of a floor.

Figure 3.4: Inductive sensor.

- **Ultrasonic sensor:**

  The ultrasonic sensor was only implemented in this process some years later within the scope of the Master Thesis titled "Controlo e Estudo Dinâmico de Ascensores para Otimização do Conforto Humano" done by André Monteiro (Monteiro 2016), which had the objective of improving the movement of the elevator considering the comfort of hypothetic passengers of the elevator.

  Given this, the sensor kit has a transmitter and a receptor of ultrasonic waves that, taking into account the time the sound took to travel, allows the system to know the exact position of the cabin.

  Figure 3.5 shows the Arduino board that was used to not only process the signal received from the sensors, but also filter it before sending an analog input to the controller.

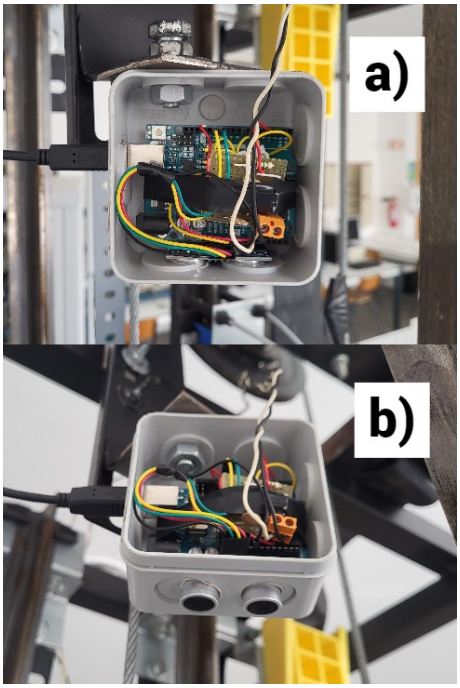


Figure 3.5: Ultrasonic sensor: a) Arduino board to process the signal, b) Ultrasonic sensor emitter and receptor.

- **Traction machine:**

  The motor at the top of the structure, also mentioned as the traction machine, is responsible for ascending and descending the cabin depending on the commands of the Variable-Frequency Drive (VFD). This means that it receives the setpoint and the direction of the movement from the VFD. This motor has a mounted break that, when energized, allows the motor to run freely. In the absence of power, the break is engaged, to ensure that the elevator ceases its movement and that it is held properly in the desired position. The traction machine can be observed in figure 3.6.

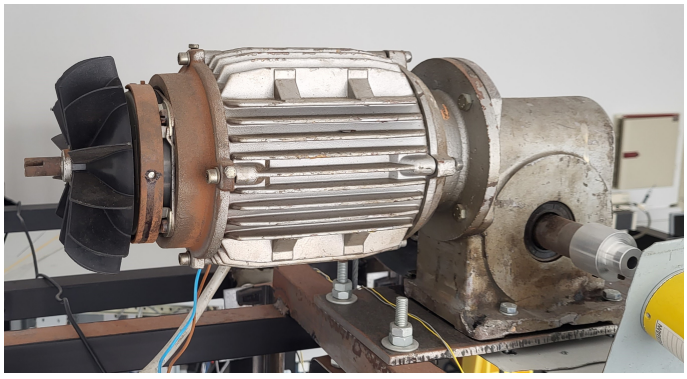


Figure 3.6: Traction machine.

- **Control Hardware**
  - **Variable-Frequency Drive (VFD):**
    In order to control the speed of the cabin, a Variable-Frequency Drive was needed, and the specific VFD used was manufactured by Omron and is represented in figure 3.7. This VFD sends the reference frequency, received from the controller, to the actuator, receiving also its actual frequency. The communication between the VFD and the PLC can be done through two different methods: the first is done by digital and analog signals and the second is the Modbus Protocol. The VFD has a HMI built-in to allow users to access and change parameters in real-time.

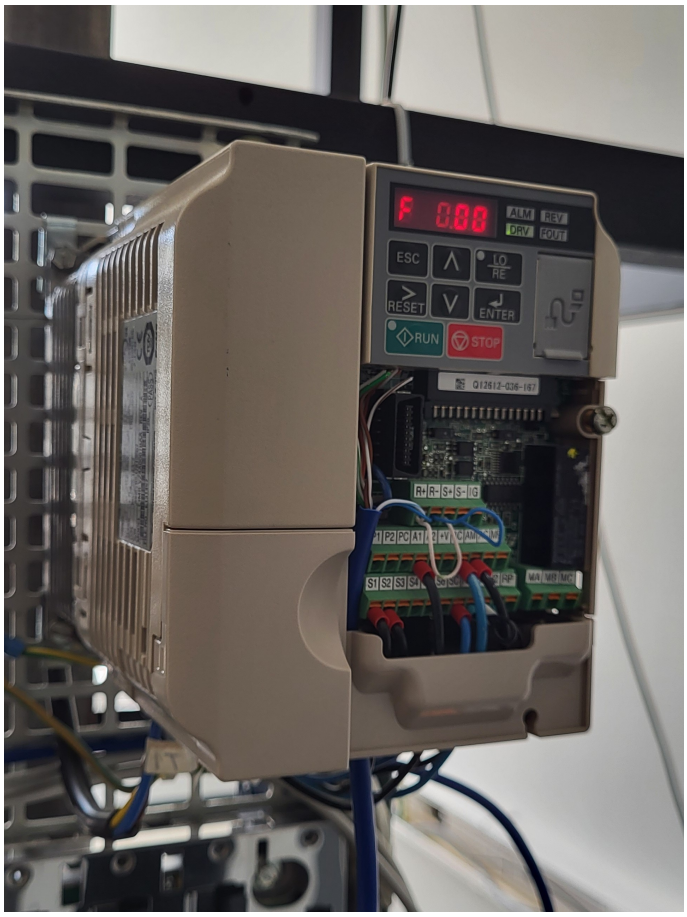


Figure 3.7: Variable-Frequency Drive (VFD).

  - **Programmable Logic Controller (PLC):**
    This PLC contains the control algorithm of the process. It sends the reference frequency to VFD which was previously calculated by the PID controller. This PLC also receives all the feedback of the system, as it has modules of digital and analog inputs and outputs. Overall, it has a rack composed by the power supply, the CPU, the modules of digital and analog inputs and outputs, the

communication module and the module with the web server, as shown in figure 3.8.

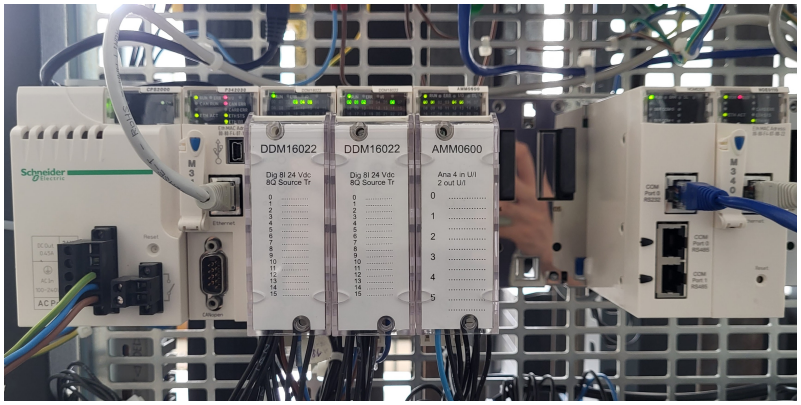


Figure 3.8: Programmable Logic Controller (PLC).

- **Command and Signaling Unit:**

  Attached to the structure is installed a command and signaling unit with 3 buttons for each floor, a button for the emergency stop and a LED light that lights up signaling the operation of the system. With this control unit, the operator is able to operate the elevator locally. In figure 3.9 the unit is shown and it is possible to identify, starting from the top: Signal Light, Floor 2 pushbutton (B2), Floor 1 pushbutton (B1), Floor 0 pushbutton (B0) and finally the Emergency Stop pushbutton (PE).

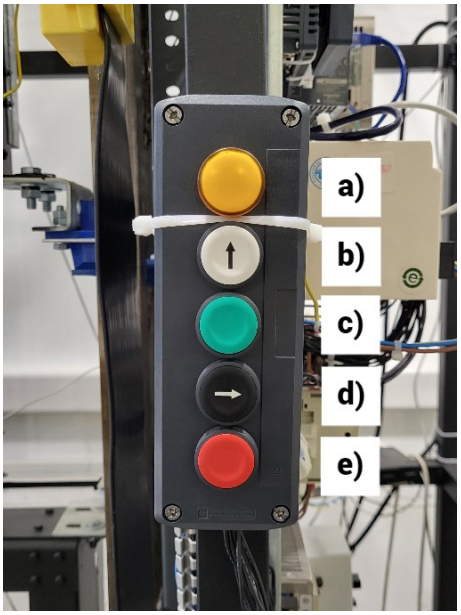


Figure 3.9: Command and signaling unit: a) Operation signal light, b) Floor 2 pushbutton (B2), c) Floor 1 pushbutton (B1), d) Floor 0 pushbutton (B0), e) Emergency stop pushbutton (PE).

3.2.2 Functional Specifications

In this section, the considerations for the implementation of the control algorithm will be reviewed, as there were some requirements in terms of operation and safety. These requirements, adapted from Santos 2022, were the following:

- Pressing the pushbuttons to select a floor, the elevator must move to the selected floor, as expected.
- Pressing the emergency stop pushbutton, the cabin should stop immediately.
- The motor brake must be normally engaged, except when there is an intent to move the elevator.
- The position must be controlled using a closed control loop.
- The system must be able to detect an error and cease the operation of the elevator if it cannot operate safely.
- In case of an ultrasonic sensor failure, the elevator should be able to change its operation mode to a less efficient open-loop controller.

The last two considerations are the foundations for the implementation, and proposal, of this work, given that this detection of errors and the transition of controllers are the focus of an upper layer of supervision.

3.3 Supervision Approach Proposed

3.3.1 Proposed Methodology

As mentioned before, the main objective of this dissertation is to implement a supervisory system capable of analyzing and detecting faults in order to overcome them, when possible, and thus maintain a stable operation of the process.

Regarding the detection and identification of the system faults, this system needs to have a Fault Tolerant Control (FTC) so that it is prone to be changed in the event of a fault or a failure. When a fault compromises the functioning or integrity of the structure, the system to take action: it can be either a critical failure, when the system needs to stop immediately, or a fault that can be bypassed, when there are redundant methods of control.

Taking these points into consideration, there were some important steps to follow during this work:

1. **Understand the functioning of the freight elevator:** As shown in the previous section, it was necessary to understand the operation of the elevator in order to have a better notion of the system, both in terms of hardware and software.

2. **Configure the hardware needed:** At this point, the hardware needed for this work was already included in the electrical schematic of the elevator, but not operational, so it needed some configuration that will be further explained.

3. **Enable exchange of data between Supervisor, Controller, and HMI:** From the moment that all the devices were on the same network, it was possible to receive feedback from the controller in the supervisor and in the visual interface, and then the opposite, sending commands over the network from the supervisor to the controller. The network and communication protocols used will also be described in the next section.

4. **Implement a "read-only" version of the supervisor:** with this first version it is possible to view the values of the system variables.

5. **Implement a version of the supervisor with input features:** This step is different from the previous in the sense that, besides the improvement of the supervision algorithm, the control algorithm needed also to be changed in order to receive external inputs, such as operator commands and reference signals to the VFD.

6. **Understand possible faults or failures:** In this step the main goal was to understand what could go wrong either on the elevator operation or the controller side. Depending on the type of fault, critical or not, the actions to be taken by the supervisor were defined, which are then transmitted to the controller.

7. **Enable the reconfiguration of the controller based on the type of fault:** In the case of faults that can be tolerated, the system needs to be redundant in such a way that its stability is not affected. Given this fact, and in the case of some faults, the controller needs to be adjusted to overcome the faulty signal of a sensor, for example. In this case, the supervisor will detect the difference in the behavior of the process and trigger some action. By having, and testing, several types of faults beforehand, it is possible to know what can be done to overcome each one of those faults, so the controller must be already configured for these situations.

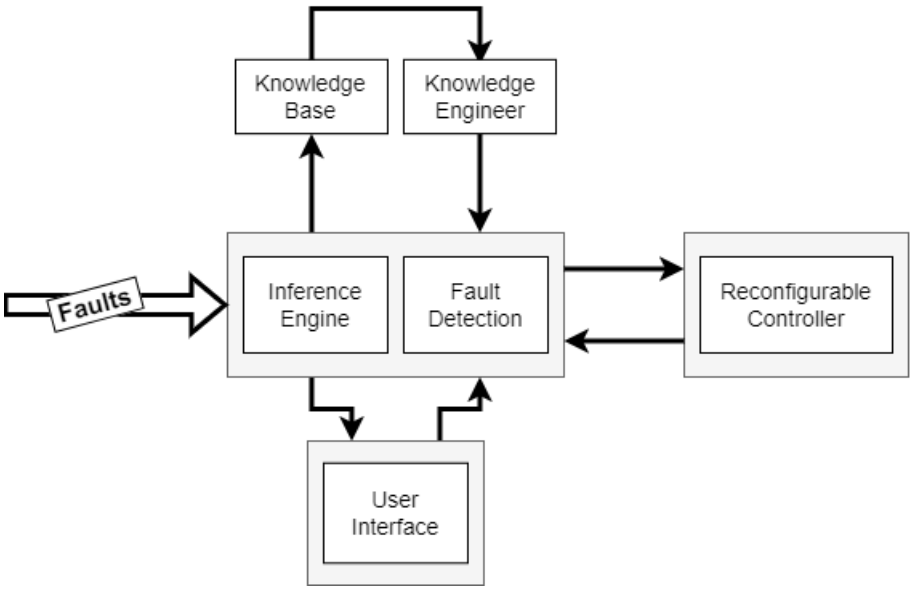


Figure 3.10: Architecture of a knowledge based approach for fault detection.

Therefore, it is fair to say that the fault detection system implemented has a knowledge-based approach, as shown in figure 3.10, given that the detection is based on rules that result in boolean values. The faulty scenarios were previously gathered so, based on some conditions, the supervisor can understand that one specific scenario is happening, triggering the bits required for the controller reconfiguration.

Given this, the proposed architectures for this work will be presented next so that it is possible to have an overview of the whole system and, moreover, everything with which this supervisor interacts.

3.3.2 Functional Specifications

In this section, the requirements for the supervisor algorithm are the reviewed, and these are the following:

- Receive inputs from the user in order to change parameters in the control of the process.

- Send the user configurations back to the controller.
- Receive information from the controller in order to monitor the status and the functioning of the system.
- Send the feedback from the controller and the process back to the user through a visual interface.
- Visually present the operation of the system to the user, allowing to interact with it.
- When monitoring the status of the process, detect faults and abnormalities in the operating procedure.
- Alert the user about the faults, abnormalities or other events occurring in the system.
- When possible, modify the control process in order to overcome faults while maintaining a stable operation.
- Stop the operation of the system when the supervisor detects a fault that can cause critical damage.
- Allow the interaction with the system via web browser with the same functionalities as in the local visual interface.

3.3.3 High Level Architecture (HLA)

As stated in the previous section, the high-level architectures proposed will be described, which allows to better understand the system as a whole.

In figure 3.11, it is possible to see the high-level architecture that was the base for this work. It is also possible to note the division between the two layers of control and supervision. As it was mentioned before, the development of the control algorithm of this system was the focus of a previous work, which means that now the focus is the supervision, shown in the upper layer.

The controller is the device that deals directly with the process plant, receiving and sending all the signals and references to control the system. Thus, the supervisor needs to interact directly with the controller since all the inputs are available there, which means that, if the supervisor needs to intervene with the system, the interface is always done through the controller. This communication between the two s is critical to the integrity of the supervisor as it needs the controller inputs to know the operation status and also needs to send information back to it in the case of a failure or a change in a setpoint made by an operator through the HMI, for example.

Still in the supervision layer, in figure 3.11, and besides the supervisor , it is also possible to identify the HMI that enables the user to interact with the process. It will be

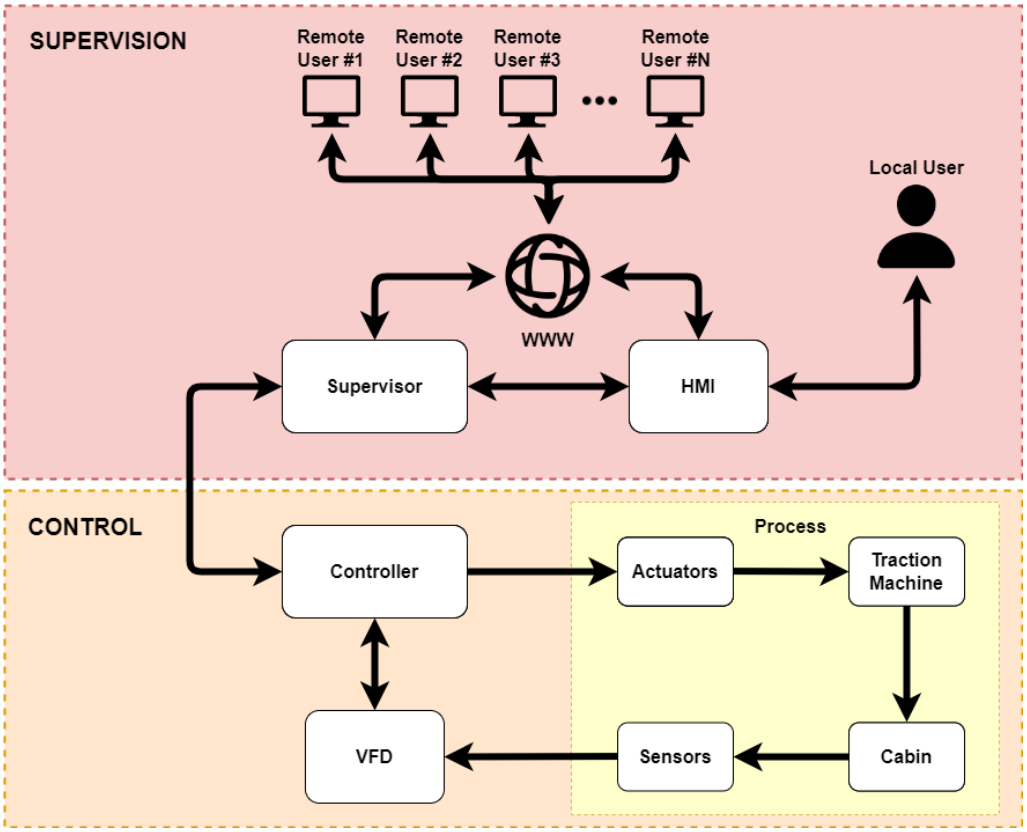


Figure 3.11: High Level Architecture (HLA) proposed.

possible to send the cabin to a desired position, change the setpoint of frequency of the motor or even re-calibrate the elevator if it is needed.

