Universidade de Aveiro Departamento de Engenharia Mecânica
2021

Daniel Filipe Silveira Coelho

A Predictive Maintenance Approach based on Time Series Segmentation

Uma abordagem de manutenção preditiva baseada na segmentação de séries temporais

Universidade de Aveiro Departamento de Engenharia Mecânica
2021

Daniel Filipe Silveira Coelho

A Predictive Maintenance Approach based on Time Series Segmentation

Uma abordagem de manutenção preditiva baseada na segmentação de séries temporais

Relatório de Projeto apresentado à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica, realizado sob orientação científica do Doutor José Santos, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro, e do Doutor Eugénio Rocha, Professor Associado do Departamento de Matemática da Universidade de Aveiro

o júri / the jury

presidente / president	**Prof. Doutor Miguel Armando Riem de Oliveira** Professor Auxiliar da Universidade de Aveiro
vogais / examiners committee	**Prof. Doutora Ana Maria Pinto de Moura** Professora Auxiliar da Universidade de Aveiro
	Prof. Doutor Eugénio Alexandre Miguel Rocha Professor Associado da Universidade de Aveiro

agradecimentos / acknowledgements

Agradeço a todas as pessoas que contribuíram direta ou indiretamente na realização deste projeto, bem como todas as pessoas que me ajudaram durante todo o percurso académico.
Agradeço aos meus orientadores, Professor Doutor José Santos e Professor Doutor Eugénio Rocha por todo o acompanhamento e ajuda ao longo deste projeto. Agradeço também a todos os profissionais da Bosch Thermotechnology, S.A. que me ajudaram sempre que necessário, em especial ao Engenheiro Duarte Almeida.
Por fim, agradeço também à minha família por todo o apoio incondicional ao longo deste anos.
O presente estudo foi também realizado ao abrigo do Projeto "Augmented Humanity" [POCI-01-0247-FEDER-046103], cofinanciado pelo Programa Operacional Competitividade e Internacionalização e pelo Programa Operacional Regional de Lisboa do PORTUGAL 2020, através do Fundo Europeu de Desenvolvimento Regional.

keywords

Predictive Maintenance, Time Series Segmentation, Time Series, Data Analysis, Data Processing, Anomaly Detection, Forecasts

abstract

The increase in automation provided by Industry 4.0 combined with the growing competitiveness in the market highlights the importance of intelligent maintenance. Companies must rethink current maintenance strategies in order to detect failures before they occur. This is the motto of predictive maintenance, through the analysis of data from equipment it is possible to predict when failures will occur and act in accordance with the forecast.
This project, in addition to developing a platform capable of receiving and processing data in real-time from different equipment, also proposes a predictive maintenance approach based on time series segmentation. This new predictive maintenance approach was applied to data from a mechanical press, located in Bosch Thermotechnology, S.A., having achieved an efficiency of 90.91%. Throughout the document, all elements of the developed system are discussed in detail, from the data acquisition systems to sending forecasts on the condition of the equipment to a visualization platform.

palavras-chave

Manutenção Preditiva, Segmentação de Séries Temporais, Séries Temporais, Análise de Dados, Processamento de Dados, Deteção de Anomalias, Previsões

resumo

O aumento da automatização proporcionada pela Indústria 4.0 aliada à crescente competitividade no mercado destaca a importância de uma manutenção inteligente. As empresas devem repensar as atuais estratégias de manutenção de modo detetar de forma antecipada as avarias. Este é o lema da manutenção preditiva, através da análise dos dados dos equipamentos é possível prever quando as avarias irão ocorrer e agir em conformidade com a previsão.

Este projeto, para além de desenvolver uma plataforma capaz de receber e processar dados em tempo real de diversos equipamentos, também propõe uma abordagem de manutenção preditiva baseada na segmentação de séries temporais. Esta nova abordagem foi aplicada a dados de uma prensa mecânica da Bosch Thermotechnology, S.A., tendo-se alcançado uma eficiência de 90.91%. Ao longo do documento é abordado em detalhe todos os elementos do sistema desenvolvido, desde os sistemas de aquisição de dados, até ao envio das previsões sobre a condição do equipamento para uma plataforma de visualização.

Contents

List of Tables

Intentionally blank page.

List of Figures

List of Abbreviations and Symbols

n_{ano} Maximum number of detected anomalies to send the *Safe* alert.

n_{mes} Minimum number of messages in the aggregation to activate the *Working* status.

t_{agg} Aggregation time, in seconds.

t_{for} Time interval between each segment analysis, in minutes.

t_{max} Maximum virtual time in anomaly detection, in minutes.

t_{min} Minimum virtual time in anomaly detection, in minutes.

ABOD Angle Based Outlier Detection.

AET Acoustic Emission Testing.

ANN Artificial Neural Network.

API Application Programming Interface.

AR Autoregressive.

ARMA Autoregressive Integrated Moving Average.

ARMA Autoregressive Moving Average.

CM Corrective Maintenance.

COPOD Copula-Based Outlier Detection.

CSV Comma-Separated Values.

DL Deep Learning.

g g-force.

I2C Inter-Integrated Circuit.

IoT Internet of Things.

IQR Interquartile Range.

IRT Infrared Thermography.

KNN K-Nearest Neighbors.

LDA Linear Discriminant Analysis.

LOA Lubrication Oil Analysis.

LSTM Long Short-Them Memory.

MA Moving Average.

MAR Missing at Random.

MCAR Missing Completely at Random.

MCAR Missing Not at Random.

ML Machine Learning.

MOM Message Oriented Middleware.

PCA Principal Component Analysis.

PdM Predictive Maintenance.

PM Preventive Maintenance.

ppm Parts per Minute.

REST Representational State Transfer.

RNN Recurrent Neural Network.

rpm Revolutions per minute.

RUL Remaining Useful Life.

SARIMA Seasonal Autoregressive Integrated Moving Average.

SCK Serial Clock.

SDA Serial Data.

SDI/MISO Serial Data In.

SDO/MOSI Serial Data Out.

SOS Stochastic Outlier Selection.

SPI Serial Peripheral Interface.

SVM Support Vector Machine.

TSDB Time Series Database.

UCM Ultrasonic Condition Monitoring.

VSA Vibration Signature Analysis.

Intentionally blank page.

Chapter 1

Introduction

In recent years, the industrial environment has changed profoundly due to successive technological developments in manufacturing processes. The new concept is called *Industry 4.0*. *Industry 4.0* enables the manufacturing sector to become digitized with sensors in all manufacturing components. The modernization intended by *Industry 4.0*, using new technologies such as IoT (*Internet of Things*) devices, databases on the WEB, embedded systems and other automation solutions, is essential to support and facilitate actions by maintenance. Predictive maintenance (PdM) is therefore necessary to achieve the objectives proposed by *Industry 4.0*.

PdM has played an increasingly important role in the lives of companies. It uses data analysis techniques to detect anomalies in equipment allowing maintenance to occur before a break occurs. The goal of PdM is to only perform maintenance when it is really required, neither before nor after. This has several advantages, such as, decreased maintenance time, reduced of unscheduled equipment downtime, increased equipment lifespan and reduced maintenance cost [Tran Anh *et al.* 2018].

This project was developed at Bosch Thermotechnology, S.A., belongs to the predictive analysis case study within the mobilizing project *Bosch Industry 4.0 - Augmented Humanity*.

1.1 Objectives

This project has three objectives, the first is to develop a system that receives data from sensors from different equipment in real-time, proceeding with its processing and storage. The second objective is to implement a PdM system on the *Haulick & Roos* mechanical press. The third and last objective, is to develop a visualization platform that contains all the data from the sensors of the various equipment as well as the forecasts issued by the PdM system implemented.

The developed system must have the capacity to receive data from different equipment simultaneously and in real-time and proceed with its processing and storage. Thus, it is possible to converge all equipment data on a single platform, facilitating access to data, both for analysis and visualization. With the increase of IoT devices in the manufacturing facility, there is an exponential increase in the data produced, and there is an increasing need for a platform that has the capacity to receive all the data and store it according to its origin.

The *Haulick & Roos* press has had several breakdowns over the past few years resulting in high economic costs. Because of this, we propose the development of a PdM system based on an new approach. Instead of using all the data from the operation of the press, a segmentation of the time series is carried out first and then the created segments are analyzed. In this way, it is expected that it is possible to detect anomalies more easily and therefore, using less complex algorithms than the common approaches.