Both the supervisor and the HMI have web servers that, when enabled, can be accessed by the users and they have full access to the applications, i.e., read and write authorizations, depending on the user.

Diving into the architecture shown above, it is now possible to draw other architecture shown in figure 3.12, where the three main layers are described as process, control and supervision layers.

The process layer is composed of actuators, the elevator itself and sensors. The control layer interacts with the process layer by sending the frequency reference from the VFD to the actuator, in order for the elevator to move to the desired position. During the elevator movement, the VFD will receive the feedback of the actuator frequency to control the output. The frequency setpoint present in the VFD is calculated by the PID controller which, in its turn, has received feedback from the sensors about the actual position of the cabin. This closed control loop provides important information about the process, which allows the PID to obtain the frequency needed to reach a position while receiving the actual location.

Regarding the supervision layer, it will interact directly with the controller. The

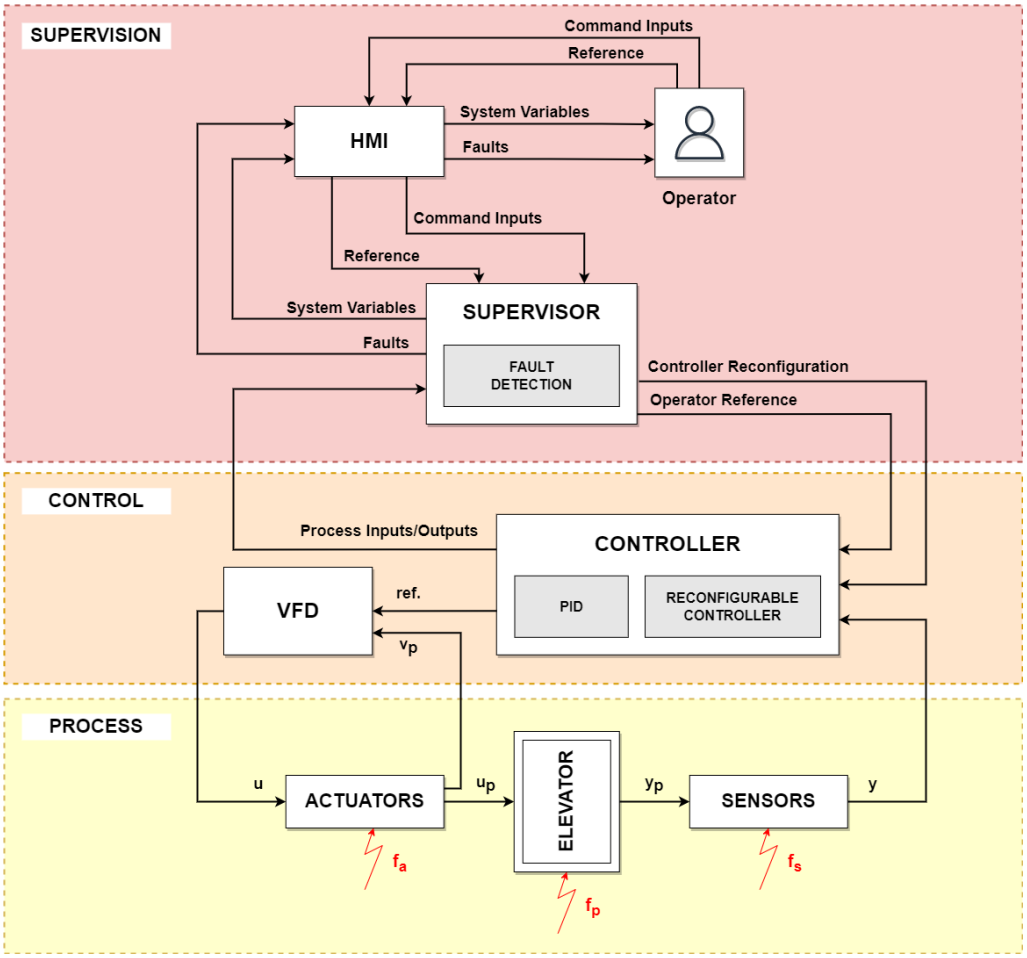


Figure 3.12: Layered High Level Architecture (HLA) with interactions between devices.

supervisor receives the actual state of the system variables, i.e., inputs and outputs, and the system of fault detection will identify possible causes of failure. In this case, and if it is a fault that can be tolerated, a signal for the reconfiguration of the controller will be returned to the controller, which will result in a different algorithm for the control of the process. For example, if the controller loses the signal of a sensor, or if it is abnormal, the supervisor detects this error, alarming the operator, but the elevator needs to maintain a stable performance, which means that, if before the controller needed the feedback of this sensor to control the position, now the controller needs to ignore the absence of this sensor and needs to work as if it was not present. Therefore, the conditions for controlling the elevator need to change. So, the supervisor is responsible for detecting the faults, communicating to the controller that occurred some faults, and where they occurred, and finally triggering the reconfiguration of the algorithm. For this, the controller needs to be programmed with this eventual reconfiguration in mind.

3.3.4 Technologies Implemented

In this section the technologies implemented will be described, as well as their functionalities and their role in the process of supervision.

- **Controller Schneider M262**

  The controller, shown in figure 3.13, is the PLC Modicon M262 developed by Schneider Electric, and it has many powerful resources.

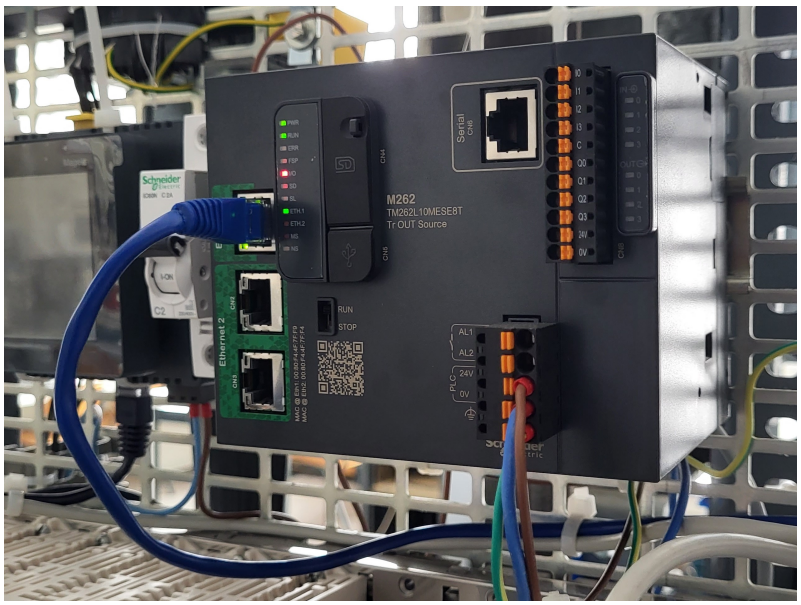


Figure 3.13: PLC Modicon M262.

First of all, the M262 supports all five automation programming languages, i.e., Instruction List (IL), Structured Text (ST), Function Block Diagram (FBD), Sequential Function Chart (SFC) and Ladder Diagram (LD), which makes this controller flexible for every automation network.

In the development of the application, two of these languages were used, specifically ST and LD. The majority of the application was developed in ST as it takes up less storage space and has a higher processing speed comparing with graphical languages, such as LD.

The controller has various communication ports such as: one RJ45 Serial Line port, one USB programming port, one RJ45 Ethernet port and one RJ45 Ethernet switch with a double port. Besides communication ports, the controller has also four fast digital input ports and 4 fast digital output ports. The PLC supports other modules for communication or inputs but in this case all the information is received through the Ethernet port, so no additional module is needed.

Regarding the memory, the PLC has 256MB of RAM, 32MB of which are available for the application execution, 1GB of integrated Flash memory to save a backup of

the program.

The average execution time for 1000 instructions is 5 microseconds and it takes an average of 500 microseconds to retain 1000 variables, either read or write.

- **HMI Schneider Magelis STU655**

  The Human-Machine Interface used was the Schneider Magelis STU655, as shown in figure 3.14, which allows the operator to have functionalities such as visualization and interaction with the process.

  Regarding the memory of this device, it has 32MB of Flash memory for the application, 64kB of FRAM memory for data backup and 64MB of DRAM memory for executing the application.

  This HMI has several communication interfaces, such as: RJ45 port for Serial link connection, USB type A port for connecting peripherals or a USB memory stick, USB type mini-B for programming and application transfer and RJ45 connector for Ethernet TCP/IP.

Figure 3.14: HMI Schneider Magelis STU655.

3.4 System Supervision Development

3.4.1 Communications Network

In figure 3.15 it is possible to observe the hierarchic architecture designed for this system with the static IP addresses associated to each device. On the bottom, the plant control layer is represented by the PLC that controls the logic of the process and the VFD that controls the speed of the cabin. On the middle layer there is a supervision layer with the PLC responsible for monitoring and the HMI that allows the interface between the user and the process. On the top layer is represented the production control, where all the users, who are part of operation or maintenance teams, can be connected to the network, and the servers, having full control over the system. In table 3.1, the assigned IP addresses for the relevant devices are listed.

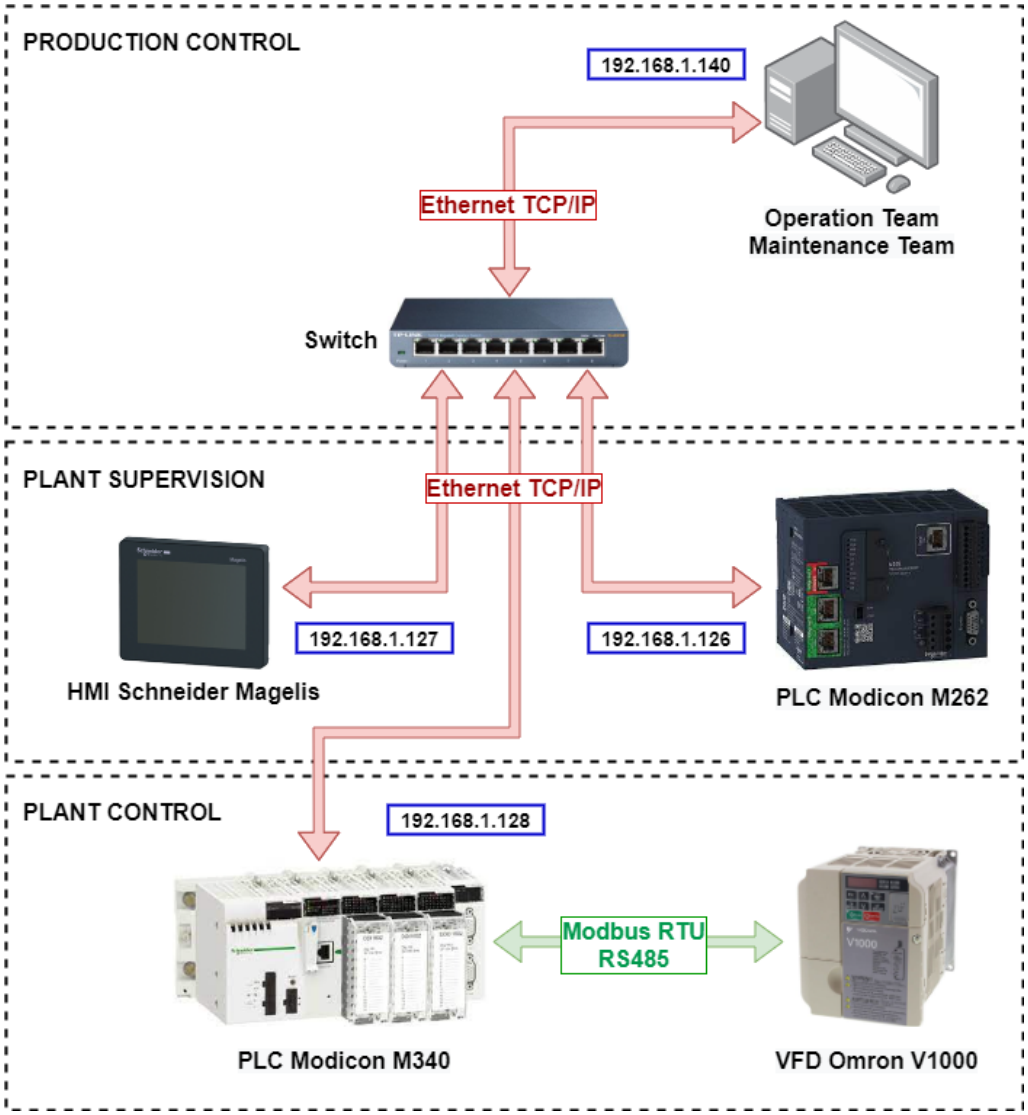


Figure 3.15: Network of industrial communication protocols implemented.

Device	IP address
PLC M262	*192.168.1.126*
HMI STU655	*192.168.1.127*
PLC M340	*192.168.1.128*
User machine	*192.168.1.140*
Gateway: *192.168.1.1*	
Subnet Mask: *255.255.255.0*	

Table 3.1: List of assigned IP addresses.

In order to have all the devices connected to the same network, and to avoid installing extra communication modules, that would increase the cost of the project, the most logical choice was to have an Ethernet Bus, materialized as a switch, where all the devices were connected and accessible through an IP address.
Therefore, a local network was created in order to connect these devices. This network is also a first step in the implementation of the global network of the automation laboratory, given that one ambition, in the near future, is to have all the devices of this room connected to the same network so that the processes can be accessed remotely by both teachers and students. Once it is possible for teachers and students to remotely transfer their applications to the processes and also visualize what is happening in real-time, it will improve significantly the flexibility of the classes.

3.4.2 Hardware Configuration

Regarding the configuration of the hardware, this section presents the parameters taken into consideration when configuring the devices.

- **PLC M340**

  The configuration, and programming, of the Controller M340, is done through the tool Unity Pro XL. Figure 3.16 demonstrates the controller configuration in Unity Pro XL, which replicates the assembly done on the physical controller.

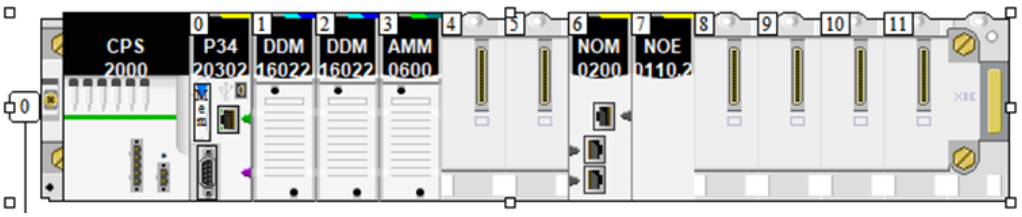


Figure 3.16: Rack configuration of M340 PLC in Unity Pro XL.

In figure 3.16 it is possible to identify, from left to right:

  - Power supply CPS 200.

- Slot 0: CPU P3420302.
- Slots 1–2: Digital Inputs and Outputs module DDM16022.
- Slot 3: Analog Input and Output module AMM0600.
- Slots 4-5: Empty.
- Slot 6: Serial link module NOM0200.
- Slot 7: Ethernet TCP/IP network module NOE0110.
- Slots 8-11: Empty.

This controller is connected to the network by Ethernet TCP/IP and by Modbus RTU to the VFD. The parameters for both connections were already configured, but the IP address for the Ethernet connection needed to be updated, which requires a rebuild of the project. The configuration is associated with the Ethernet port of the CPU and it is shown in figure 3.17.

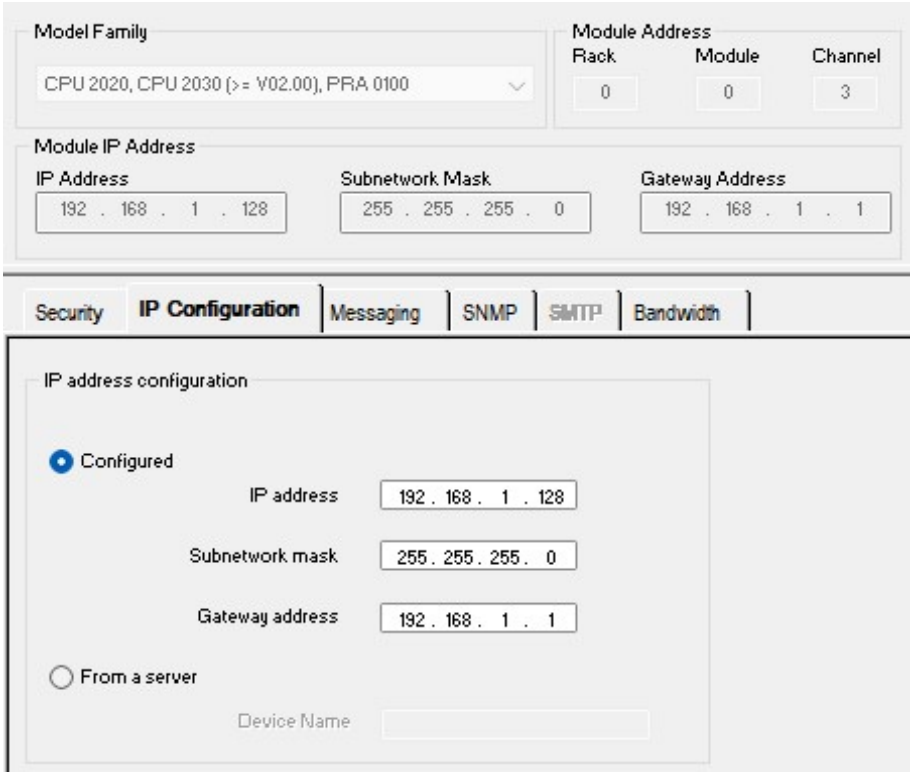


Figure 3.17: CPU Ethernet port configuration.

- **PLC M262**

  The M262 configuration is made in the new software platform of Schneider known as Ecostruxure Machine Expert (V2.0). In this program the device was added to the project, as shown in figure 3.18, which is the Logic Controller M262 with reference TM262L10MESE8T.

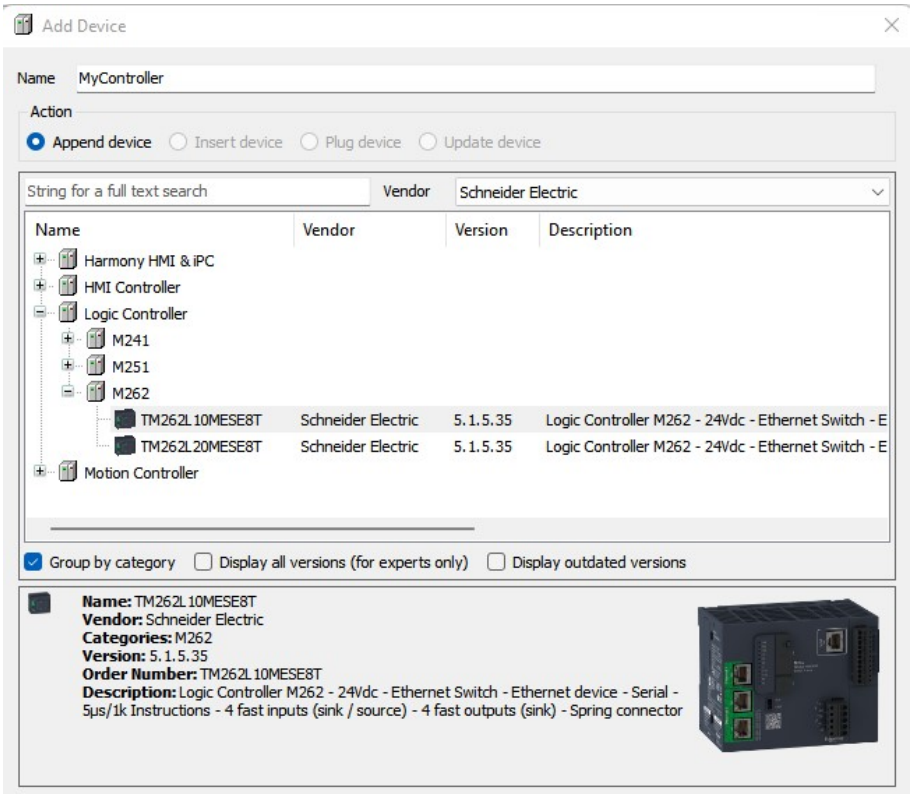


Figure 3.18: M262 PLC configuration in Ecostruxure Machine Expert.

Given that the controller has two Ethernet channels, only the Ethernet port 1 was configured as shown in figure 3.19.

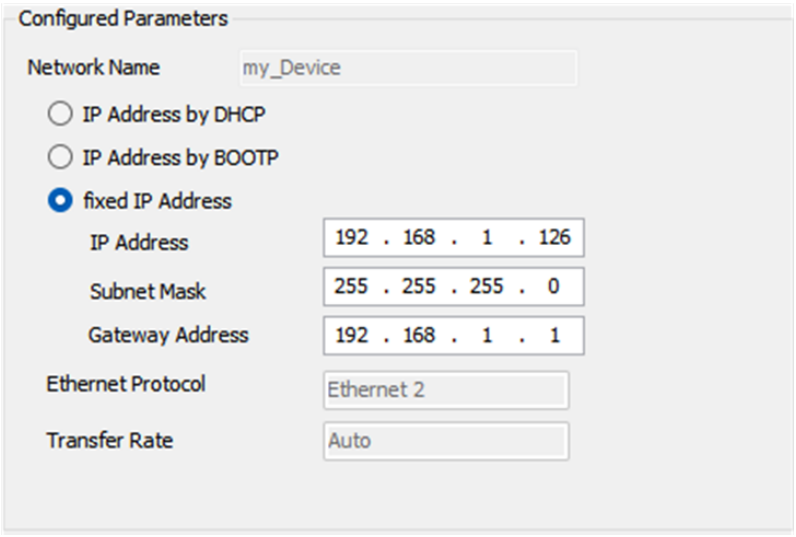


Figure 3.19: M262 PLC ETH1 port configuration.

As this PLC is responsible for the supervision of the system, it needs to retrieve information from all sources in order to check the functioning of the process and determine changes in the control variables or even possible causes of concern. Given

this, this device needs to scan variables from two other devices: the PLC that controls the process and the HMI which can have inputs from the user.

These two devices are configured as slaves from the point of view of the M262 since this is the one retrieving information, which makes the M262 the master. This way, they can exchange data by using a request-reply mechanism, in which the master control the flow of information between itself and the slaves. Both devices were configured as Modbus TCP slaves with an I/O Scanner. The configuration of the slaves is shown in figure 3.20 and figure 3.21.

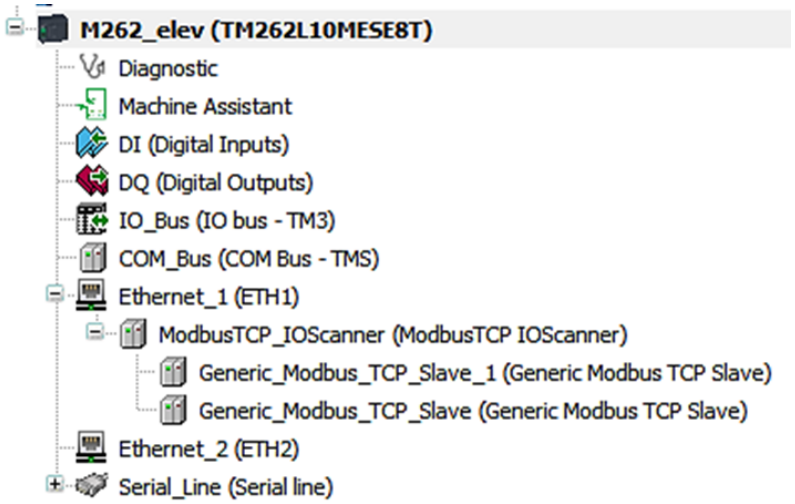


Figure 3.20: Configuration of I/O Manager and Modbus TCP slaves.

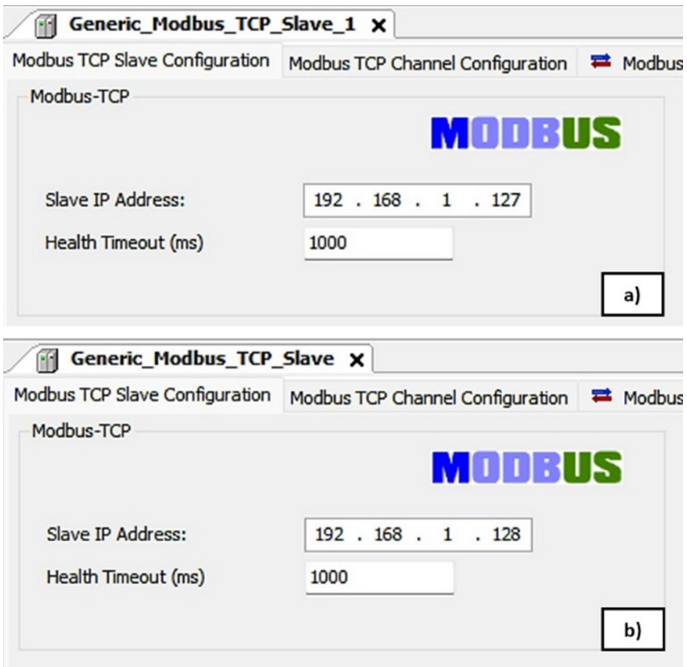


Figure 3.21: Modbus TCP slaves: a) slave HMI, b) slave M340 PLC.

- **HMI STU655**

  Regarding the HMI, it was configured using another platform from Schneider called Vijeo Designer. This platform allows to configure the visual interfaces and also to deploy the running application to a web server where the device can be accessed remotely, through its IP address. The main configuration of the HMI can be seen in figure 3.22.

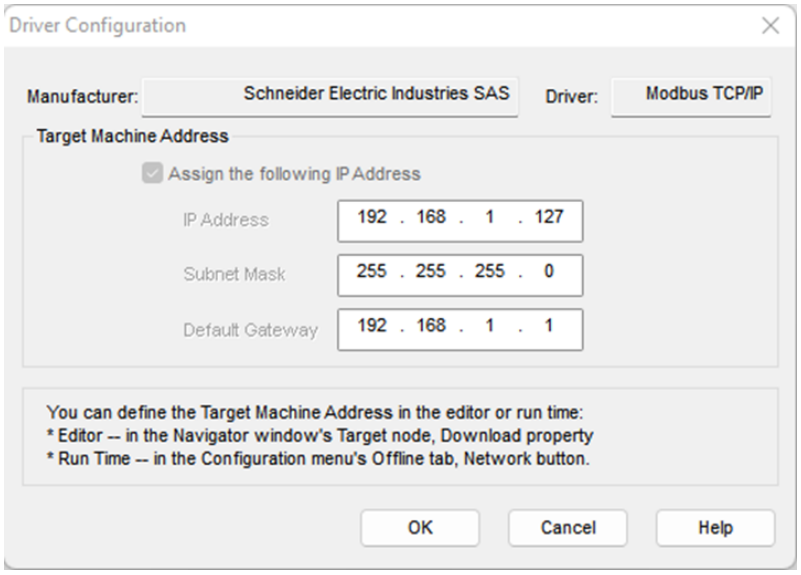


Figure 3.22: HMI STU655 configuration in Vijeo Designer.

This HMI receives and sends information directly to the supervisor PLC, which means that this device needed to be also configured as a connection. In the HMI I/O Manager, the M262 was configured as Modbus TCP/IP connection, as shown in figure 3.23.

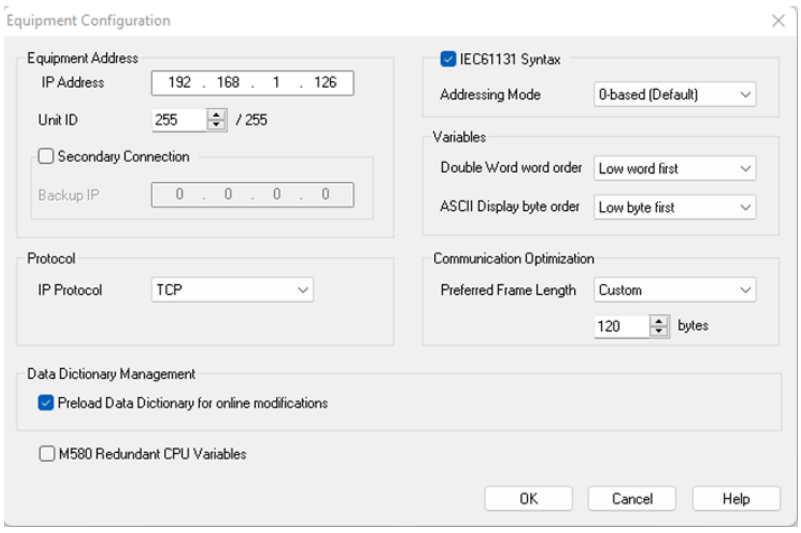


Figure 3.23: Configuration of connection between the HMI and the M262 PLC.

Resuming, the interactions between the three devices are represented in figure 3.24. The M262 retrieves information through a Modbus TCP/IP, which means that it is basically the Modbus RTU protocol, that is the information language and formatting, running on Ethernet. The TCP/IP is what facilitates the data transmission by granting the correct delivering of the data packets (TCP) to the correct addresses (IP).

The Supervisor M262 performs an important role, considering that it is the link between the control of the process and the user. It gathers and analyzes the data that is vital for a well-functioning system and presents this data to be monitored either in the visual interface or via the web.

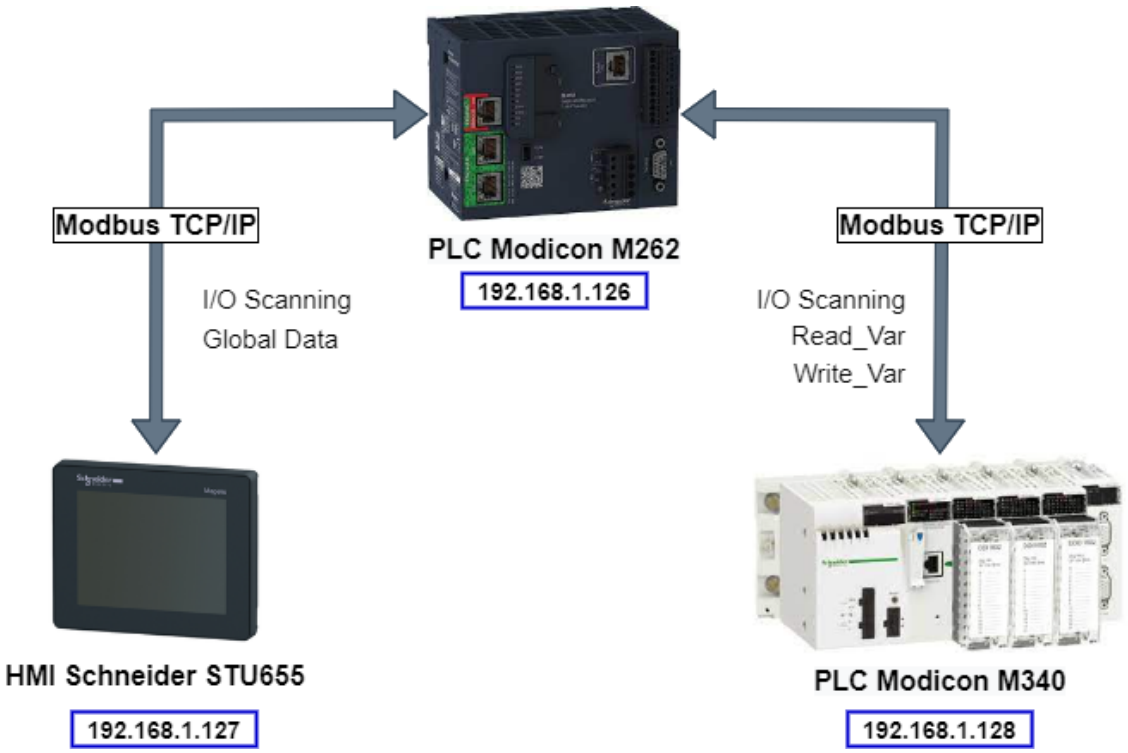


Figure 3.24: Architecture of Modbus communications. M262 PLC as the master and M340 PLC and HMI as the slaves.

3.4.3 Process Supervision

In order to exchange data between the two PLC, the variables were gathered into arrays of words to make the data transmission easier. That being said, in the controller three types of words were created to put together all the values from the same type, so there was two words with Boolean, one array with integer and one array with real values.

To transmit information between the controllers, the interface is made through READ_VAR and WRITE_VAR blocks of logic. The M340 allocates the values of the variables in words or arrays and the M262 accesses those locations of memory. Since the M262 PLC acts as the master, while the M340 is a slave, then it is only the M262 that uses

READ_VAR or WRITE_VAR blocks to control the flow of information. If the M340 has the information available in the correct addresses, the M262 will receive that information making use of this block. In the opposite case, when the M262 needs to write information in the M340, it is again the M262 making use of the WRITE_VAR block to send information to a specific address in the M340. In figure 3.25 there is an example of a READ_VAR and WRITE_VAR from the M262 application.

```
ADDM_0(
    AddrTable:= endereco,
    Execute:= out_blink,
    Addr:= '3{192.168.1.128}',
    Done=> addm_done,
    Error=> error1,
    CommError=> comerror);


READ_VAR_0(
    Execute:= addm_done,
    Abort:= FALSE,
    Addr:= endereco,
    Timeout:= readvar_timeout,
    Done=> readvar_done,
    Busy=> busy,
    Aborted=> ,
    Error=> error_1,
    CommError=> commerror_1,
    OperError=> ,
    ObjType:= ObjectType.MW,
    FirstObj:= 30,
    Quantity:= 4,
    Buffer:= ADR (buffer_read));
```

```
ADDM_0(
    AddrTable:= endereco,
    Execute:= out_blink,
    Addr:= '3{192.168.1.128}',
    Done=> addm_done,
    Error=> error1,
    CommError=> comerror);

WRITE_VAR_0(
    Execute:= addm_done,
    Abort:= ,
    Addr:= endereco,
    Timeout:= ,
    Done=> done_write,
    Busy=> busy_write,
    Aborted=> aborted_write,
    Error=> error_write,
    CommError=> commerror_write,
    OperError=> opererror_write,
    ObjType:= ObjectType.MW,
    FirstObj:= 140,
    Quantity:= 8,
    Buffer:= ADR(GVL.send_m340));
```

Figure 3.25: Examples of READ_VAR and WRITE_VAR in Machine Expert.

On the left it is shown one of the executions of READ_VAR in the program. The function ADDM converts a string into an address where the *"3"* means that the message is received through Ethernet from IP *"192.168.1.128"* which is the M340 PLC. The READ_VAR function gives feedback about the state of the connection, in case if it was aborted or if there was an error. The last parameters mean that the controller wants to read 4 variables with the %MW type, starting on %MW30 of the M340 PLC. The values read will then be stored in the address of the buffer *"buffer_read"* already in the M262.

On the right, and almost similar, it is shown one example of the WRITE_VAR function. The parameters are almost the same, except the last ones where it has the information to write. In this case, it is stated that the M262 PLC needs to write the information stored in the address of its global variable (GVL) *"send_m340"*, which will write 8 values with the type %MW starting in the address %MW140 of the M340 PLC.