The visualization platform must be accessed from any device within the company and must contain all the data from the sensors associated with each equipment. The platform must allow the visualization of data not only at the current time, but also at any moment in the past. In addition to visualizing data from the sensors, it must also integrate the forecasts of the PdM system so that the information reaches the maintenance team to act according to the forecast made.

1.2 Document Organization

This document is organized as follows: Chapter 2 presents the company that provided the realization of this project, explaining the mechanical press that is the target of study of the PdM system. Chapter 3 presents the state of the art, covering the basic concepts associated with the problem to be solved. Chapter 4 describes the proposed solution in general as well as its architecture. Chapter 5, 6, 7 and 8 describe the implementation of the proposed solution. Chapter 9 expresses the results obtained using different evaluation metrics and finally, Chapter 10 presents the conclusions and future work.

Chapter 2

Company

This chapter presents the company that provided the existence of this project, Bosch group. This chapter is divided into three sections, starting from the most general to the most specific. Section 2.1 addresses the Bosch group, indicating some important facts in its history. The Section 2.2 concerns the presentation of Bosch Thermotechnology, S.A. addressing both its activity and its functional structure. Finally, Section 2.3 describes the mechanical press that is the target of the PdM system.

2.1 Bosch Group

In the year 1886, in Stuttgart, Robert Bosch created a mechanical precision and electrical engineering workshop that later became a globally operating company. That year was the beginning of the Bosch group's activity. Initially, the group focused on installing telephone systems, and it took 13 years to enter the construction, repair and sale of precision mechanical and electrical engineering equipment. Since then, it has diversified the market, betting on several business areas. Today, the Bosch group is present in more than 60 countries.

The Robert Bosch Foundation owns 92% of the Bosch group, this foundation is a charity focusing on promoting natural and social sciences, including public health, science, education, society and culture, and international relations. The remaining 8% are divided between the Bosch family and Robert Bosch GmdH.

Currently, the Bosch group presents 4 major business sectors, mobility solutions, industrial technology, consumer goods and energy and construction technology. Figure 2.1 shows the percentage of each of these sectors in total sales in 2014 [Mayeen 2015].

The products and services of the Bosch group are designed to be efficient, comfortable and increase customer safety, these principles are reflected in the group's slogan, "Invented for life".

2.1.1 Bosch Group in Portugal

In 1911, 25 years after the creation of the workshop, Bosch installed itself in Portugal. About 90% of its production in Portugal is exported to over 60 countries worldwide, making Bosch one of the 10 largest exporters in Portugal. Moreover, Bosch is one of the country's largest industrial employers.

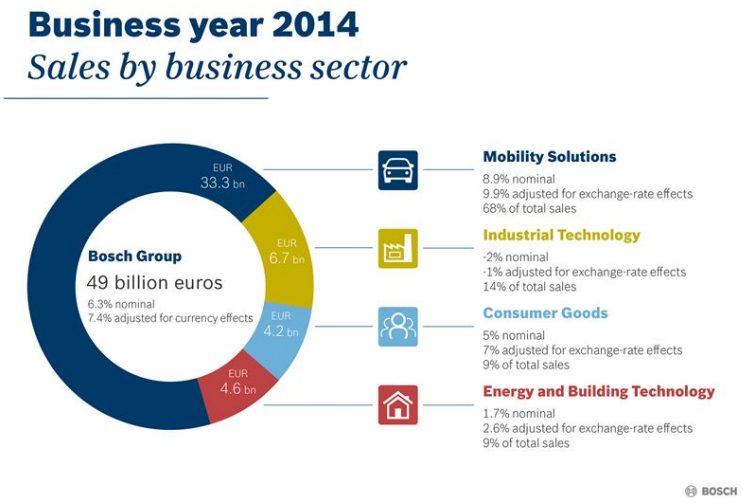


Figure 2.1: 2014 Sales of Bosch [Mayeen 2015]

Bosch is represented in Portugal by Bosch Thermotechnology, S.A., in Aveiro, Bosch Car Multimedia Portugal, S.A, in Braga, and Bosch Security Systems, S.A., in Ovar. These companies develop hot water solutions, car multimedia and communication and safety systems, respectively. The group also has, in Lisbon, a sales office and a subsidiary of BSH home appliances.

2.2 Bosch Thermotechnology, S.A.

Bosch Thermotechnology, S.A., Figure 2.2, started its activity in Cacia, Aveiro in 1977, under the designation of "Vulcano Termodomesticos", based on a licensing agreement with Bosch. Until 1998, the company was constituted by a totally national capital. In that year, the Bosch group acquired the majority of its capital, incorporating it into the thermotechnology division.

Bosch Thermotechnology's activity is the design, development, production and assistance of water heating equipment. In addition to the production of domestic water heating systems, which represents more than 60% of sales, the company also supplies more innovative products, such as floor boilers, solid fuel boilers, cogeneration systems and industrial boilers. The equipment produced is associated with international brands with great prestige in the area of thermotechnology, such as Bosch, Buderus and Junkers.

2.2.1 Company Functional Structure

Bosch Thermotechnology, S.A. in terms of infrastructure is divided into three major departments, administration, development and production. The administration is in charge of coordinating all projects, the company's existing resources as well as all activities related to the thermotechnology division. The development department is in charge of all research, creation and development of products in order to satisfy the needs of the market. Finally, the production department is in charge of the whole process related to

Figure 2.2: Bosch Thermotechnology, S.A. in Cacia, Aveiro [Bosch 2020]

the manufacture of products, management of the factory floor and efficient production.

The present work focuses on the production department, as the focus is the analysis of data related to equipment present in the manufacturing facility. This department is composed of several production processes, however, it can be divided into two major areas, transformation of raw material and assembly lines. The mechanical press, which is the target of the PdM system, is located in the first area, since it transforms metal sheets into various products ready to move on to the next process.

Figure 2.3 presents an outline of the material flow, in which the press is inserted, from the coils with the raw material to the assembly lines.

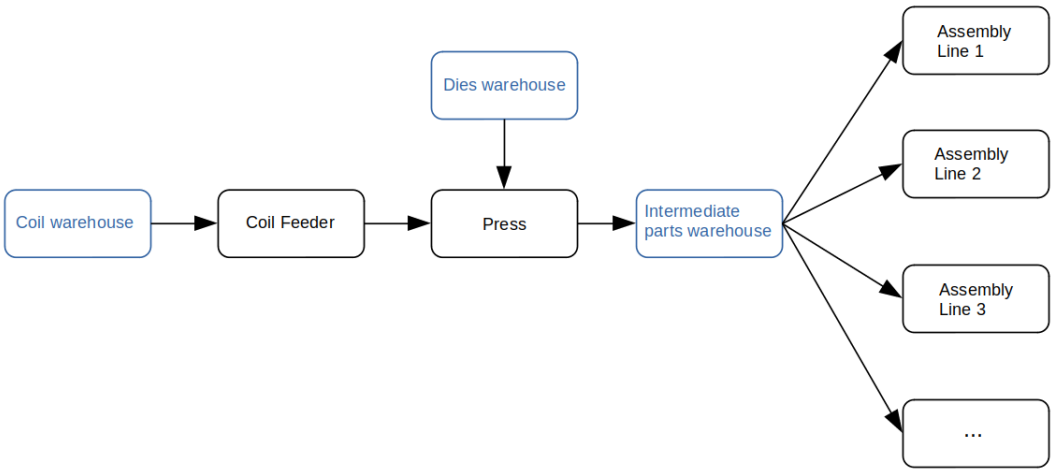


Figure 2.3: Material flow

The press is the first stage of transformation of the raw material. At its entrance

there is a coil feeder whose function is to continuously supply the steel sheet that will undergo all operations. The dimensions of the sheet are not always the same, both the thickness and width of the sheet may vary. After the sheet has undergone all operations caused by the die, it reaches the end of the press with the desired geometry. The part is then considered intermediate product and is stored until needed in the assembly line. This intermediate part warehouse supplies several assembly lines, each of which produces a final equipment. Next to the press are also two warehouses, a warehouse that contains all the coils with the different sheets and a warehouse that contains all the dies. These two warehouses are located close to the press in order to shorten the path made when the part to be produced changes.

2.3 Description of the Press

The data that served as the basis for the development of this project was produced by sensors housed in a stamping press, *Haulick & Roos RVD 200T*, Figure 2.4.

A stamping press is a machine used to shape and cut sheet metal by using a die to perform the deformation. The die is mounting into a press, and depending on the shape of the die, the shape of the final state of the metal sheet will be different. The dies are made of high-strength alloys for the purpose of withstanding high stresses and high temperatures during the forming period. Based on these characteristics and knowing that it is intended that these dies have a long life, they are naturally quite expensive to manufacture [Santos 2012].