3.4.3.1 Data Logging

One important feature for the supervision of processes is the possibility of recording values of variables or states in order to allow an operator to supervise these values later. In this case, the supervisor PLC has a functionality implemented that enables the recording of files with the values needed.

After the logging of the values, the supervisor can extract these files from the embedded web server in the M262 supervisor. This is an important functionality in the event of a maintenance intervention, for instance. The technician can start the logging of variables in the HMI and, in the text file, it records the exact time of the reading and the values from the variables wanted, such as values from sensors or alarms, for example.

Figure 3.26 shows an example of formatting for a file logged from this system. The files can be accessible through the web or the Machine Expert program.

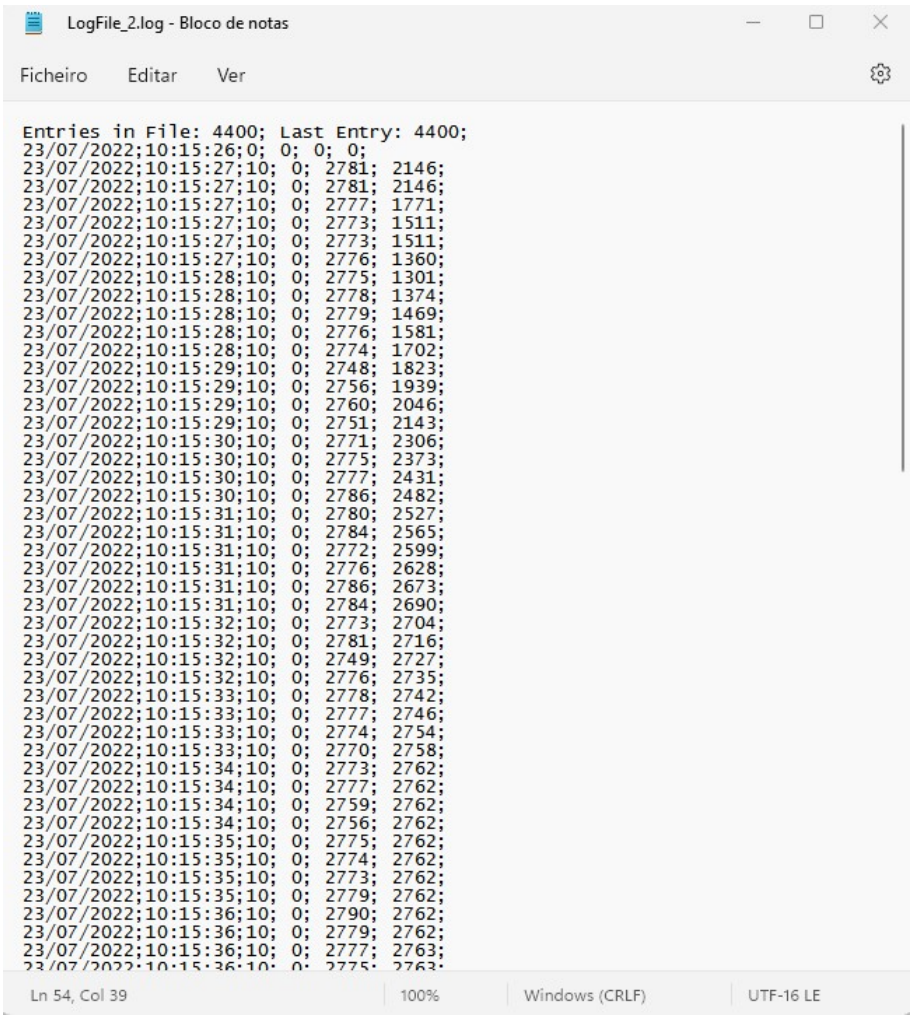


Figure 3.26: Example of a log file recorded in the M262 PLC.

3.4.4 Human-Machine Interface (HMI)

One important aspect of supervision systems is the ability to visualize and receive warnings about the functioning of the system.

The goal for this interface is to have a simple and intuitive design, while having all the essential functions to the user. This way, the panel gives the operator a reliable tool to fully control the process

The HMI application can be divided into these sections:

- For security reasons, the HMI has an authentication window to only allow the access to authorized users.
- Main panel where the general operation of the elevator can be viewed.
- Control panel where the commands and setpoints can be changed and triggered.
- Alarms panel where there is a log of the events and failures of the system.
- Trend graphs panel where the variables are logged so it is easier to visualize them as a function of time.

In figure 3.27 it is possible to see the authentication window, where the user must insert his credentials in order to access the HMI.

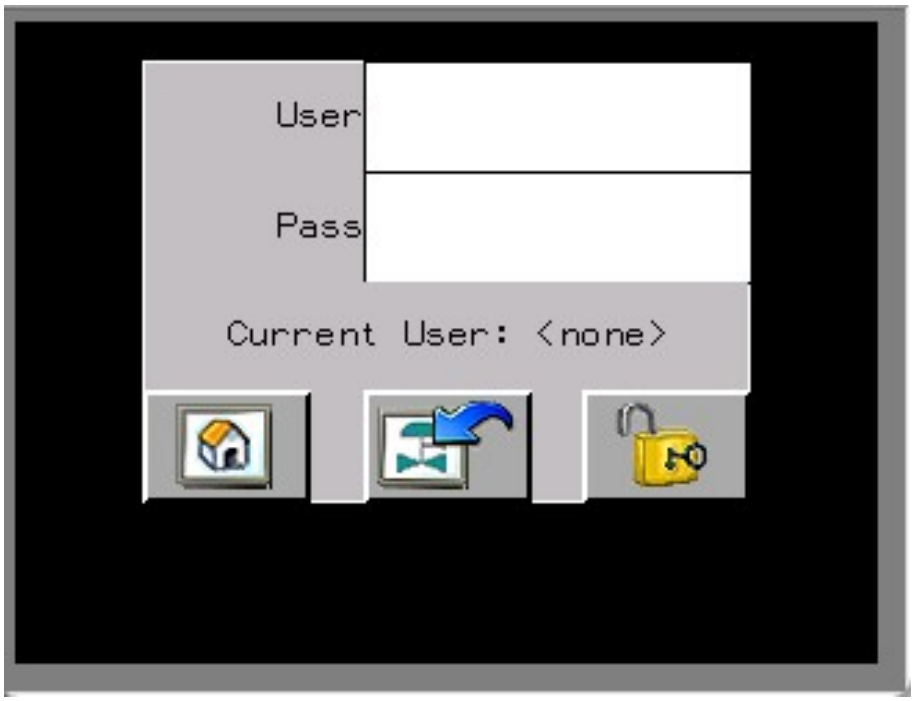


Figure 3.27: Login window to access the HMI.

The main panel, shown in figure 3.28, illustrates:

- The elevator course, as well as relative and absolute position values of the cabin.
- The direction of the elevator movement, indicating the movement up or down.
- The distribution of floor and end-of-stroke sensors along the path and they indicate when the cabin is in each of these positions.
- The perspective of the elevator seen from above, where the reed-switches are represented and also indicate their state.

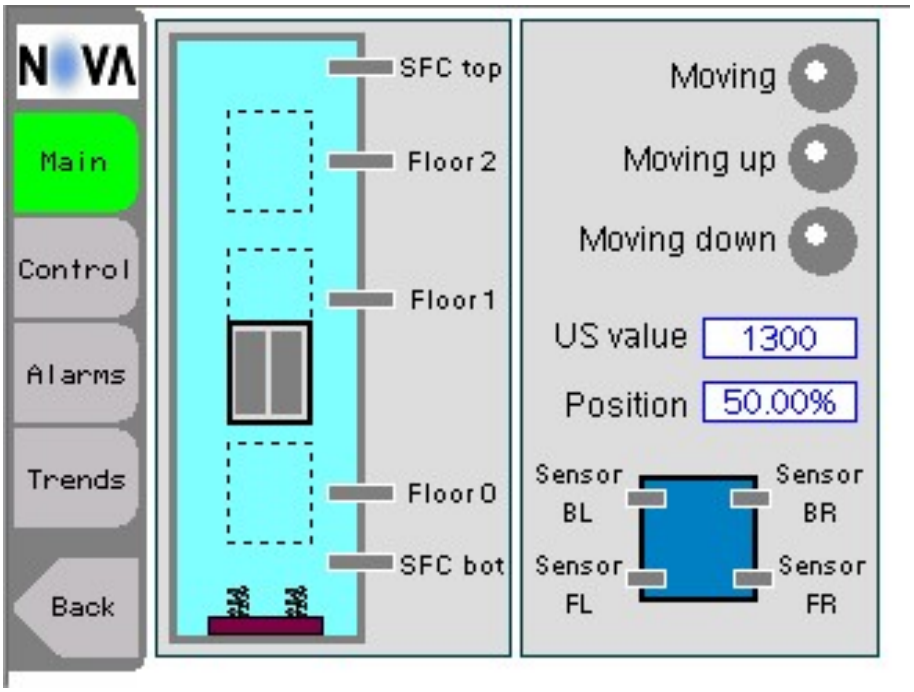


Figure 3.28: Main panel from the HMI.

The control panel, shown in figure 3.29, illustrates:

- A replication of the command unit with the command buttons for sending the elevator to each floor, the emergency stop button and the signal light that indicates the elevator movement.
- The button to start the calibration of the elevator.
- The indication of either the system is calibrated or not calibrated (when it occurs an error).
- The feedback of the sensors along the path and also the indication of the emergency stop.
- The field to enter the desired position to send the elevator, which can be used, for example, in case of maintenance.

- The field to enter the desired maximum VFD frequency in order to limit the maximum speed that the cabin can reach.
- Two buttons that start the logging of files, one with the important variables of the process, such as sensors and position, and one with the error variables of the system.

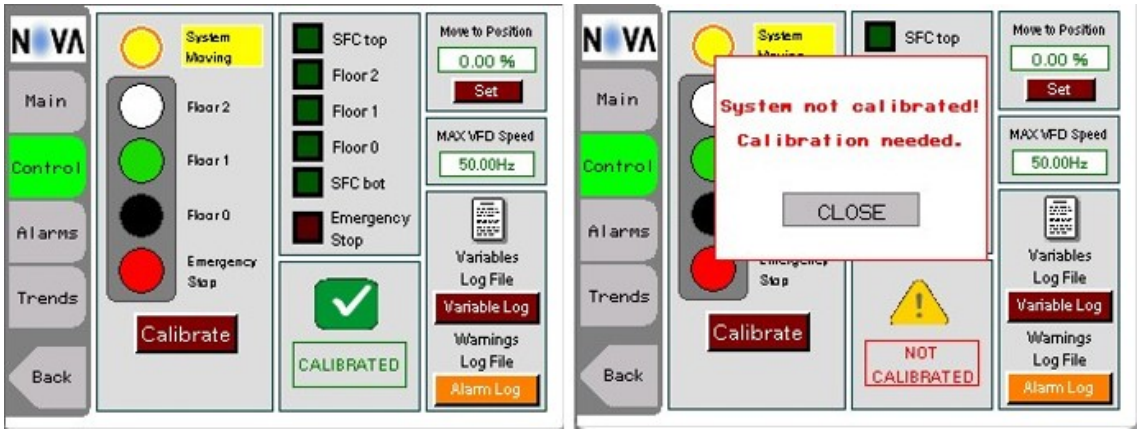


Figure 3.29: Control panel from the HMI, on the left. On the right appears an example of a popup window showing an incorrect action by the user.

The alarm panel, shown in figure 3.30, illustrates:

- The table with a log of all the events and events of the system. When an alarm is set, it has the color red and when the error resets, a similar line appears in green.
- An alarm banner that appears when the system triggers an error. The banner disappears when the error is rectified.

The alarm panel, shown in figure 3.31, has four choices of trend graphs:

- A relation between the current VFD speed with the target speed and the maximum speed.
- A relation between the actual position, in percentage, and the positions of the three floors.
- A relation between the actual absolute value from the ultrasonic sensor and the absolute values from the three floors.
- A relation between the digital floor sensors and the magnetic sensors on top of the cabin.

In figure 3.31 it is also possible to see an example of a graph with the relation between the relative position of the elevator and positions of each floor.

Figure 3.30: Alarm panel from the HMI.

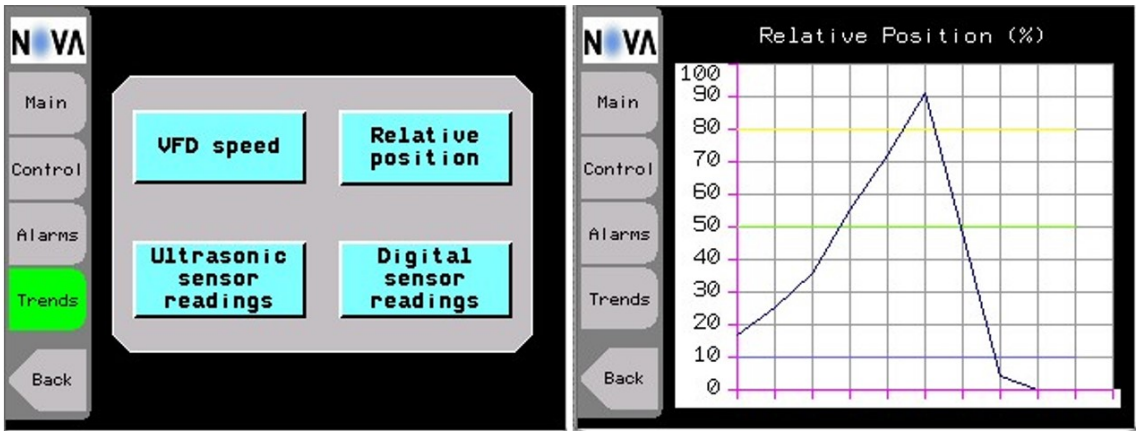


Figure 3.31: Trends panel from the HMI, on the left. On the left there is an example of one of the trend graphs.

3.4.4.1 Web server

All the three main devices, the controller PLC M340, the supervisor PLC M262 and the HMI STU655, have embedded web servers that allow the users to interact with them and receive vital information through the web browser.

In Vijeo Designer, the tool from Schneider to program the HMI, it is possible to configure the web server for the user to access the interface. In the web server, the user has the same capabilities as in the physical HMI while working remotely.

By accessing the IP address of the HMI device, in this case *http://192.168.1.127*, the Web server shows the interface illustrated in figure below. The replication of the HMI can be accessed through the tab Web Gate.

In figure 3.32 it is possible to observe the main panel when the IP address from the HMI is accessed. This page has tabs enabled for monitoring, diagnostics and maintenance.

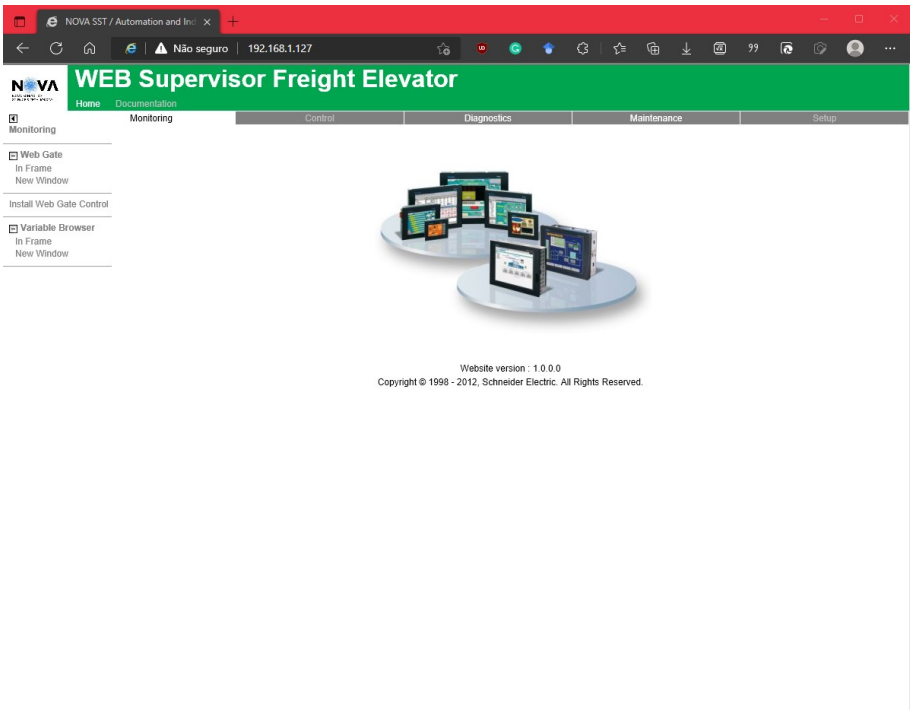


Figure 3.32: Main panel from the HMI Web server.

In the Web Gate tab is where the HMI can be accessed and where the user can interact with the system. Depending on the level of security attributed to the Web server, the user can send inputs to the process and change parameters as if it was done in the local HMI. This feature is a major improvement for this system since it will be accessed by remote users, such as professors or students.

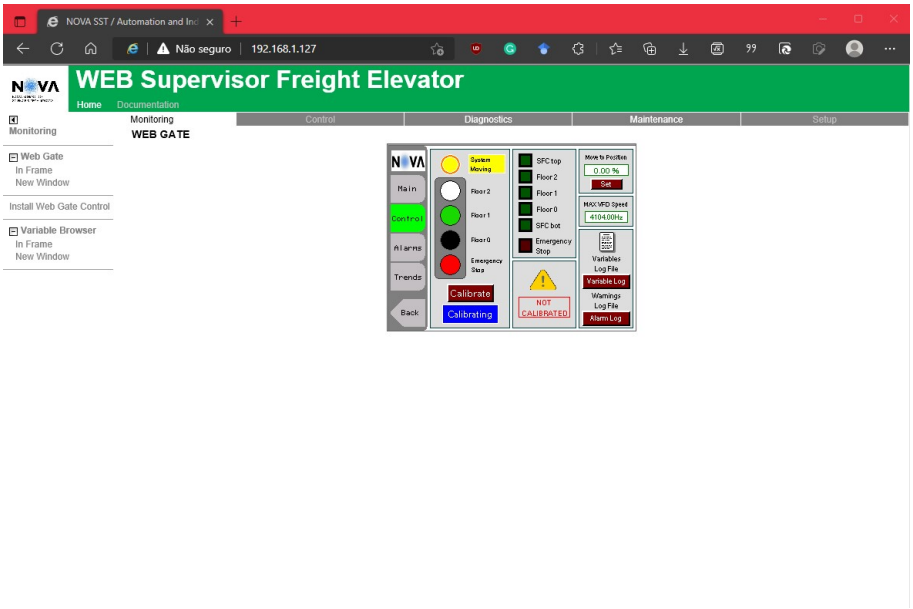


Figure 3.33: Web Gate interface where the HMI can be interacted with.

Regarding the M262 PLC, it has also a Web server with various functionalities to monitor and debug what is happening in the controller. The Web server can be also accessed through its IP address in the browser, i.e., *http://192.168.1.126*. The main interface can be seen in figure 3.34.

The Web server has many tabs such as Monitoring, Diagnostics, Maintenance and Machine Assistant. One important feature is the Diagnostics that allow to, for example, see the state of the Ethernet connection or even with a specific device. For example, it is possible to enter an IP address from a device and see if the connection is OK, which is exemplified in figure 3.35 with the IP address from the controller M340.

Another feature that is useful is the monitoring of variables in real time, given that, in the Monitoring tab, the Web server allows the user to enter a variable and see it represented in the osciloscope, as seen in figure 3.36. This can be an important and easy way of debugging malfunctions remotely.

Figure 3.34: Main panel from the M262 PLC Web server.

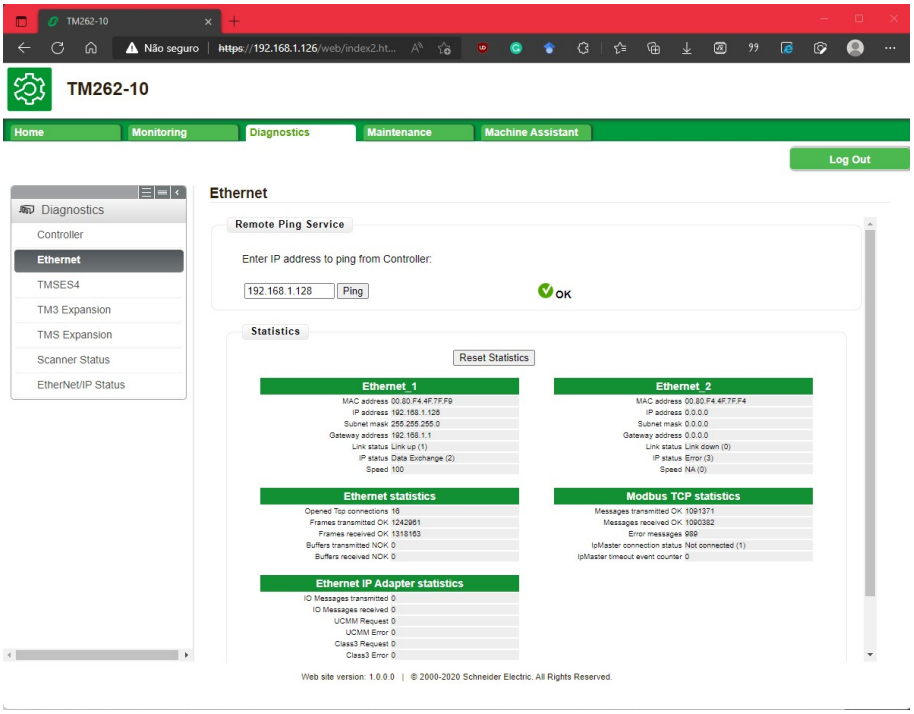


Figure 3.35: Ethernet tab from the M262 PLC Web server.

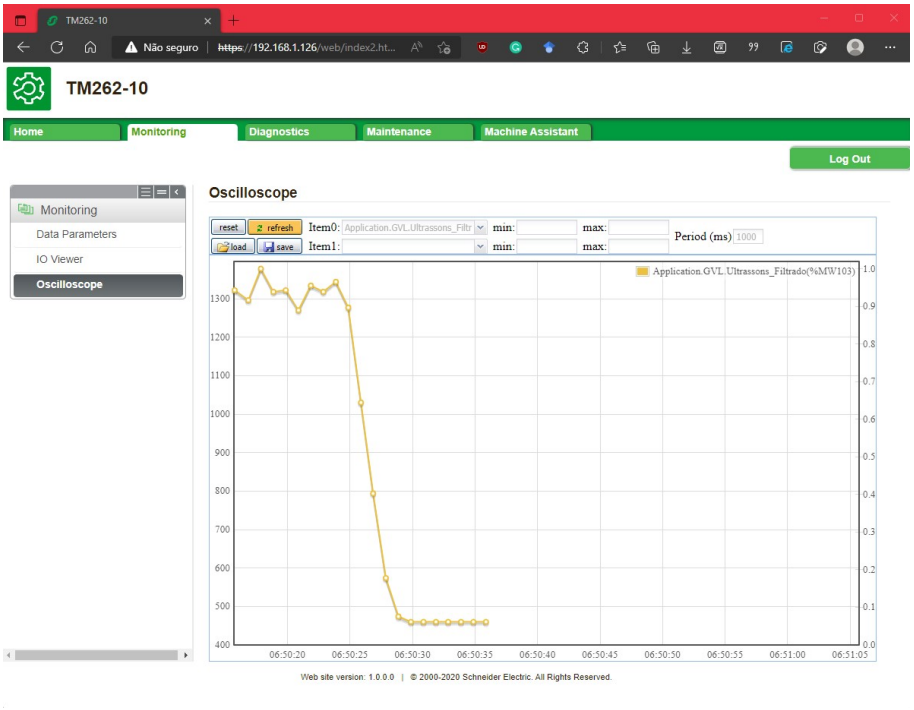


Figure 3.36: Osciloscope feature from the M262 PLC Web server.

3.4.5 Faults and Failures Detection and Diagnosis

Related to supervision of processes, there is a very important function that is the detection of faults and failures in the process. The supervisor M262 is the responsible for this detection of errors given that it gathers the essential information from the devices, resulting in a broad vision of the system.

The error detection is based on a knowledge database that contains the failure scenarios and the conditions required to trigger each scenario. Based on this data, it is possible for the PLC to verify each condition and detect faults or failures, by using an inference mechanism. The eventual errors will then trigger bits related to each failure to notify both the operator and the controller PLC.

Given this, the errors were divided in two categories in order to divide the actions to take for each category:

- Faults: errors that can be corrected, or are even temporary, and do not require immediate intervention in the system.

- Failures: errors that require immediate action in the system, otherwise it may result in damage to the equipment or even injuries to people involved.

Based on these two categories, the possible faults and failures to happen in the system are described in table 3.2.

Faults	Errors in the ultrasonic sensor readings.
	Missing signal from floor 0 sensor (P0).
	Missing signal from floor 1 sensor (P1).
	Missing signal from floor 2 sensor (P2).
	Missing signal from ampoule FR.
	Missing signal from ampoule FL.
	Missing signal from ampoule BR.
	Missing signal from ampoule BL.
Failures	SFC top reached, out of bounds.
	SFC bot reached, out of bounds.
	Motor working with brake engaged.
	Elevator not running and brake is detached.
	Emergency stop activated.

Table 3.2: List of faults and failures.

Regarding faults, there is no action regarding the missing signals from floor sensors or ampoules since that could be a temporary error and it could only need some physical adjustment that caused the sensor to be misreading. In this case, the operator is still notified of the error so that the maintenance of this sensor can be planned. The equipment does not need to stop for an immediate repair since it will not affect the performance of the system.

Figure 3.37 shows the expected states of the ampoule sensors in each state, i.e., in each floor and between floors. The cabin can be seen from above and, as stated before, the left ampoules are mono-stable and the left ones are bi-stable. The ampoules on the left (BL – Back Left - and FL – Front Left) signal when the cabin enters a floor, so the floor sensor will be active too. The combinations of the right ampoules (BR – Back Right - and FR – Front Right) signal in which zone the cabin is.

When the operator starts the process of calibrating the elevator, the controller registers all the values of the ultrasonic sensor equivalent to each floor, which means that it is possible to verify if the sensor readings are deviating from the initial values registered.
In the case of the ultrasonic sensor fault, when its readings are deviated from the expected values, it triggers the reconfiguration of the controller. This reconfiguration deactivates the closed-loop controller, based on the ultrasonic sensor values for the position feedback, and activates an open-loop controller that is based on the signal of the ampoules and floor sensors. This way, the operation of the elevator maintains its usability but lacks in feedback to the user, as the only feedback of actual position is the signal from the ultrasonic sensor that, in this case, may be defected.

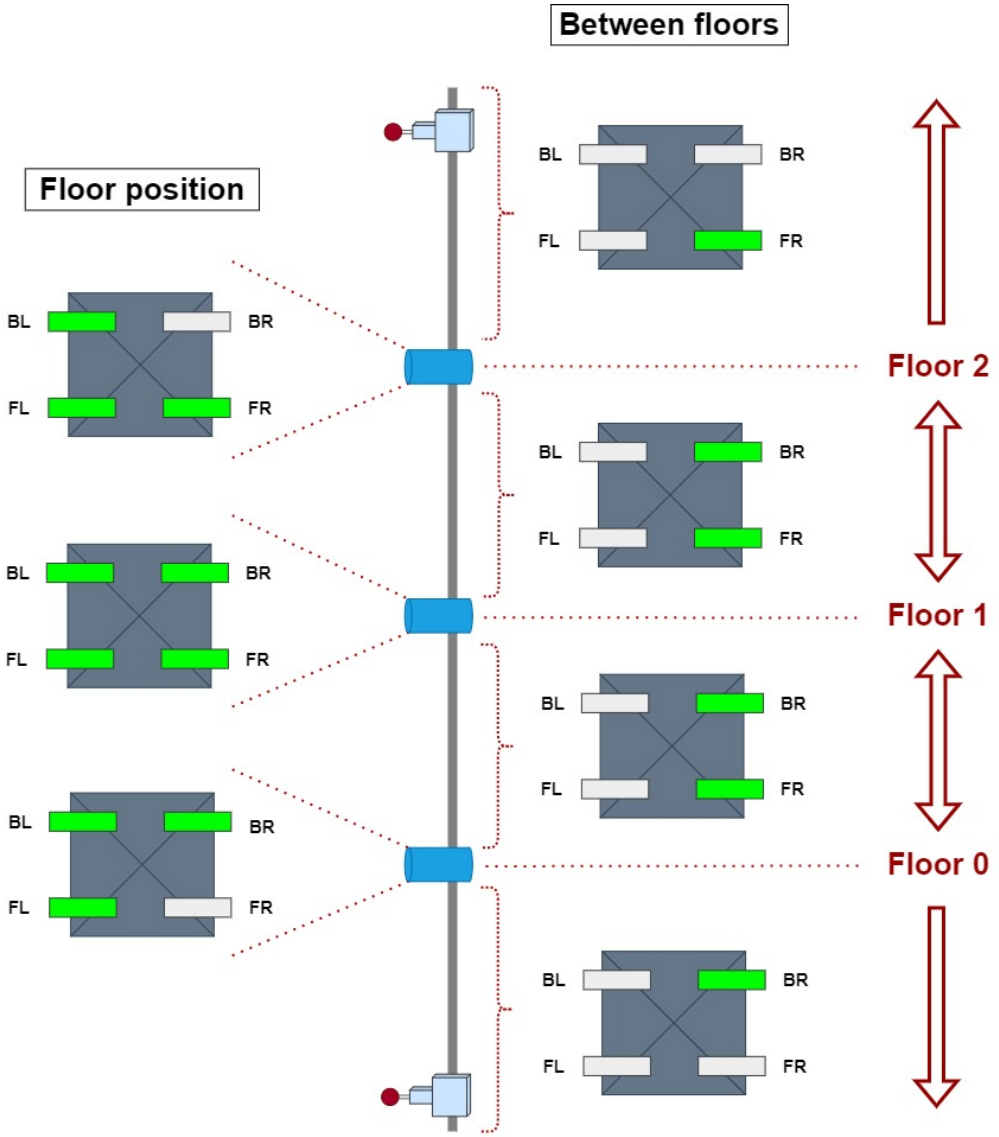


Figure 3.37: Expected states of magnetic ampoules in floors and in between floors.

Regarding failures, the scenarios in table 3.2 may result in component damage which, for safety reasons, triggers an immediate stop in the system.

In the case of the SFC (End-of-stroke) sensors actuation, it means that the elevator went out of bounds, which is not permissible by the control algorithm. If this happens, then the system must stop and inform the operator for the need of an urgent intervention, so the system will not resume its operation until the error is acknowledged.

Regarding the actuators, there are two possible failures that may result in stress and overload of components. When the traction machine is rotating in order to move the cabin, the brake must be disengaged, otherwise the friction created between the brake and the motor, besides the weight of the cabin, will result in overheating and possible tear of these two components. By contrast, when the elevator receives an order to stop, the brake must engage immediately to stop the movement of the cabin. In both these cases, (*motor on* +

brake on, or *motor off + brake off*), the system should stop and warn the operator that an inspection is necessary.

Lastly, if the emergency stop button is pressed, the system also gives a warning in order to log this event for future analysis. Even though it is the operator that activates the emergency stop of the system, a manually stop should always be registered because it is an event that indicates a malfunction.

In figure it is possible to see the fluxogram of the logic behind the detection of faults in sensors. Since all the detection is based on inferences, the program will review every condition when the elevator stops in a position. Then the supervisor counts the number of faults detected. If there was only one fault, then it should be a temporary fault in a sensor, but if there are more than one fault, then the problem might be in the ultrasonic sensor, since it probably stopped in the wrong position, which resulted in the lack of multiple signals.

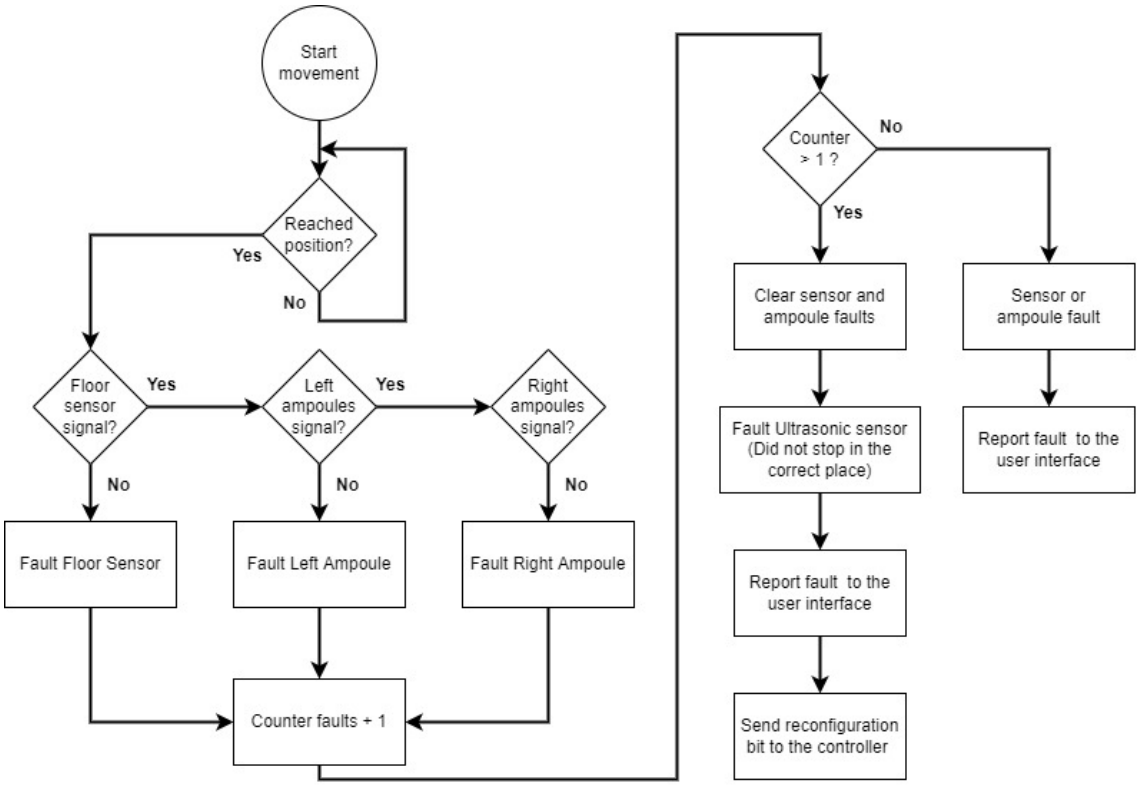


Figure 3.38: Fluxogram for the detection of faults on the sensors.

3.4.6 System Reconfiguration

In the previous section, it was stated that in the case of the failure of the ultrasonic sensor the controller would reconfigure in order to maintain a stable operation. Since the ultrasonic sensor value is the main source of information to control the elevator, then, in the case of its failure for any reason, and to avoid stopping the elevator, the controller

must find other sources of information.

Besides the ultrasonic sensor, that returns an analog value indicating the actual position of the cabin, the elevator still has the floor sensors that indicate when the cabin is passing by the floor and the four superior ampoules, two of which (mono-stable ampoules) indicate when the cabin is in a floor and the other two (bi-stable ampoules) indicate which floor it is.

To sum up, figure 3.39 shows the fluxogram of control based on the two controllers, whether there is an error related to the ultrasonic sensor or not. In this case, the error flag is triggered, and this will decide between the open-loop control, when there is a deviation in the position reading, and the closed-loop control, when the system is functioning as expected.