In addition to the die, there are two more main components that are part of the press, bolser plate and ram/slide.

Bolser plate is placed on top of the press bed and is a large block of metal, on which the lower part of the die is mounted. This component has no movement. The ram or slide guidance is a moving element, on which the upper part of the die is mounted, and is a critical element in the conformation of the part, and should not present distortions in its trajectory with the risk of failure.

The presses are divided into three large groups depending on the systems that drive them, they can be mechanical, hydraulic or servo controlled. A mechanical press is based on the principle of converting a circular motion of a motor into a linear motion of a slide. In a hydraulic press, the cinematic system consists of hydraulic cylinders that perform linear trajectories, similar to the slide trajectory. Finally, a servo press is a machine that uses a servomotor as the input power.

The *Haulick & Roos* press is a mechanical press, more precisely, of the *Crank* type. It works as follows: The press uses a crank link and a drive shaft connected to each other. The crank link is connected to the motor, and when the motor rotates, the crank link rotates as well, making the drive shaft rotating. The drive shaft is attached to a connecting rod by a rotational joint. The connecting rod is, in turn, attached to the slide by a rotation joint. The slide has a prismatic joint, only allowing a one dimensional path in both ways. It is through these mechanisms that the rotational movement of the motor is transmitted to the slide.

This press has several dies, depending on the part to be produced it is necessary to use the appropriate die. All dies are of the progressive type, that is to say, the sheet enters with automatic feeding from coils, and is subject to successive operations

of cutting, folding or stamping until the finished part is obtained. For each press stroke, one part is finished.

Figure 2.4: *Haulick & Roos RVD 200T*

This press is in operation 24 hours, 5 days a week. Over the 24 hours it presents many periods of stops, on the one hand it presents planned stops, such as, start of shift meeting and die change, on the other hand, it presents unplanned stops. The purpose of the PdM system is focused on reducing unplanned stops by analyzing data from sensors installed on the equipment.

Currently, the *Haulick & Roos* press is subject to two types of maintenance, preventive and corrective. In Section 3.1 these terms are discussed in more depth.

Table 2.1 was elaborated in order to present the main characteristics of the press.

Table 2.1: Main characteristics of the press

Brand	*Haulick & Roos*
Type	Mechanical
Die Type	Progressive
Table Length	1550 mm
Table width	1200 mm
Stroke Length Available	20-160 mm
Minimum Cadence	30 ppm
Maximum Cadence	120 ppm
Maximum die length	1450 mm
Maximum die width	800 mm
Maximum die height	490 mm
Maximum die weight	1.5 T
PLC	Simatic S7 300

As shown in Table 2.1,the PLC Simatic S7 300 controls the press, this PLC and the press are a standard machine sold by *Haulick & Roos*. However, this machine is old and the PLC Simatic S7 300 does not have the possibility of connecting to a TCP/IP network, not even an RJ45 connection. Due to this limitation, Bosch has implemented a supervision system based on the Beckhoff CX2020 that has the capacity to:

1. Through digital connections to the Simatic S7 300;
   (a) Block the press;
   (b) Know the exact moment of a hit;
   (c) Know the machine's mode of operation.
2. Through the OPC-UA connection to Simatic S7 300;
   (a) Monitor all variables already measured by the old system;
   (b) Ensure that the machine is correctly parameterized.
3. Acquire new data through new sensors;
4. Assist the operator in his operations;
5. Confirm through RFID that the die placed is the correct one for the part to be produced;
6. Send all information in real-time to *Nexeed MES*.

All data collected from the press is sent to the Bosch MES, *Nexeed MES*, in an XML file. This file is sent whenever a part is finished, that is, when there is a press stroke. Usually, the time interval between strokes is 1 second. The data present in the XML file, sent by the PLC CX2020, has two different sources:

- Sensors implemented on the equipment and which communicate directly with the PLC CX2020;
- Communication with the PLC of the press (Simatic S7 300) via OPC-UA.

Regarding the first data source, this was implemented by Bosch Thermotechnology, S.A. with the aim of acquiring all the relevant information that can characterize the condition of the press. Sensors were placed on the engine, hydraulic tank, transmission shafts and on the electrical panel.

In the engine, two properties are measured, the temperature, Figure 2.5, and both the stator current and the rotor current, Figure 2.6.

In this press there are four hydraulic pumps, and in order to control the oil characteristics, there is a temperature sensor, and a level sensor in the hydraulic tank. In addition to the temperature and level measurement, there is also a pressure sensor for each hydraulic pump, as can be seen in Figure 2.7. Each pump also has a sensor that measures the current drawn.

For monitoring the vibration of the press there are four high precision accelerometers, two of which are on the left and right crankshaft bearings, and the other two are on the left and right connecting rod bearings. However, in the XML file, only one value

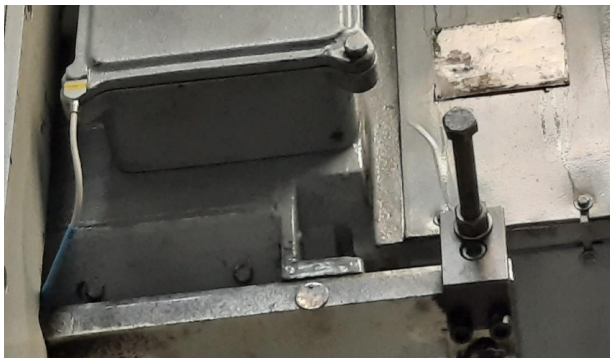
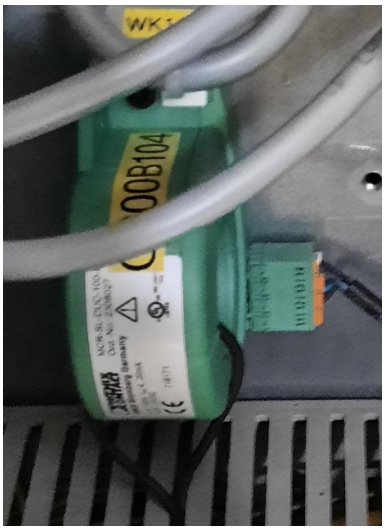


Figure 2.5: Engine temperature sensor



(a) Engine current rotor sensor

(b) Engine current stator Sensor

Figure 2.6: Engine current sensors

corresponding to the averages of these four measurements is sent. The device that performs this pre-processing is a vibration processing unit that is connected to the 4 accelerometers.

As for the electrical panel, there are two temperature sensors, one on each side of the panel. These sensors can be seen in Figure 2.8.

Regarding the second data source, this form was implemented in order to extract the variables measured by the PLC that controls the press. To this end, a device was installed in the equipment's electrical panel that allows communication via OPC UA. This device is connected to the PLC that controls the press and it is possible to extract the quantities measured by it. This communication via OPC-UA allows to obtain several variables among which, engine current speed, stroke speed, crankshaft position in degrees and die height.

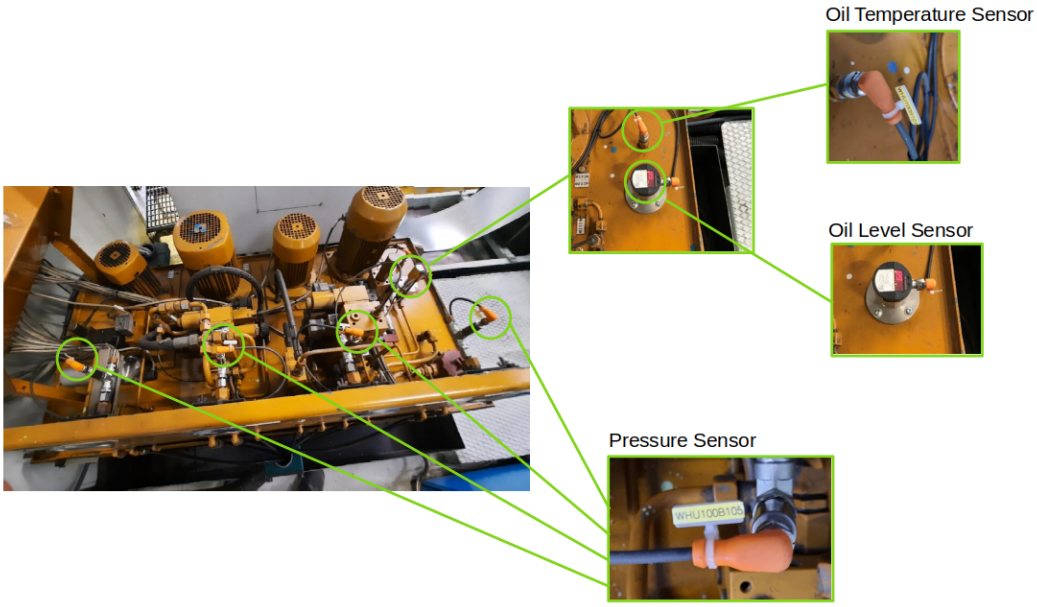


Figure 2.7: Sensors in the oil tank

(a) Temperature sensor on the left of the electrical panel

(b) Temperature sensor on the right of the electrical panel

Figure 2.8: Temperature sensors of the electrical panel

Chapter 3

State of the Art

In this chapter, all the concepts related to the project theme are addressed. Since the project aims to develop a platform that processes data from multiple equipment and implement a predictive maintenance system associated with the *Haulick & Roos* press, the first concept to address is *Predictive Maintenance*. The second concept to be addressed is *Time Series*, since, as will be explained further on, it plays a central role in the developed predictive maintenance system.

3.1 Predictive Maintenance

Machines, like humans, suffer from fatigue and exhaustion over time. If machines that show signs of "disease" (hot fumes, high vibrations, etc.) are not treated (maintenance) they will become less efficient, which will decrease both the quantity and the quality of the parts produced. The lack of careful maintenance can even cause the equipment to stop, which brings quite high costs.

The cost of a machine's breakdown is difficult to measure. There are studies that point to a value of around 4 to 15 times the cost of maintenance [Krar 1994]. In addition to the value of the maintenance cost itself, it is also necessary to add the cost of the downtime, parts that are not being produced and, in the worst case, orders that have not been fulfilled. According to [Chris Coleman *et al.* 2017], "poor maintenance strategies can reduce a plant's overall productive capacity between 5 and 20%."

At the beginning of the industrial revolution, those responsible for maintenance were the workers themselves. With the evolution of machinery and the increase in the complexity of the equipment, there was a need to create maintenance departments to deal with the problem [Jimenez-Cortadi *et al.* 2020].

From that time until now, there has been a lot of technological evolution, and today there are several ways to carry out maintenance without being dependent on the experience of an operator. The needs of each company determine the best way to maintain its equipment in order to optimize its resources.

Maintenance can be divided into three major branches, corrective, preventive and predictive. Each of these types of maintenance has its advantages and disadvantages and it is necessary to know these types well to see which type is most appropriate for the problem in question.

Corrective Maintenance (CM) has, as the name implies, a corrective character, that

is, it only acts after problems occur. As this maintenance only happens when a machine breaks down, there is always an uncertainty of the time remaining for the maintenance to occur, the maintenance time itself and the severity of the equipment failure. A feature in favor of this type of maintenance is the maximization of the equipment's lifetime, since it is used until failure occurs.

This type of maintenance is applied in processes where the stoppage of equipment does not have a major impact on production or in low-cost equipment in which, when a failure occurs, cheap maintenance or even replacement is possible, such as a lamp.

Figure 3.1 presents an sketch of how CM works.

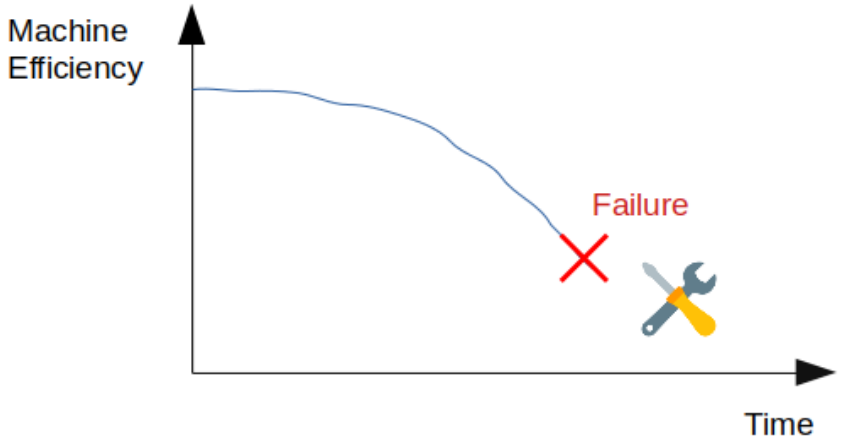


Figure 3.1: Sketch of CM

Preventive Maintenance (PM) attempts to prevent equipment failure by performing periodic maintenance. One of the major questions in this type of maintenance is the criteria for defining the interval between maintenance, as it is not possible to know when the rupture will occur, it is usually opted for short maintenance intervals, which implies wasting useful life of the equipment and an increase of the planned downtime.

Figure 3.2 presents an sketch of how PM works.

Predictive Maintenance (PdM) is a type of maintenance that acts before the problem occurs. It is based on sensor data, data analysis and *Machine Learning* (ML) techniques. The main objective of this type of maintenance is to maximize the time between maintenance operations and to minimize both the cost and the number of maintenance carried out on the equipment.

Figure 3.3 presents an sketch of how PdM works.

More and more companies are choosing this type of maintenance as the primary form of maintenance for high-value equipment for several reasons, one of which is the amount of devices capable of acquiring and sending data, IoT devices. Another reason is, according to [Chris Coleman *et al.* 2017], the fact that only recently the technology needed to perform predictive maintenance has become accessible.

The implementation of predictive maintenance has the advantages of both corrective maintenance and preventive maintenance without compromising the physical integrity of the equipment [Gonçalves Da Silva and Ferreira 2019]. According to [Chris Coleman *et al.* 2017] the most significant advantages of this type of maintenance are as follows:

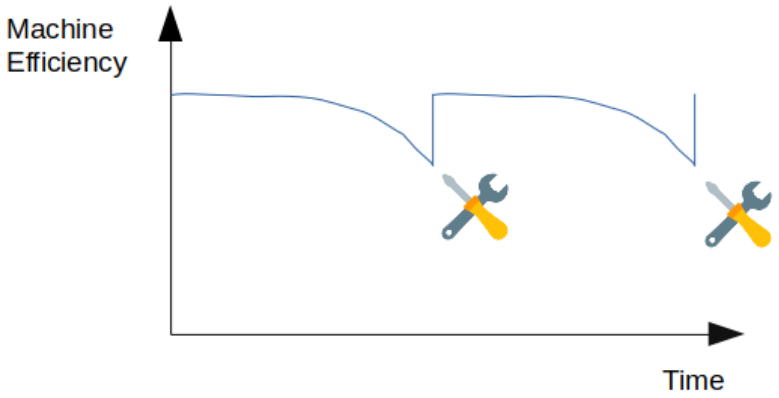


Figure 3.2: Sketch of PM

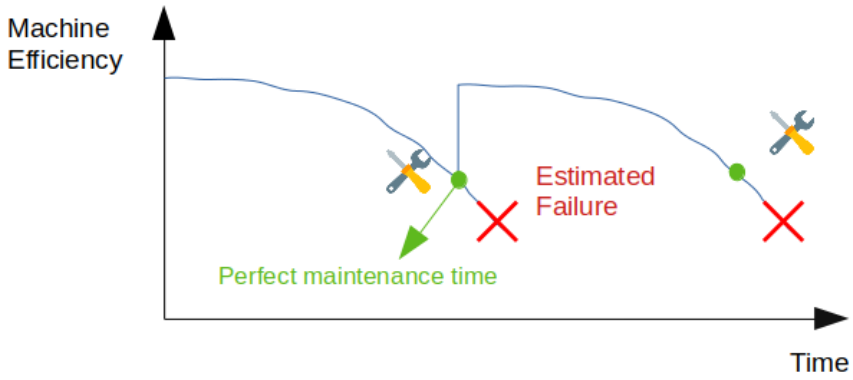


Figure 3.3: Sketch of PdM

- Increased equipment life (33-60%);
- Reduced maintenance costs (10-15%);
- Reduced time to plan maintenance (20-50%);
- Increased equipment uptime (10-20%).

PdM has, as explained above, very significant advantages over other types of maintenance. However, according to [J. M. Wetzer *et al.* 2000], to implement this system it is necessary to fulfill some requirements, such as:

- Existence of data that characterizes the operation of the equipment over a long period of time, containing data both under normal and abnormal operating conditions;
- Existence of indicators capable of defining the state of failure and degradation;
- Diagnostic tools needed to process data along with degradation indicators;

- Prediction tools in order to predict when the equipment will fail.