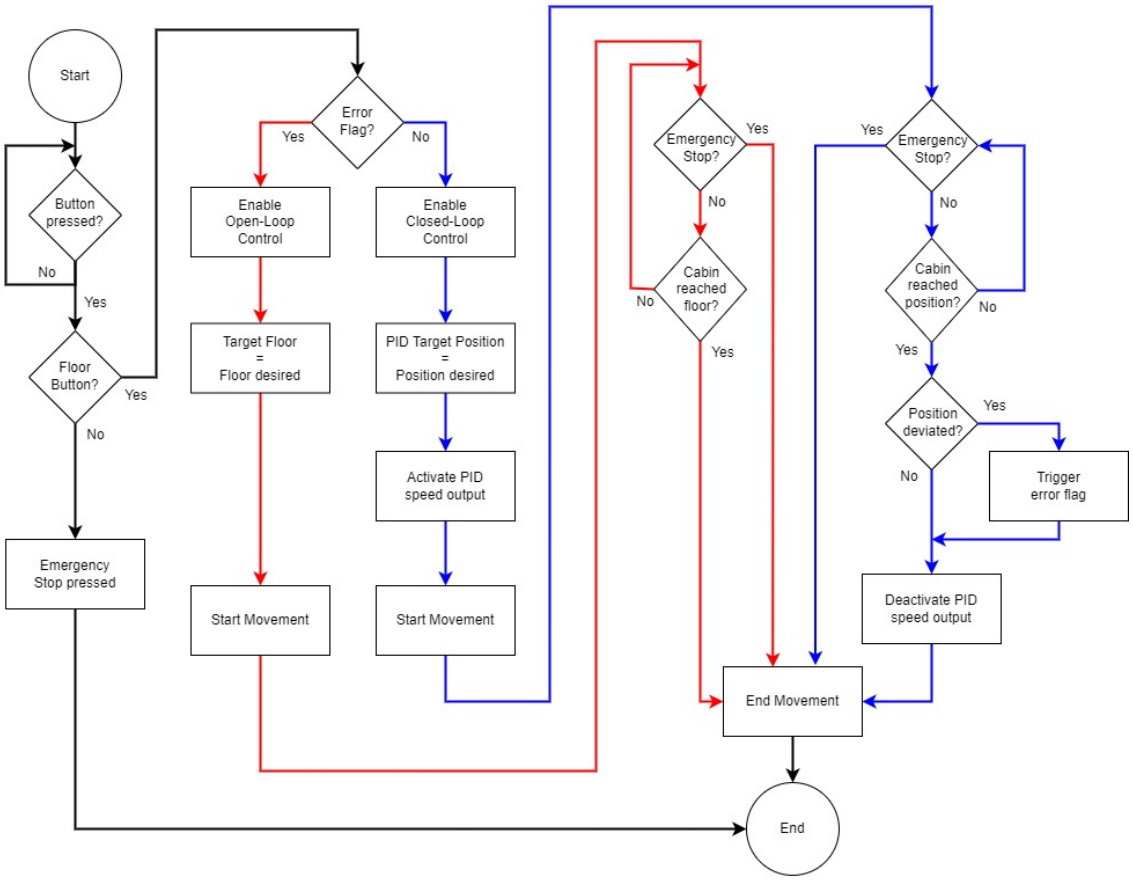


Figure 3.39: Fluxogram for the open-loop and closed-loop control of the elevator.

With the closed-loop control, the elevator uses the ultrasonic sensor reading in order to retrieve its actual position and the PID controller to send the speed setpoint to the VFD. When the PID reads the value from ultrasonic sensor and it detects that the cabin has reached the desired position, the supervisor PLC compares the value read with the expected position and it will know if there is a deviation between the actual and the expected position values. This comparison is done by relating the ultrasonic sensor

reading and the signals from the magnetic sensors. If one of those signals are deviated, then some sensor might be defected, either the magnetic sensors or the ultrasonic sensor.

With the open-loop control, the system makes use of the floor sensors and the magnetic sensors at the top. This can be considered as a simpler controller, since it has no feedback between floors besides the ultrasonic sensor reading that might be defected, when using this open-loop controller. The elevator will start his movement with a stable speed until it reaches a floor, which will make the cabin reduce its speed until it finds the floor sensor.

4

Experimental results

4.1 Introduction

In this chapter the experimental results are presented and discussed.

The first section is related to the results obtained with both controllers: open-loop and closed-loop.

The second section shows the results regarding the HMI and the supervisor PLC, simulating an interaction with the system locally and remotely, where the user changes parameters and send commands to the process.

The third section is related to the fault and failure detection, both in the process and control levels.

Finally, there is an analysis of the results obtained.

4.2 Control Level

4.2.1 Open-loop control

The open-loop controller is the simpler method to control the elevator, since it has no feedback from the actual position of the cabin to control the desired speed and to have a smoother stop when reaching the destination.

Given this, the objective of the experiments with the open-loop control is to reach the desired floor using the VFD frequency, to make the elevator move in the right direction, the magnetic sensors and the floor sensors.

With this controller, the maximum frequency setpoint is half of the established for the system, otherwise the elevator would gain speed enough to not stop in the right place, considering its inertia and considering that the frequency reference has sudden changes, as opposed to a gradual change. The elevator has two different speed setpoints, one between floors, which is half of the maximum frequency, and another when it reaches a floor, which is a fifth of the maximum, in order to have enough time to verify if it is the

desired floor.

In this experiment, demonstrated in figure 4.1, there was a sequence of commands to test the controller, being floor 2 → floor 0 → floor 1 → floor 0 → emergency stop, seen in the bottom chart, and it is possible to see the current frequency of the VFD (absolute value), the maximum frequency and the target frequency, in the chart above, giving this controller a stable performance.

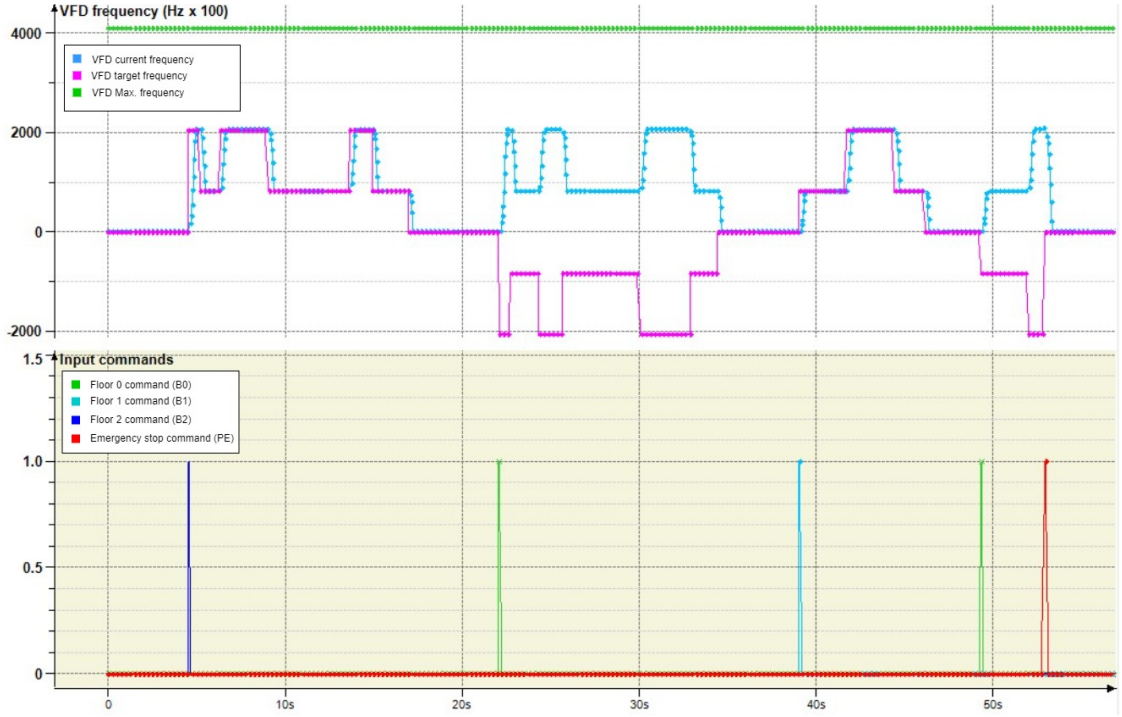


Figure 4.1: Experimental results of the open-loop controller.

4.2.2 Calibration

When starting the system, the elevator needs calibration. This means that the elevator has no reference points for any floor before calibrating. During the calibration, which is initiated by the operator, the elevator runs through its entire course in order to assign each relevant position a value from the ultrasonic sensor. This raw value is then translated into a value in percentage to express each position relatively to the entire course.

The calibration algorithm sends the elevator to the lowest level, which is the position related to the bottom limit switch and then it runs the entire path, with a constant speed, to locate the floors and record each value for them, until it reaches the end of the course, which is the top limit switch. As stated before, these values are then translated into percentages of the path and these relative positions are the values that the PID controller uses for the position setpoint.

In figure 4.2 it is possible to see that, during the movement of the cabin, as seen in the top chart, all the values associated with floors and limit switches start at zero, and

while passing by those sensors, the algorithm records these values which, in the end of the calibration, are used as references to calculate its relative value.

The ultrasonic sensor reads the position of the elevator rather proportionally, but between the floors 1 and 2, the sensor loses the track of the elevator, as the ultrasonic wave may reflect in other parts of the structure. This deviation was a constraint during the experiments and will be seen through all the tests performed, so it had to be considered as a characteristic of the system.

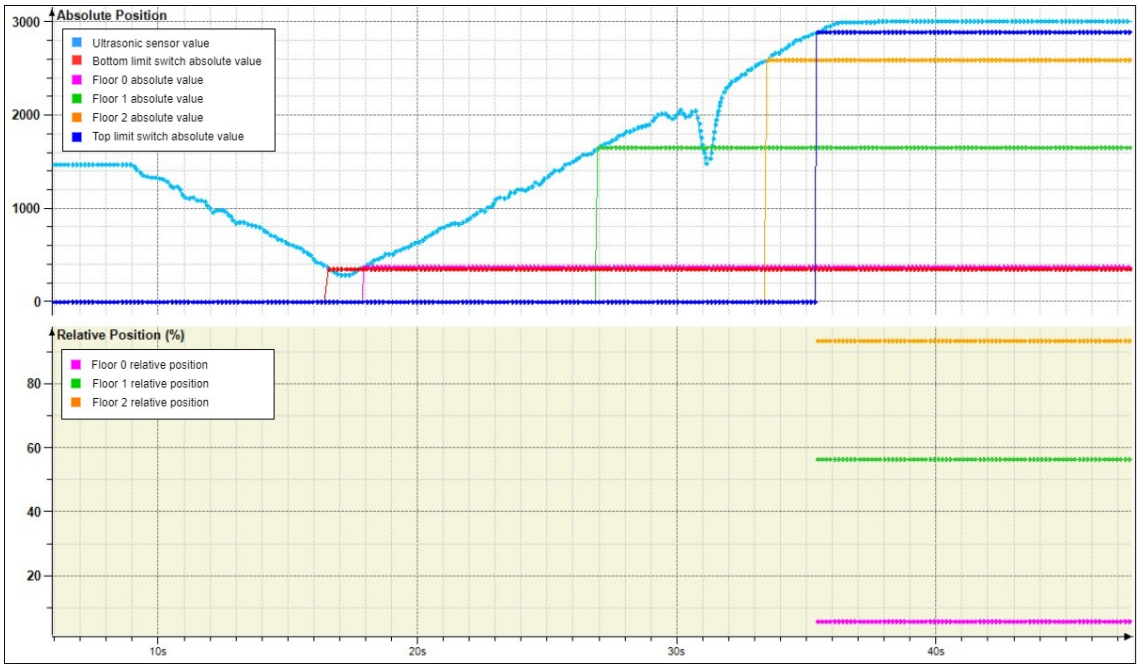


Figure 4.2: Experimental results of the elevator calibration.

In this experience, the values recorded from the sensor are now compared with the height measured between the ultrasonic sensor and the top of the cabin's counterweight. The figure 4.3 with the measurements can be seen below. With these measurements, it is possible to draw a relation between these two values, as shown in figure 4.4, and it is possible to observe that, even considering the deviations of the sensor measurements, the values are distributed along the trend line, with the equation $y = 0,0415x + 0,0153$, which shows a proportional relation, as expected.

4.2.3 Closed-loop control

After calibrating the system, it is now possible to test the PID controller, since it makes use of the relative positions of the floors to control the output value of frequency.

Figure 4.5 illustrates the cabin's relative position during its movement and the setpoints for each floor. In the chart below it is shown the input commands from the local buttons. The PID allows the elevator to have a gradual acceleration and deceleration. To exemplify

US sensor value	Height (cm)
0	0
340	12
377	17
1643	70
2619	108
2900	120

Figure 4.3: Relation between the values from the ultrasonic sensor and the height measured.

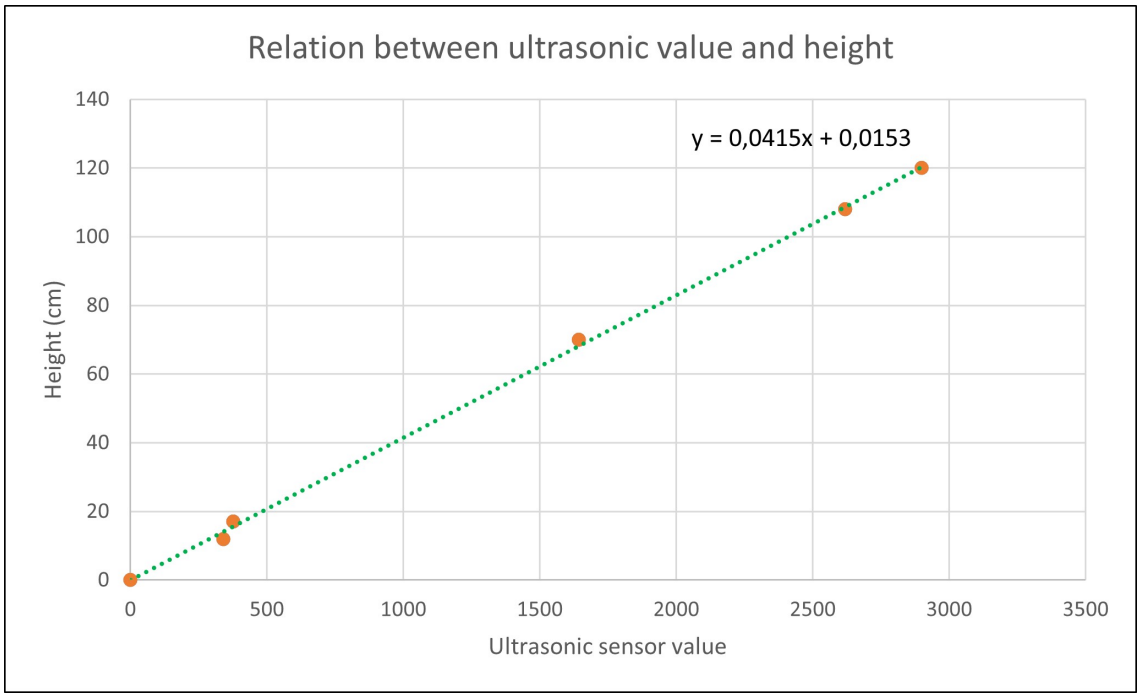


Figure 4.4: Trend line and equation associated to the relation between sensor values and the height.

the movement of this controller, the same sequence of commands was used, i.e., floor 2 → floor 0 → floor 1 → floor 0 → emergency stop.

As seen before, the deviation between the two superior floors is evident, with the value oscillation, while in the other positions the movement is performed as expected. With the last command, the elevator stops as soon as the emergency stop button is pressed which is the most important when the elevator needs to stop immediately.

In figure 4.6 it is shown the relation between the movement between the floors 0 and 2, and the VFD target speed and current speed. This shows a gradual change in the frequency reference in order to have a smooth acceleration when initiating the movement and a smooth stop when reaching the desired position.

When compared to the open-loop control algorithm, this gradual change in the VFD,

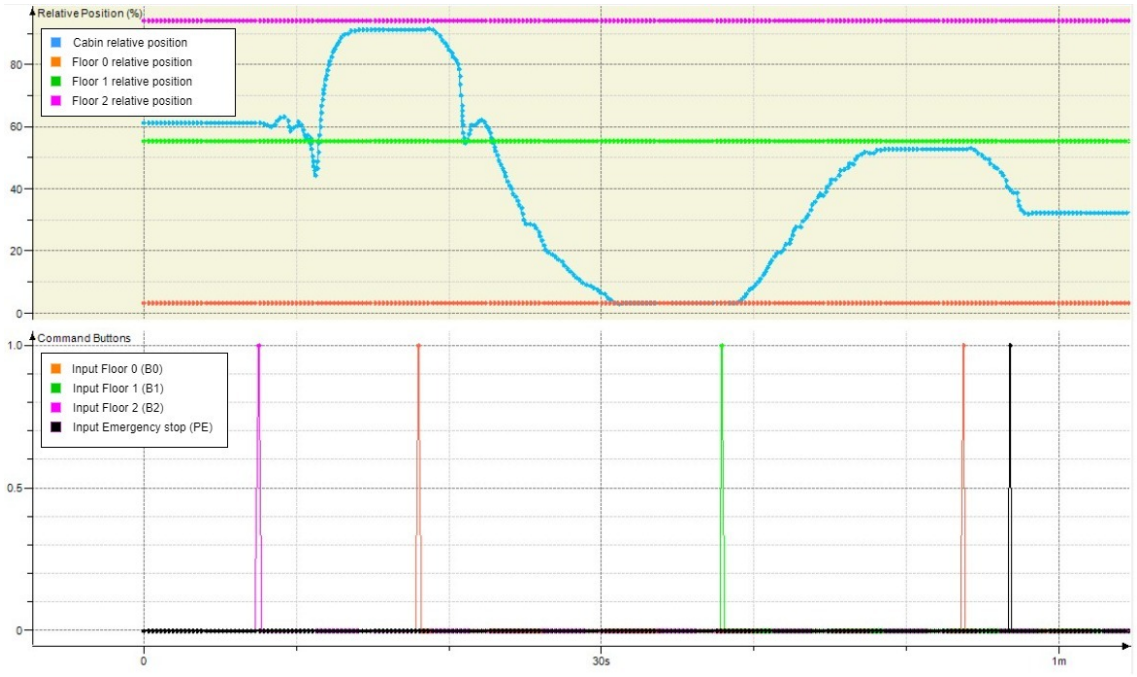


Figure 4.5: Experimental results of the closed-loop controller.

which is the output from the PID, is enough to increase the safety of the system, the comfort of the passengers, to prevent damage in the components and to extend the life span of the equipment.