A predictive maintenance system requires the identification of input variables, usually sensor data, and identification of output targets, usually RUL (remaining useful life) or possibility of the machine working without breaking for a certain period of time or number of cycles.

With the purpose of building a predictive maintenance system that satisfies the previous conditions, there are some steps to be taken. First, it is necessary to place sensors that can characterize the behavior of the equipment, both in good conditions and in failure conditions. Then it is necessary to proceed with the acquisition of the physical quantities measured by these sensors. Afterwards, it is necessary to proceed with the storage of all acquired data, for this, it is important to carefully define the best data storage platform. Moreover, it is necessary to process the stored data in order to prepare the information for the next step. After data processing, comes one of the most important steps in this process, machine decision making, in which important characteristics are extracted and patterns are discovered in the data. At the end, there is the visualization of the data, in which both the raw data and the forecasts made are presented to the user [Santiago 2019].

To summarize the entire process, Figure 3.4 was made.

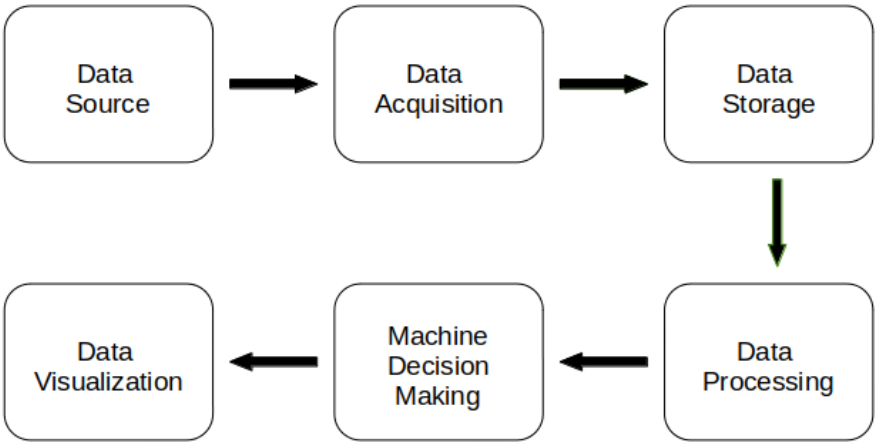


Figure 3.4: Steps of a PdM model

A detailed description of each step is now presented, as well as an indication of some algorithms most used in each step.

3.1.1 Data Source

The first step to be taken when developing a predictive maintenance model is to study the monitoring techniques that are capable of characterizing the operation of the equipment. For mechanical systems the most used monitoring techniques are [Hussien and Alla 2018]:

- Vibration signature analysis (VSA);
- Acoustic emission testing (AET);

- Infrared thermography (IRT);
- Lubrication oil analysis (LOA);
- Ultrasonic condition monitoring (UCM).

Vibration signature analysis (VSA) is based on continuous measurement of acceleration over time and is widely used to determine the general condition of an equipment. The vibration analysis detects repetitive motions of parts in rotation or in oscillatory periods. This technique is able to detect many problems, such as unbalance, misalignment, rolling element bearing faults and electrical effects [Hussien and Alla 2018]. This technique is the most suitable for predictive maintenance problems in which the equipment has rotating parts, such as, for example, hydraulic pumps or stamping presses.

Acoustic emission testing (AET) is a technique used to analyze sound waves from equipment that has defects or discontinuities. All equipment has some type of noise, and through the analysis of noise over time it is possible to detect that at some point the frequency or amplitude of the noise suffered a variation, and this variation may have been caused by some change in the normal operation of the equipment. Fatigue, friction, loss of material, cavitations and leakage are some examples of problems that can be detected using this technique [Hussien and Alla 2018].

According to [Hussien and Alla 2018], "Temperature is one of the most common indicators of the structural and functional health of equipment and components.", analyzing the way in which temperature varies over time allows the detection of a wide variety of defects, such as physical damage to components and problems with electrical connections. Infrared thermography (IRT) is the process of using thermal images to capture infrared radiation emitted by objects [Chou and Yao 2009]. Analyzing these images over time, it is possible to detect variations that may induce that there is or will be a problem with the equipment.

Lubrication oil is used to reduce friction between moving surfaces. Lubricating oil analysis (LOA) can provide warnings about a defect in the equipment 10 times earliar than vibrations analysis [Zhu *et al.* 2013]. LOA includes a series of analyzes, such as analysis of fluid viscosity, oxidative properties and contamination of the fluid, such as metallic particles, air, etc. [Hussien and Alla 2018].

Ultrasound, according to [Rienstra and Hall 2002] is defined as "sound waves having a frequency above the limits of human hearing, or in excess of 20,000 cycles per second ". Several physical events cause sound at both auditory and ultrasonic frequencies, and the analysis of these frequencies can detect the existence of a problem in the equipment. Ultrasonic condition monitoring (UCM) is a technique that uses contact and non-contact instruments to detect high frequency ultrasonic emissions in order to detect the health status of the equipment. This technique has the ability to detect problems at the level of bearings, lubrication issues, gear damage, pump cavitations and leaks [Hussien and Alla 2018].

3.1.2 Data Acquisition

Data acquisition is the process of collecting and storing the measurements made by the sensors. For this step, it is necessary to know the communication protocols of the installed sensors, and to program data acquisition devices, such as micro controllers,

PLC's or other devices to communicate with the sensors, respecting the rules of the protocols. After the data is in the data acquisition devices, it is necessary to send the data to a database, where it will be stored and later processed.

The acquisition of data in a predictive maintenance system has to be carried out with high frequencies, so that it is possible to analyze the equipment's operation in great detail and, consequently, produce forecasts with greater validity.

3.1.3 Data Storage

Typically, target systems for predictive maintenance are systems that produce data over time. This type of data, as will be explained in Section 3.2, is called time series. For this reason, a *Time Series Database* (TSDB) is commonly used. These types of databases are optimized for time series data.

There are many types of TSDB, each with its advantages and disadvantages. However, regarding popularity there is one that distances itself from the others, *InfluxDB*.

Figure 3.5, taken from the official page of *db-engines*, shows the popularity of *InfluxDB* compared to other types.

Rank			DBMS	Database Model	Score		
Jan 2021	Dec 2020	Jan 2020			Jan 2021	Dec 2020	Jan 2020
1.	1.	1.	InfluxDB	Time Series	26.32	+0.17	+5.18
2.	2.	2.	Kdb+	Time Series, Multi-model	7.96	+0.30	+2.46
3.	3.	3.	Prometheus	Time Series	5.71	-0.04	+1.63
4.	4.	4.	Graphite	Time Series	4.68	+0.01	+1.33
5.	5.	5.	RRDtool	Time Series	3.19	-0.11	+0.42
6.	6.	7.	TimescaleDB	Time Series, Multi-model	2.93	-0.05	+1.01
7.	7.	8.	Apache Druid	Multi-model	2.67	+0.08	+0.79
8.	8.	6.	OpenTSDB	Time Series	2.35	-0.10	+0.38
9.	9.	9.	FaunaDB	Multi-model	1.91	-0.11	+1.11
10.	10.	11.	GridDB	Time Series, Multi-model	0.83	+0.01	+0.30

Figure 3.5: Ranking of the various types of TSDB [DB-Engine 2021]

However, for time series data, the best solution is not always a TSDB, sometimes the best solution is a relational database. There are several factors that influence this choice, such as scalability, query complexity and data model flexibility.

3.1.4 Data Processing

Before sending the data to the next steps, it is necessary to perform data processing, both to increase the speed of the process and to obtain more reliable results. There are two approaches to data processing, batch processing and stream processing.

Batch processing integrates all the data in one package and processes them all at once [Martin *et al.* 2015]. This type of processing requires great computational capacity and high storage memory.

Stream processing operates over a continuous stream of data arriving at high speed, processing only one item or one batch of items each time [Tank 2021]. This type of processing requires less computational capacity than the type of processing explained above.

Data processing is one of the most important steps in a predictive maintenance model. The data that comes directly from the sensors usually comes with a lot of noise, with several values missing and in many of the cases, it comes with information that does not add any value to the forecasts to be made, therefore it is absolutely necessary to proceed with the study and treatment of the data from the sensors.

Data processing can be divided into three major categories, data cleaning, data transformation and data reduction [Jimenez-Cortadi *et al.* 2020]. Below is an in-depth explanation of each of these categories, explaining their goals as well as the methods that exist to achieve those same goals.

Data Cleaning

The raw data may be incomplete, there may be data influenced by the actions of the operators, sensor failures, etc. Because of this, it is extremely important to perform a data cleaning process to facilitate the work of machine decision making.

The biggest problems that are the focus of this category are missing data and the presence of outliers/anomalies.

The presence of missing values is a very common feature in most datasets. In the case of predictive maintenance, the most common causes are loss of communication with the sensor, data sources with irregular updates, among others.

According to [Mack *et al.* 2018], it is possible to categorize missing data into three types:

- **Missing completely at random (MCAR)**;

  The probability that the data is missing is the same for all observations. That is, there is no relationship between the missing data and the other values in the dataset.

- **Missing at random (MAR)**;

  The probability of an observation being missing depends on available information in other variables.

- **Missing not at random (MNAR)**.

  There is a mechanism or reason why data is not being inserted into the dataset.

Understanding the reason why the data is missing helps to understand which is the best method to deal with it.

There are three solutions to deal with missing data, eliminate observations in which there is missing data, univariate imputation and multivariate imputation.

In the first case, only observations that present values for all variables are studied, the rest are ignored. One of the great advantages of this method is the preservation of the distribution of the variables, however, the major disadvantage is the possibility of eliminating a large part of the dataset if the percentage of missing data is high.

In the case of univariate imputation, the average or median of a given variable is usually used to estimate all missing data associated with that variable. The advantage of this method over the previous one is that in this case there is no removal of observations, however, it has as a disadvantage the distortion of the original distribution of the variable.

Finally, multivariate imputation techniques make the data imputation using the information present in the remaining variables considering the same observation. Examples of this method include K-Nearest Neighbors (KNN) imputation which average the value of the K closest samples in order to predict the missing value, or multivariate imputation of chained equations (MICE) where "a series of models whereby each variable is modeled conditional upon the other variables in the data " [Galli 2020].

After the missing data problem is solved, it is important to proceed with the identification of the outliers to decide how they will be processed. Outliers are "extreme values that abnormally lie outside the overall pattern of a distribution of variables " [Kwak and Kim 2017]. Outliers can have quite negative effects on some ML models, since they introduce bias in the statistical representation of the data. According to [Galli 2020], there are 3 techniques to detect outliers:

- **Gaussian distribution**;

  This technique assumes that the data have a normal distribution. For a given value not to be considered an outlier, the following comparison must be valid.

$$\mu - 2 \cdot \sigma < x < \mu + 2 \cdot \sigma \tag{3.1}$$

  Where x is the value being checked whether it is outlier or not, μ is the mean, and σ is the standard deviation.

- **Interquartile range (IQR)**;

  This technique is a general approach in which the IQR is calculated according to the following expression.

$$IQR = 75^{th}Quantile - 25^{th}Quantile \tag{3.2}$$

  For a value not to be considered an outlier, the following comparison has to be fulfilled.

$$25^{th}Quantile - 1.5 \cdot IQR < x < 75^{th}Quantile + 1.5 \cdot IQR \tag{3.3}$$

- **Quantiles**.

  In this technique, only $95^{th}Quantile$ and $5^{th}Quantile$ are calculated. Here, for a value not to be considered an outlier, the following comparison must be valid.

$$5^{th}Quantile < x < 95^{th}Quantile \tag{3.4}$$

There are several methods to treat outliers, the most common approaches involve removing outliers from observations, defining upper and lower limits for each variable and replacing outliers by an estimation. There is also the possibility to apply non-linear transformations, such as logarithm or sigmoid functions [Filipe and Gameiro 2020].

It is important to note that not every value that is outlier is an abnormal value to the behavior of the data, that is, there are situations in which the expected value is in fact an outlier. Before using any outlier removal technique, it is important to study why the outlier exists, so as not to remove important data from the dataset.

Data Transformation

After the data cleaning step is completed, it is necessary to manipulate the data taking into account the ML model that will be used. According to [Galli 2020], there are three major transformations that can be made, variable transformation, discretization and feature scaling. It is not always necessary to carry out the three transformations, everything depends on the type of ML model that will be used and the type of data that exist. In the next paragraphs, each of these transformations will be explained.

ML linear models assume that the variables follow a Gaussian distribution, however most of the time the variables have a skewed distribution. It is possible by applying a mathematical transformation to transform a skewed distribution into a distribution that looks more like a Gaussian distribution [Galli 2020]. Logarithmic and exponential are two of the most used mathematical transformations. An example of the effect of the logarithmic transformation is shown in Figure 3.6.

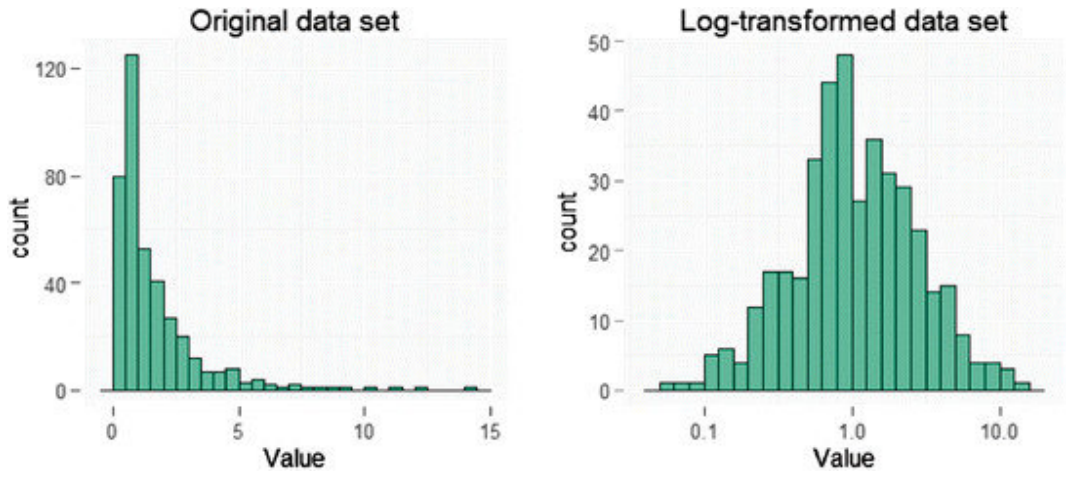


Figure 3.6: Example of the effect of a logarithmic transformation on the distribution of a dataset [Komorowski *et al.* 2016]

Discretization is the process of transforming continuous variables into discrete variables by creating a set of contiguous interval that span the range of the variable's values. This technique is also called binning, where each bin corresponds to each interval. Discretization manages to increase the homogeneity of the distribution over the different intervals [Galli 2020]. In addition to improving the homogeneity of the distribution, it is also able to deal with outliers, placing them in the lower or upper range. The most common discretization techniques are equal-width, in which the width of each interval is assumed to be constant, and equal-frequency, in which each interval is guaranteed to have the same frequency of observations [Han *et al.* 2011].

The magnitude of the variable values does not normally come in the same range, especially in predictive maintenance models where it is possible to have sensor values with some orders of magnitude higher than others. This particularity is very important because it affects most ML models, such as linear and logistic regression, neural networks, support vector machines (SVM), KNN, linear discriminant analysis (LDA) and principal component analysis (PCA) [Galli 2020]. The only models that are insensitive to the magnitude of the variations are models based on trees.

The purpose of feature scaling is to normalize the range of values of the different variables. There are many techniques associated with feature scaling, but according to [Galli 2020], the most popular are standardization and scaling to minimum and maximum. The standardization method calculates the mean and standard deviation of the variable in question and replaces the value of each observation with the following expression:

$$x_{scaled} = \frac{x - \mu}{\sigma} \tag{3.5}$$

Where x is the observation value before standardization and x_{scaled} is the value of that observation after standardization. This technique centers the mean of the variable at 0, scales the variance at 1 and preserves the distribution of the variable.

Scaling to minimum and maximum only calculates the minimum and maximum value of each variable, min and max respectively, and applies the following expression:

$$x_{scaled} = \frac{x - min}{max - min} \tag{3.6}$$

The result of this transformation are values between 0 and 1. In this case, the mean and variance of the distribution varies, and it is possible that the shape of the distribution will also be changed [Galli 2020].

Data Reduction

In this type of process, there is usually a lot of data available, most of which are redundant. Implementing predictive algorithms on huge amount of data can take a long time and require a lot of computational resources.

Data reduction techniques aim to obtain a lower data volume, maintaining the integrity and all patterns of the initial data. If the data reduction is well implemented, it is possible to obtain better results in machine decision making using the reduced data than those that would be obtained using all the initial data [Han *et al.* 2011].