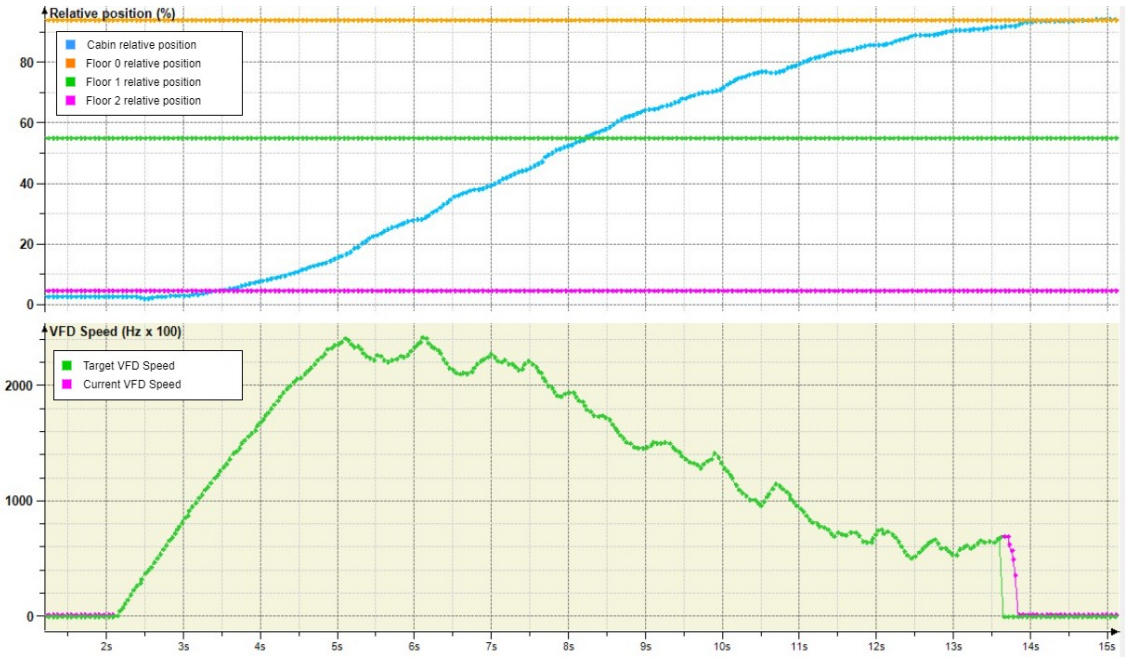


Figure 4.6: Relation between the movement of the cabin and the frequency reference.

4.3 Supervision Level

4.3.1 HMI command inputs

Similar to the experiment with the PID where the controller responded well to the input commands from the local buttons on the structure, with the HMI the operation is similar. The controller read the inputs from the HMI and from its web server almost instantly, as seen in figure 4.7, as if it was the local commands, which is possible because of the high sampling time and transfer rate between the supervisor and the controller, which can be 100Mbit/s with Ethernet TCP/IP for both the supervisor and the controller.

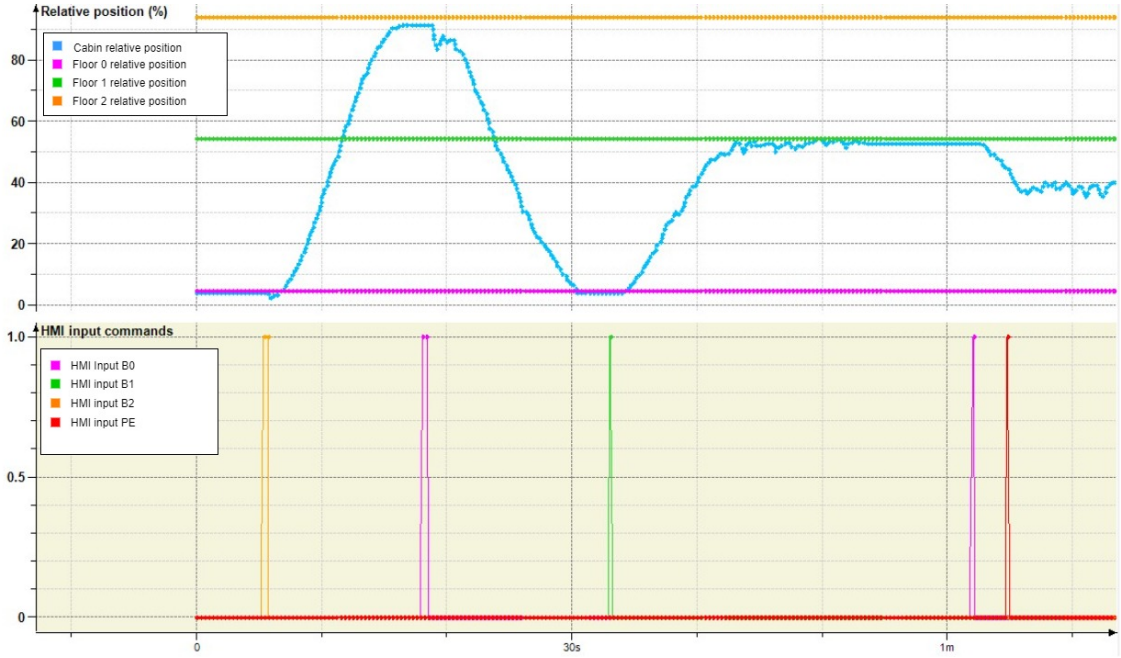


Figure 4.7: Experimental results for the operation through the HMI.

Resuming, in the system there are three different sources of direct input from the user, as seen in figure 4.8: through the local command unit, through the local HMI and through the HMI via Web.

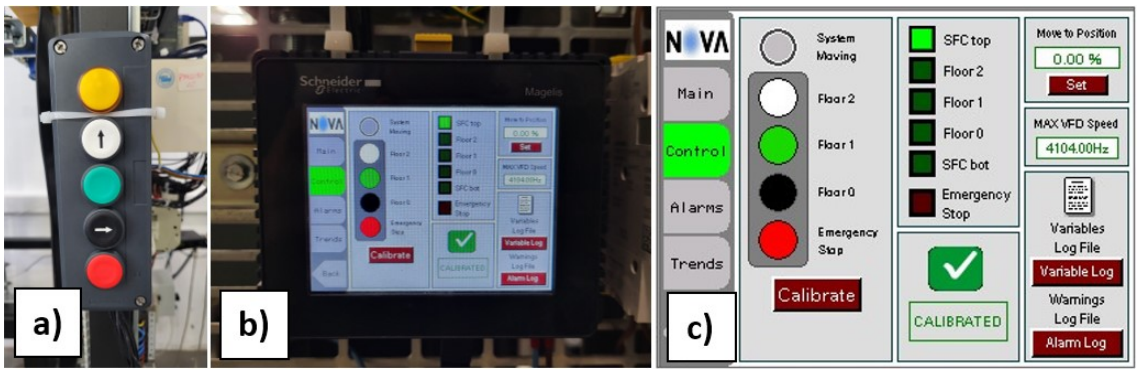


Figure 4.8: Different sources of operator inputs: a) local command unit, b) HMI, c) HMI via Web.

4.3.2 Set target position

Besides the fixed positions of the floors, the user can also give the controller a different setpoint of position, which can be useful, for example, in the case of maintenance when the cabin needs to be adjusted specifically in a position where a sensor can be accessed. Through the HMI, the operation can define a position setpoint, and execute this by activating the *"SET Target position"* as exemplified in figure 4.9. When this bit is set to 1, the elevator assumes that reference as its next target and starts the movement.

With this experiment, it was tested the positions 90%, 23% and 68%. Since there is a deviation related to the ultrasonic sensor that needs to be considered, the elevator in general had an associated error of less than 3%, which represents a difference of less than 5cm, but in some cases, as it is shown for the reference of 23%, there was an error of almost 10%, which derives from the oscillation in the middle of the course, around floor 1. In general, the elevator corresponds to what it is asked of it, but the error associated with the sensors cannot be ruled out.

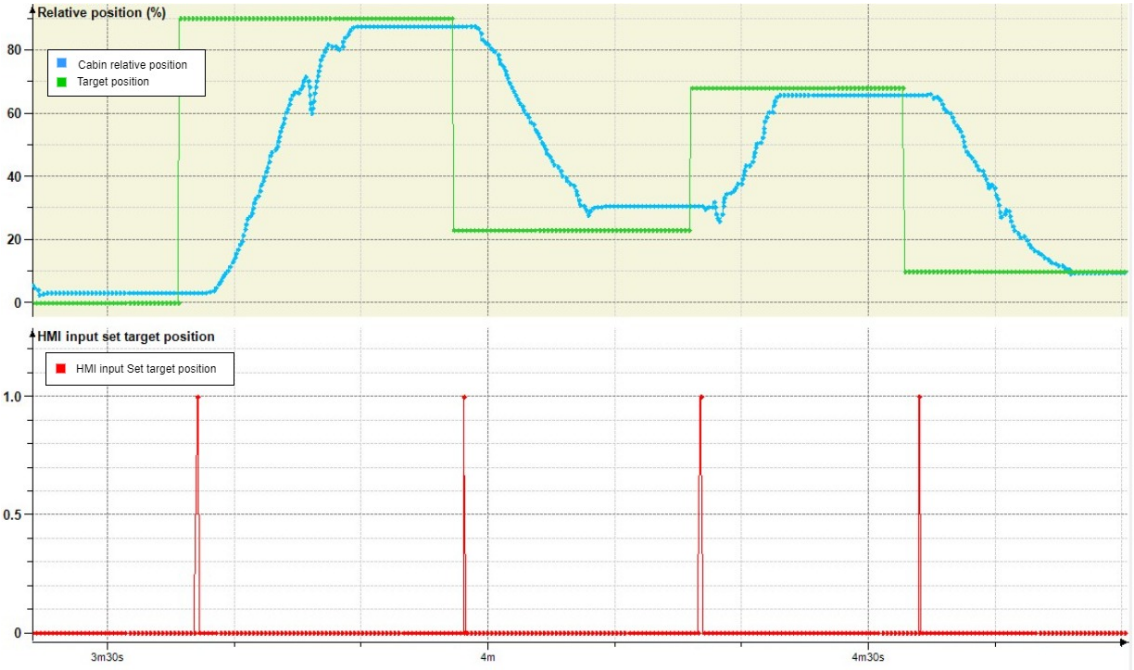


Figure 4.9: Experimental results to an input of specific positions besides the floors.

4.3.3 Set maximum VFD frequency

Besides the parameterization of a specific position setpoint, the user can also define the maximum frequency to be sent to the VFD, which can be shown in figure 4.10 that has the relation between the movement of the cabin and the VFD speed references.

This experiment was done using the open-loop controller in order to be easier to visualize the difference in the behaviour. Since the open-loop controller is programmed to half of the maximum speed, when the maximum frequency of the VFD is changed on the

HMI, from 50Hz to 20Hz, it is possible to see the reduction in the target and the current frequency outputs.

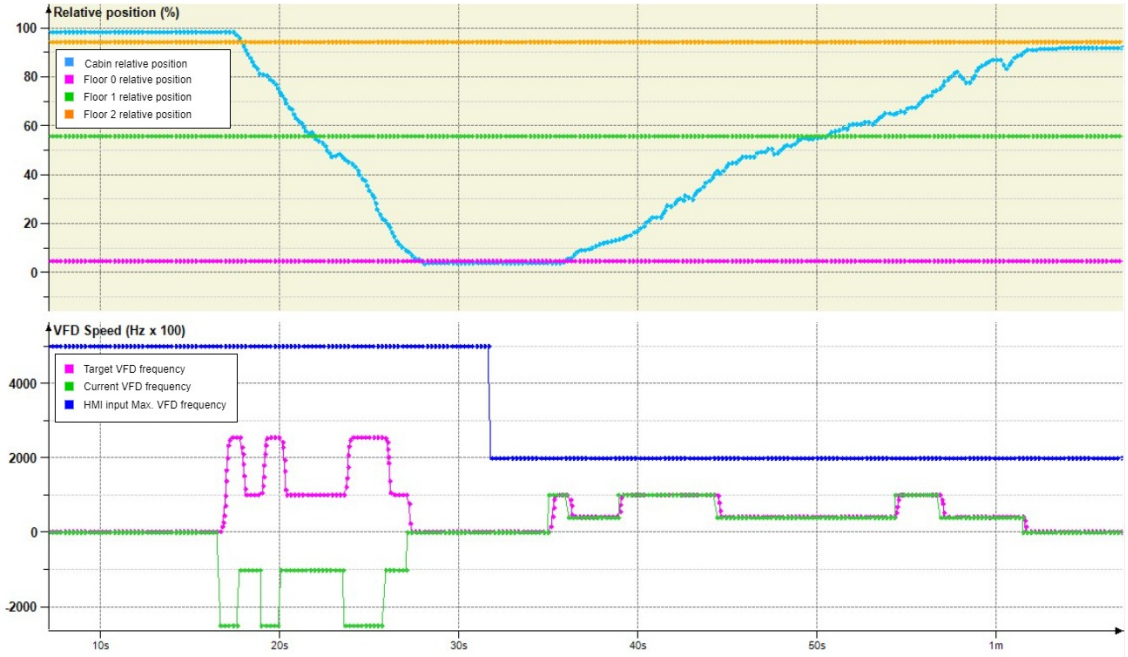


Figure 4.10: Experimental results to a user input for limiting the VFD frequency.

4.4 Fault detection

4.4.1 Floor sensor faults

Regarding the faults mentioned in the previous chapter, one fault is related to floor sensors, which happens when the cabin stops in a floor but there is no feedback signal from the floor sensor, which can indicate that it is defected or misplaced. In this case the system sets a fault bit and sends a warning to the interface. In this experiment, as seen in figure 4.11, the elevator went from the floor 2 to the floor 0 and, while it detected the sensors from floor 2 and floor 1, when it reached the desired position, in the floor 0, the sensor for this floor was not detected, which triggered a fault.

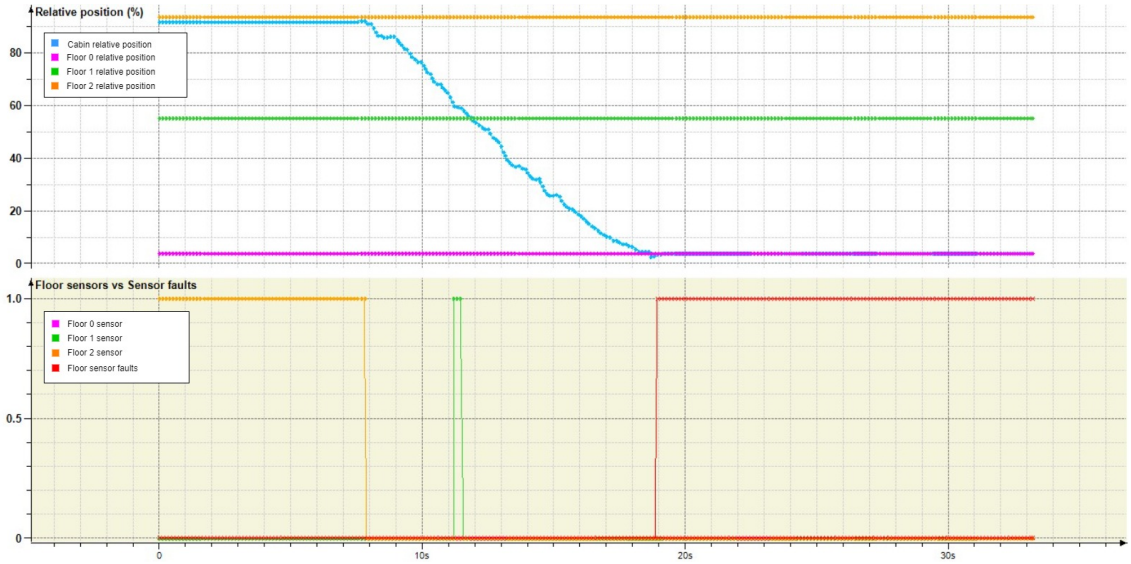


Figure 4.11: Experimental results to detecting a fault in the floor sensors.

4.4.2 Magnetic sensor faults

Besides the floor sensors, the magnetic sensor faults are detected similarly. When the elevator stops, the supervisor verifies if the combination of ampoule signals is correct, otherwise there may be an error. The figure 4.12 illustrates this error when the cabin reaches the floor 0 and one of the ampoules is not giving any feedback.

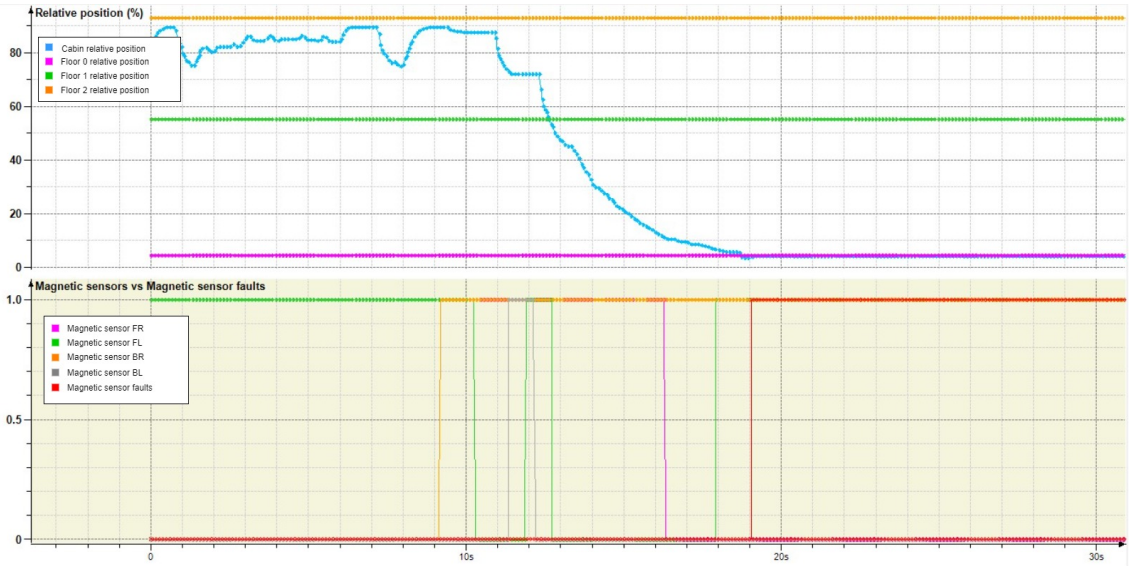


Figure 4.12: Experimental results to detecting a fault in the magnetic sensors.