There are three types of strategies with regard to data reduction, dimensionality reduction, numerosity reduction and data compression.

Dimensionality reduction is the process of reducing the number of variables that are being analyzed. The two most used techniques are discrete wavelet transform (DWT) and principal component analysis (PCA) [Han *et al.* 2011].

DWT is a linear signal processing technique that transforms a vector X into a numerically different vector X'. This new vector X' presents wavelet coefficients in their positions. Although the two vectors have the same number of elements, the vector with the wavelet coefficients can be truncated. It is possible to obtain a high approximation of the data with only a small fraction of the wavelet coefficients [Han *et al.* 2011].

PCA captures the essence of the data in some major components. Principal components are the underlying structure in the data, are the directions in which there is greater variance, where the values are more distributed. PCA is able to reduce data in its basic components, eliminating all redundant dimensions [Han *et al.* 2011].

Numerosity reduction techniques replace the initial data with alternative ways of representing the data with less volume. These techniques can be parametric or non-parametric. In parametric techniques, a model is used to estimate the data, after which the data is replaced by the model parameters. Regression models can be used for this purpose. Non-parametric methods use other ways to replace the data, such as sampling

or data aggregation [Han *et al.* 2011]. The sampling technique allows, from a high volume dataset, to extract a smaller dataset at random. Data aggregation is a widely used technique because it groups data over a given period of time by extracting the most relevant properties from that period of time, significantly reducing the size of the dataset.

In data compression, transformations are applied to the initial data in order to obtain a reduced representation of the data. If it is possible to reconstruct the original data from the reduced version, it is said that the reduction of the data is lossless. On the other hand, if only part of the initial data can be reconstructed, it is said that the reduction of the data is lossy. The other two data reduction strategies discussed above are sometimes considered types of data compression [Han *et al.* 2011].

3.1.5 Machine Decision Making

Machine decision making corresponds to the stage where data features are extracted. These features allow to obtain relations between the input data and the forecasts.

This stage can be divided into two broad categories, diagnostic and prognostic. Diagnostic focuses on pattern detection and failure identification. Prognostic focuses on failure prediction based on the patterns detected in the previous step. These two categories are closely connected, the diagnosis adds new information to the process. It is this new information that allows the transformation of an unsupervised problem into a supervised problem [Jimenez-Cortadi *et al.* 2020]. It is then possible to summarize this interaction by stating that a better diagnosis will naturally promote a better prognosis.

Diagnostics

Diagnostics deals with quantifying the damage that has occurred. This category can be divided into three stages [Efthymiou *et al.* 2012]:

- **Fault detection**;

  Responsible for detecting and reporting an anomaly in the operation of equipment.

- **Fault isolation**;

  Responsible for identifying which component is about to fail or will fail.

- **Fault identification**.

  Responsible for assessing the causes and magnitude of the failure.

In order to infer these relationships, complex algorithms, such as ML algorithms, are needed. Some of the most used algorithms in diagnostics are: Linear regression, random forest, Markov models, artificial neural networks (ANN) and support vector machines [Jimenez-Cortadi *et al.* 2020].

Prognostics

The prediction of when the equipment will fail is a task for this phase. Prognostics makes predictions about the future behavior of the equipment based on the analysis of the past behavior.

In this final step of the predictive maintenance there are two common approaches, classification approach and regression approach [Perera and Alwis 2017].

Classification approach predicts the possibility of failure in the next times or cycles. It only provides a boolean forecast, however it can be highly accurate. Regression approach predicts the RUL (remaining useful life) which means that it predicts how long or how many cycles are left until the rupture occurs. This forecast is, usually, less accurate than the previous.

Time series analysis using condition monitoring data is essential to anticipate anomalies in a predictive maintenance process, being one of the most used methods in the prognostic phase. The approaches in this type of analysis focus on a model that best fits the time series to later calculate the error between its predictions and the actual data. The most traditional strategies use statistical data, such as autoregressive integrated moving average (ARIMA) and Kalman filter [Diez-Olivan *et al.* 2019].

Another approach widely used in the prognostic phase is Deep Learning (DL). DL methods can be interpreted "as a cascade of many layers of processing units that combine the predictor features to approximate the target feature " [Diez-Olivan *et al.* 2019]. Recurrent neural networks (RNN) are a type of DL algorithm in which their great particularity is to process a sequence of inputs and save their state while processing the next sequence of inputs. This particularity is extremely important for forecasting time series since it allows to store information along the loops [Mikolov *et al.* 2010]. RNNs are becoming increasingly popular for their ability to learn from varying length sequence data, particularly those using long short-therm memory (LSTM) hidden units [Hochreiter 1997]. [Guo *et al.* 2017], developed a methodology using LSTM to predict the RUL of bearings and obtained excellent results.

3.1.6 Data Visualization

As PdM systems generate high amounts of data per day, visualization is a very important part of this system because it allows the analysis of a lot of information in a simple way using graphical tools [Santiago 2019].

The main focus of this stage is the presentation of diagnostics, prognosis and maintenance planning in a simple and intuitive way. In this way, the maintenance team is provided with an indication of the maintenance problem, its cause and its solution through simple tools, such as a smartphone [Efthymiou *et al.* 2012].

Figure 3.7 and Figure 3.8 shows an example of a dashboard implemented by Splunk Technology, in which the objective was to predict the condition of the engines that were in operation, classifying them as normal engines, warning engines and critical engines. In this way it is possible to view the percentage of engines that are in trouble, to identify which engines need immediate attention and based on the RUL to estimate the remaining useful life of each engine.

3.2 Time Series

Data can be divided into two major types, time series and cross-sectional data [Forjan 2019].

Time series data is a set of observations taken over a period of time, the existence of time on the horizontal axis makes this type of data different from the other. The

Figure 3.7: Identification of the status of each engine [Cho 2019]

maximum temperature in Aveiro over the past 20 years is an example of this type of data. Cross-sectional data consists of a set of observations at a single point in time. In this type of data, unlike time series data, there is no order in the observations. The temperature of Aveiro, Viseu and Lisbon on 11/03/2021 is an example of this type of data.

In the next subsections the topic time series will be covered in depth, explaining what they are, their decomposition, how to classify them, their properties and finally explaining why to analyze this type of data and how to do it.

3.2.1 What is a Time Series?

A time series is a sequence of data points measured over a period of time, Figure 3.9. These data points are usually obtained through the observation/measurement of quantifiable variables [Cochrane 2005]. The time interval between successive observations is usually constant, however, there are situations in which it is not, this characteristic has an influence on the characterization in the type of time series, as will be discussed in more depth below.

3.2.2 Components of a Time Series

In general, a time series consists of four distinct components that can be extracted from the observed data [Agrawal and Ratnadip 2013]. These four components are: *Trend*, *Seasonal*, *Cyclical*, and *Random*. A more detailed explanation of each of these components will be provided in the next paragraphs.

The *Trend* component reflects the general tendency that the time series has to increase, decrease or stagnate over a long period of time. It can be assumed that *Trend* is a long-term movement of the time series. Population growth over the past hundred

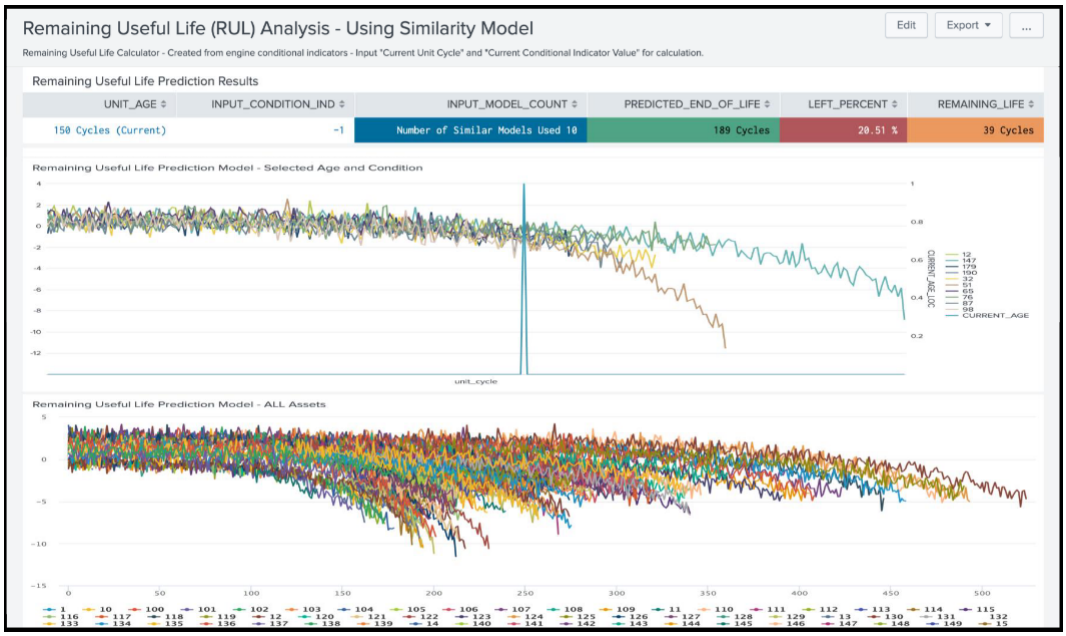


Figure 3.8: RUL identification for each engine [Cho 2019]

years clearly shows an increasing trend, while the number of people who die before the age of sixty shows a decreasing trend, given that average life expectancy continues to increase.