4.4.3 Ultrasonic sensor fault

The fault related to the ultrasonic sensor is detected indirectly, since there is no direct way of knowing if the sensor is giving a correct reading. Given this, when there are multiple sensors with deviated signals, it means that the elevator stopped in the wrong position. This results in a behavioral change in the control of the elevator.

When the ultrasonic sensor fails, the closed-loop control is no longer reliable, and the open-loop control is activated. For this, the controller needs feedback from the supervisor through a reconfiguration bit, which is triggered when there is a fault with the ultrasonic value feedback.

In figure 4.13 it is possible to see the moment when the fault and the reconfiguration bit are triggered, which is then reflected in the change of the controller. This change is also visible through the difference in the VFD frequency setpoints.

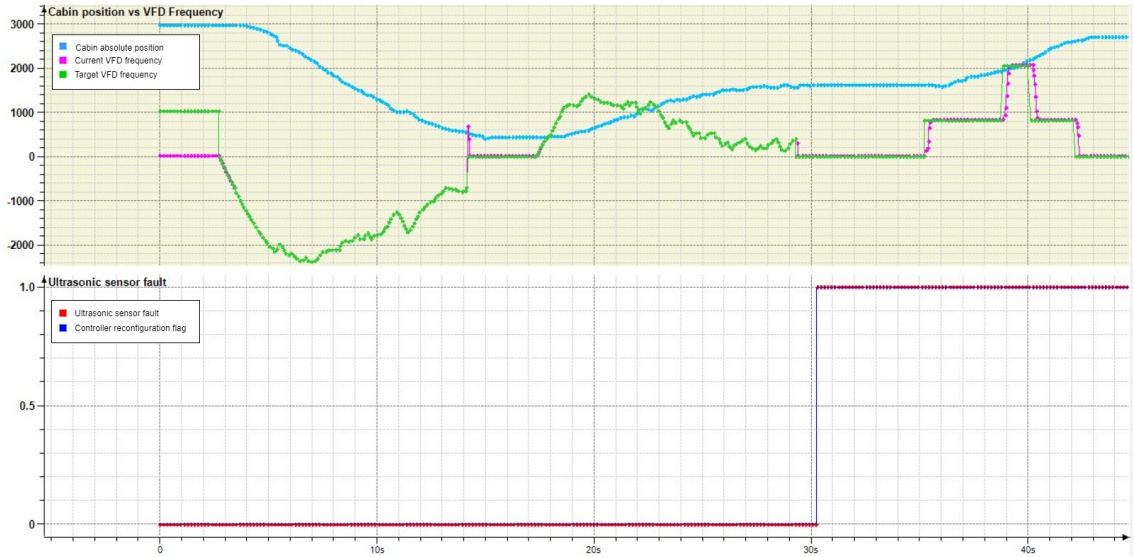


Figure 4.13: Experimental results to detecting a fault in the ultrasonic sensor.

4.4.4 Limit switch failure

Regarding the top and bottom limit switches, a failure in the system is detected if, for some reason, the limit switch activates, which means that the elevator went out of bounds. The elevator is programmed to travel between the three floors and it only goes above or below the floors during the calibration to detect the minimum and maximum level of the path. If the limit switch opens its contact, since it has a normally closed contact, then a failure has happened and the operation of the elevator cannot continue until the elevator is calibrated and this problem is prevented.

In figure 4.14, it is shown the signals from both limit switches, which are normally closed, so normally 1, and it is visible what happens when one of them opens its contact: the emergency stop is triggered in order to stop immediately the cabin and to prevent the motor from overheating when it keeps running and the cabin has reached the end of the course.

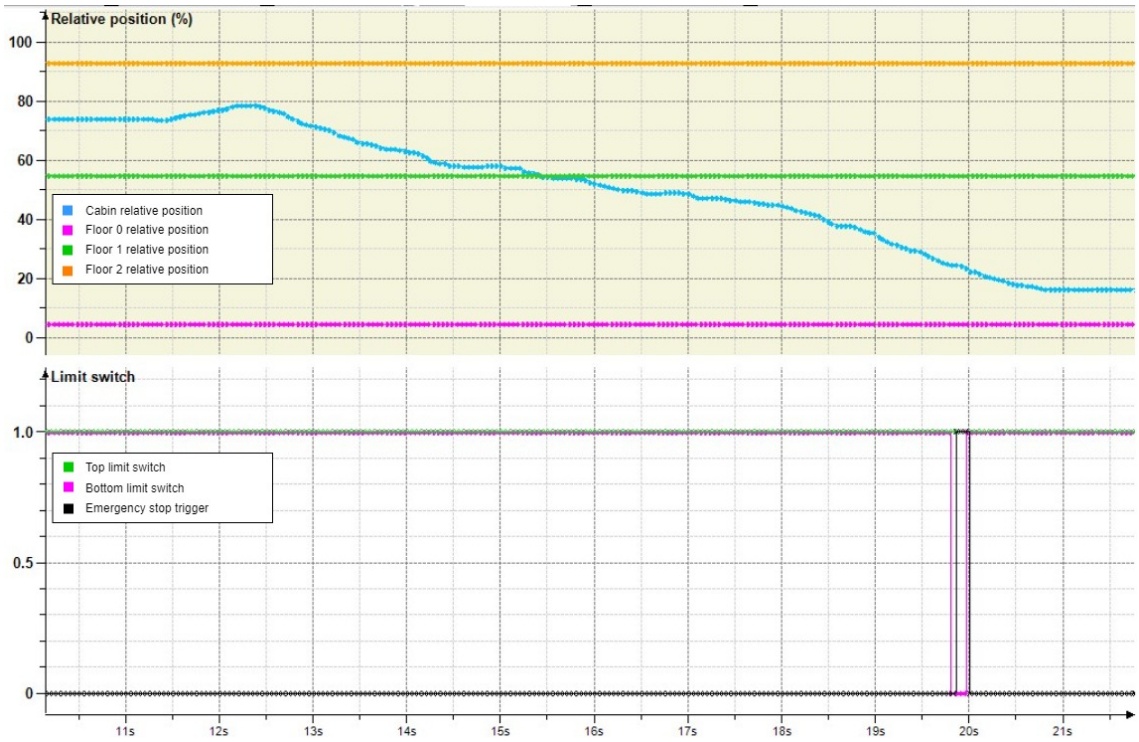


Figure 4.14: Experimental results to detecting a failure related to the limit switch.

4.4.5 Actuator failure

Lastly, the failures related to the actuators, i.e., the motor and its brake, are important to prevent material damage. If the motor runs with the brake engaged, it will overheat the motor and probably cause damage to its components. On the other hand, if the elevator is supposed to be stopped and the break is not engaged, then it is a major safety failure that can cause an accident and more than just material damage, if people are involved.

Given this, the detection of both these failures is critical to the safety and reliability of the system. In this case, in figure 4.15 there is an experiment with the motor running with the break engaged which cause a failure and instant emergency stop on the system. The supervisor waits 1 second before triggering the fault, since that if it was instant, there could be an overlap of signals and mislead the system into triggering a fault when there was no problem.

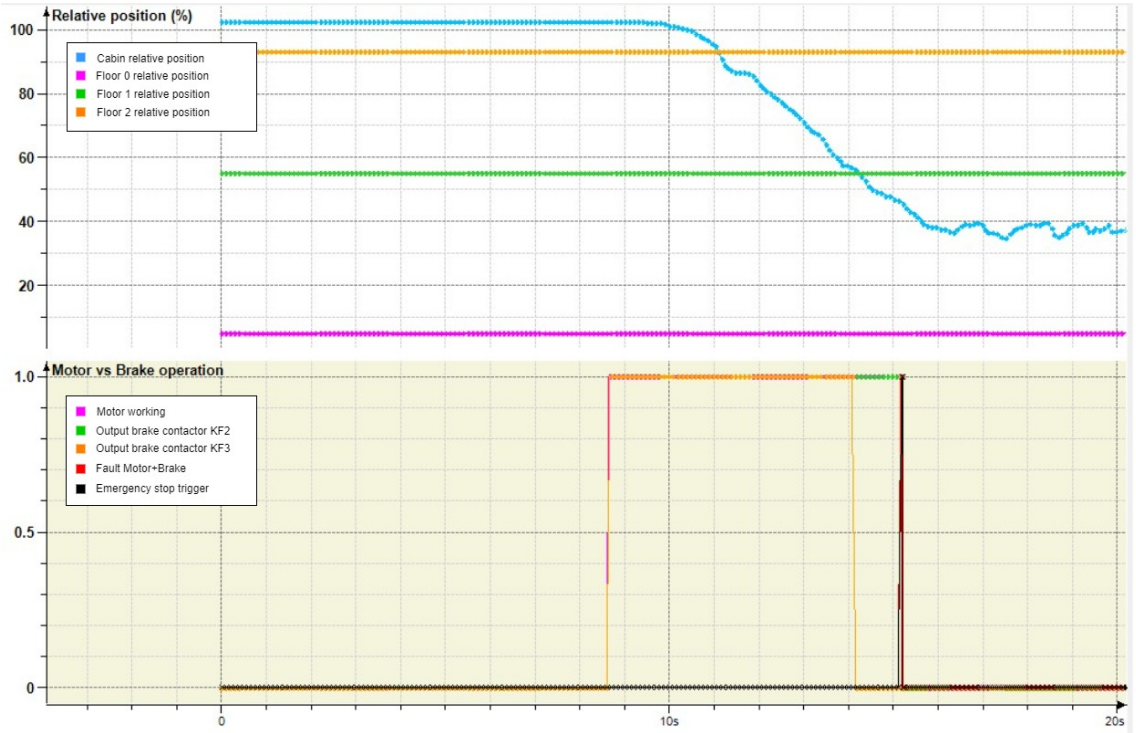


Figure 4.15: Experimental results to detecting a failure when the motor runs with the brake engaged.

5

Conclusions

5.1 Conclusions

The main objective of this work was to develop a supervision system for monitoring a process, which in this case was a freight elevator model present in the automation and industrial control laboratory of the Department of Electrical and Computer Engineering. The supervision algorithm was running in a M262 PLC that was connected via Ethernet TCP/IP to the controller and the HMI. This way, the M262 were the master in this network while the M340 controller and the interface were the slaves. These devices had to be configured as well as the communications between them in order to create a local network where all the devices were accessible through their IP address. Another objective with the creation of this network is to achieve a fully remote laboratory where all the processes present in there are accessible remotely as long as the personal machine is connected to the network. As stated before, this would be a major improvement in the methods of teaching and testing algorithms in the physical processes.

As the devices were all connected, it was possible to retrieve and send information between all the devices. With this, came the possibility of having a visual interface that allowed the user, representing an industrial operator, to interact directly with the process by observing its operation and by sending manual parameters from the interface to the controller. This interaction could be done either locally in the HMI or remotely via the HMI Web server which had all the actions enabled similar to the physical device with no conflict between these two sources of inputs.

Another objective of this work was the fault and failure detection, that was also implemented in the supervisor. Based on a database of faults previously detected in the system, it was possible to build a method for the detection of errors. These fault scenarios were gathered in order to develop an inference mechanism to compare the real scenario, where all the signals from the sensors were available, with the fault scenarios and identify possible reasons for warning. After detecting a fault, the supervisor alerts the user via the HMI and, depending on the fault or failure, takes action to overcome or resolve the

defect, where it needs an exterior intervention, or a reconfiguration in the controller.

Regarding all these points stated above, the objectives were met. The supervisor is able to send the information about all the relevant variables to the user through the HMI; it can successfully detect faults in real time and create a warning in the visual interface to alert the user; the HMI is accessible through the Web server embedded in the device; using multiple controllers, the supervisor is able to change between open-loop control and closed-loop control, depending on the fault detected in the system, in order to maintain a stable operation; the supervisor can send manual parameters entered by the operator, such as a position desired, or a limit to the VFD frequency.

Regarding limitations during this work, one that was mentioned in the chapter of the experimental results was the constant deviations from the ultrasonic sensor. This sensor is a very useful tool since it is because of this sensor that it is possible to have a feedback of the actual position of the cabin and successfully implement a PID controller in the system, otherwise it would be impossible to have any consistent feedback from any other source. Anyhow, given the sensor's working principle, it was concluded that this was not the best option as it has associated errors when it the sound wave was reflected in random surfaces, misleading the system with deviated values. Since this fact could not be avoided, it had to be viewed as a feature, since it was already expected that in certain positions the sensor would not read the correct value. Other limitation was the local network that could not be expanded to the outside due to superior authorizations. With this expansion, the process would be accessible from home, for example. As stated before, this would be a major feature for the students since the work they were doing could be monitored remotely by the teachers.

5.2 Future Work

Regarding future work, there are some points to note in the future to improve the system and continue the process of digitalization of the systems in the laboratory. The following topics might be interesting to consider in the future regarding this freight elevator model:

- Install a different and more reliable method for detecting the position of the cabin in real time. Besides a position sensor, it could be installed speed sensors and acceleration sensors, which could lead to different methods of control using, for example, sensor fusion, where the position, velocity and acceleration of the elevator could be predicted based on the other sensors and it could be easier to detect faults.

- Install temperature sensors in the traction machine to prevent overheating and unnecessary damage.

- Install a bigger version of the HMI screen, given that the one has reduced dimensions and it is difficult to interact with. Besides this, the Web server presents the interface

based on its dimensions, so even if the HMI is accessible through the Web, it is not the most practical tool to use, which would be solved with a screen with more resolution.

- Create different users for the students and teachers with different types of authorization and access. The sessions for each user could also be limited to some time in order to let everyone, i.e. the students, interact with the processes.
- Make use of the supervision and fault detection features to create a tool to monitor the progress of students in the development of applications, which can trigger alarms during their experiments in order to validate their algorithms. This solution can be used for both teaching and researching.
- Extend the local network to the outside in order to let the students access the processes from home. This feature needs authorization since that the local network needs to be connected to University network which could then be accessed by the students, and this requires more security configurations to protect the network. This can be done using, for example, the MQTT protocol which allows the network to be accessible from remote locations while maintaining its security, as long as the environment it is built in is also secure.
- Make use of the M262 as a gateway to receive information from other controllers present in the laboratory, specifically the Siemens controllers which required an interface between Modbus and Profibus to allow the communication between these controllers from different manufacturers.
- Create a centralized Web interface connected with all the web servers from the different controllers where the data from the laboratory processes could be easily accessed.

Bibliography

Ângelo, N. (2016). "Projecto e Concepção de um Sistema Elevador Monta-cargas Industrial". In.

Aström, K. J. et al. (2011). *Control of complex systems*. Springer Science & Business Media.

Baillieul, J. and T. Samad (2015). *Encyclopedia of systems and control*. Springer Publishing Company, Incorporated.

Bartoszewicz, A. (2011). *Robust Control, theory and applications*. InTech.

Blanke, M. et al. (2006). *Diagnosis and fault-tolerant control*. Vol. 2. Springer.

Bolton, W. (2015). *Programmable logic controllers*. Newnes.

Brito Palma, L. (2007). "Fault detection, diagnosis and fault tolerance approaches in dynamic systems based on black-box models". In.

Cardoso, A. J. L. (2006). "Supervisão e controlo de sistemas dinâmicos com tolerância a falhas: contribuição para uma abordagem estruturada e robusta". PhD thesis.

Erickson, K. T. (1996). "Programmable logic controllers". In: *IEEE potentials* 15.1, pp. 14–17.

Frank, P. M. (1996). "Analytical and qualitative model-based fault diagnosis–a survey and some new results". In: *European Journal of control* 2.1, pp. 6–28.

Galloway, B. and G. P. Hancke (2012). "Introduction to industrial control networks". In: *IEEE Communications surveys & tutorials* 15.2, pp. 860–880.

Irwin, J. D. (1997). *The industrial electronics handbook*. CRC press.

Jiang, J. (2005). "Fault-tolerant control systems-an introductory overview". In: *Acta Automatica Sinica* 31.1, pp. 161–174.

Khalastchi, E. and M. Kalech (2018). "On fault detection and diagnosis in robotic systems". In: *ACM Computing Surveys (CSUR)* 51.1, pp. 1–24.

Mahmoud, M. S. and Y. Xia (2013). *Analysis and synthesis of fault-tolerant control systems*. John Wiley & Sons.

Marzat, J. et al. (2009). "Autonomous fault diagnosis: state of the art and aeronautical benchmark". In: *3rd European Conference for Aero-Space Sciences, EUCASS'2009*, p. 76.

Miljković, D. (2011). "Fault detection methods: A literature survey". In: *2011 Proceedings of the 34th international convention MIPRO*. IEEE, pp. 750–755.

Monteiro, A. (2016). "Controlo e Estudo Dinâmico de Ascensores para Otimização do Conforto Humano". In.

Patton, R. J. (1991). "Fault detection and diagnosis in aerospace systems using analytical redundancy". In: *Computing & Control Engineering Journal* 2.3, pp. 127–136.

— (1997). "Fault-tolerant control: the 1997 situation". In: *IFAC Proceedings Volumes* 30.18, pp. 1029–1051.

Petruzella, F. (2017). *Programmable Logic Controllers*. 5th ed. Vol. 1. McGraw-Hill Education.

Santos, D. (2022). "Protótipo Ciber-Físico de Elevador Monta-cargas". In.

Stouffer, K., J. Falco, and K. Scarfone (2011). "Guide to industrial control systems (ICS) security". In: *NIST special publication* 800.82, pp. 16–16.

Zhang, P. (2010). *Advanced industrial control technology*. William Andrew.

Zhang, Y. and J. Jiang (2008). "Bibliographical review on reconfigurable fault-tolerant control systems". In: *Annual reviews in control* 32.2, pp. 229–252.