Seasonal component is a component that is repeated periodically. There are many factors that can cause seasonal variations, such as atmospheric conditions, traditional habits, etc. A good example is sales, sales from year to year may increase, however there is a seasonal component that represents significant sales growth in December and a decrease in August.

The *cyclical* variation represents medium term changes in the time series, caused by events that are repeated in cycles. The cyclical component has a longer lasting effect than seasonal component and can vary from cycle to cycle. According to [Ostashchuk 2017], the cyclical component is often integrated with the *Trend* component as it also has medium/long-term effects.

Random or *Irregular* variations in a time series are caused by unpredictable events, which do not behave regularly or are repeated in a certain pattern. There are methods of time series analysis that focus on filtering these random variations to detect the other components. Earthquake can be considered an event that causes random variations.

Figure 3.10 shows an example of the decomposition of time series into components using the additive model that will be explained in the next paragraph. In this case, as explained earlier, there was a union between the trend component and the cyclical component.

In addition to visualizing each of the components of a time series, it is also important to describe mathematically how these components interact with each other. According to [Agrawal and Ratnadip 2013], there are two models that are generally used in time series, the additive model and the multiplicative model.

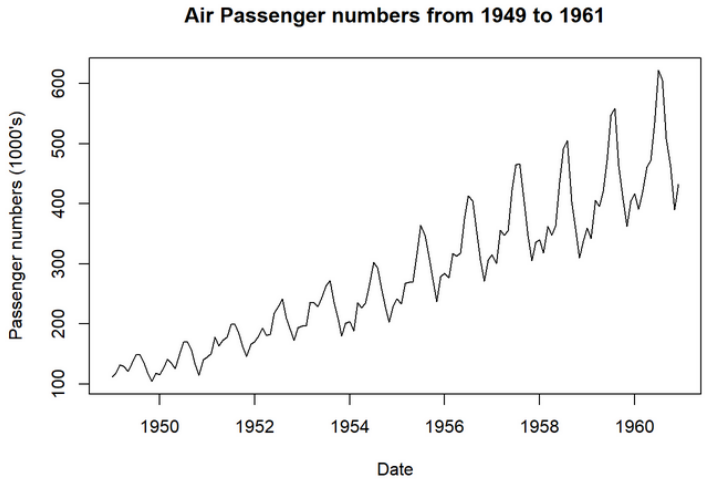


Figure 3.9: Airline passengers over time [Futter 2017]

- Additive model

$$Y_t = T_t + S_t + C_t + R_t \tag{3.7}$$

- Multiplicative model

$$Y_t = T_t * S_t * C_t * R_t \tag{3.8}$$

Y_t is the observation made, T_t is the *Trend* variation at time t, S_t is the *Seasonal* variation at time t, C_t is the *Cyclical* variation at time t and finally, R_t is the *Random* variation at time t.

The additive model assumes that the four components are independent of each other, while the multiplicative model assumes that the four components are not necessarily independent of each other and that, therefore, they can affect each other.

3.2.3 Classification of Time Series

There are several ways to classify time series depending on the criteria used. In the next paragraphs, some of the most significant ones will be addressed, such as, number of variables, regularity, seasonality, stationarity and autocorrelation.

Time series can be categorized into two major classes, univariate or multivariate. A univariate time series is a set of measured values related to a single variable over time. When a time series involves more than one variable to be measured, the series is said to be multivariate. Multivariate time series can be further categorized into homogeneous and heterogeneous depending on the relationship between the variables [Fawumi 2015]. If one of the variables is useful to predict the future of another variable in the time series, it is said that the series is homogeneous, if this does not happen, the series is said to be heterogeneous. In homogeneous series, if a value of one variable is changed, the value of another variable will be changed as well, given the relationship between the variables.

Regularity is another property that distinguishes two types of series, regular or irregular. The difference between these two types of series is the duration between timestamps of two successive measurements. All regular series have equal times between two successive timestamps, while irregular series have times that can vary. For example, the

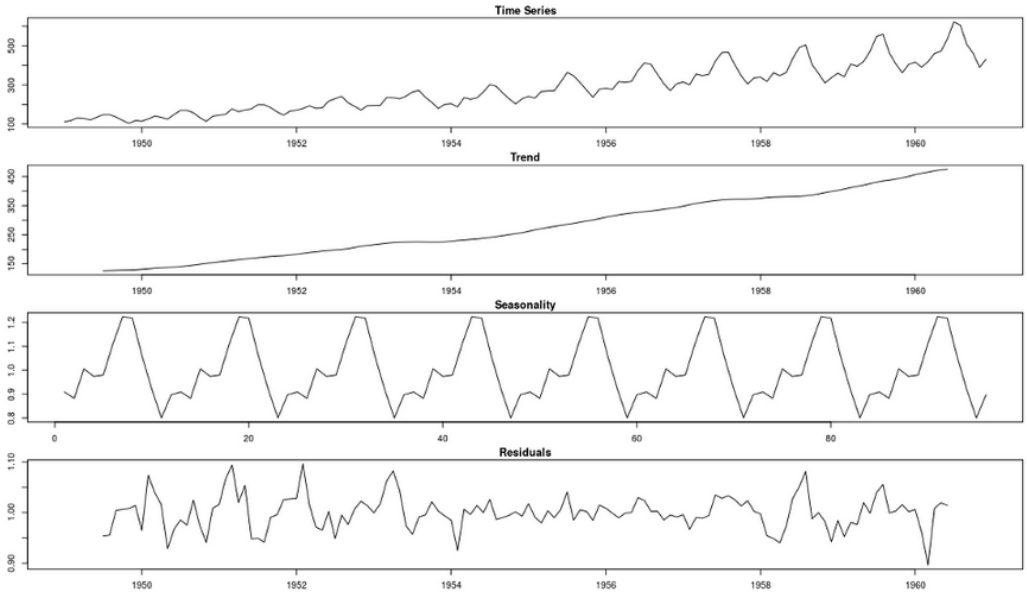


Figure 3.10: Time series components [Ostashchuk 2017]

measurement of the acceleration of a shaft by an accelerometer with a frequency of 10 Hz is an example of a regular time series, while for example, the stock price does not show measurements at regular time intervals and is, therefore, an irregular time series.

Time series as to seasonality can be considered seasonal or non-seasonal. A seasonal time series is a series where it is possible to observe a repetitive pattern over time. As mentioned earlier, seasonal variation is in many cases linked to variations over the course of a year, such as changes in sales in December due to Christmas.

Another classification of time series is based on the property stationarity. In this case, the series can be stationary or non-stationary. A stationary time series is a time series in which statistical properties such as mean, variance and covariance remain constant over time. These series show a balance with respect to their mean over time, even with fluctuations, their average value remains relatively constant.

A time series can also be classified based on the rate of dependency between new observations and their predecessors [Ostashchuk 2017]. A very important concept in this situation is autocorrelation. The autocorrelation function calculates the correlations between the time series and copies of it shifted at different points in time. The investigation of autocorrelations in a time series allows the detection of important dependencies in the series, allowing to verify if the time series presents a behavior that can be predicted or only shows a random behavior. According to autocorrelation, a time series can be divided into two types, long memory time series and short memory time series. Long memory time series are time series in which the autocorrelation function decreases slowly, meaning that there is a strong relationship between the posterior and anterior values at different time points. On the other hand, short memory time series are those in which the autocorrelation function decreases rapidly, with little or none relationship between the previous and the later data. This type of time series is not suited to the concept of predictive maintenance, since in predictive maintenance one of the premises is that the value of sensors over time follows a certain pattern and when a failure occurs

this pattern is changed.

In order to summarize all types of times series classifications, Figure 3.11 was elaborated.

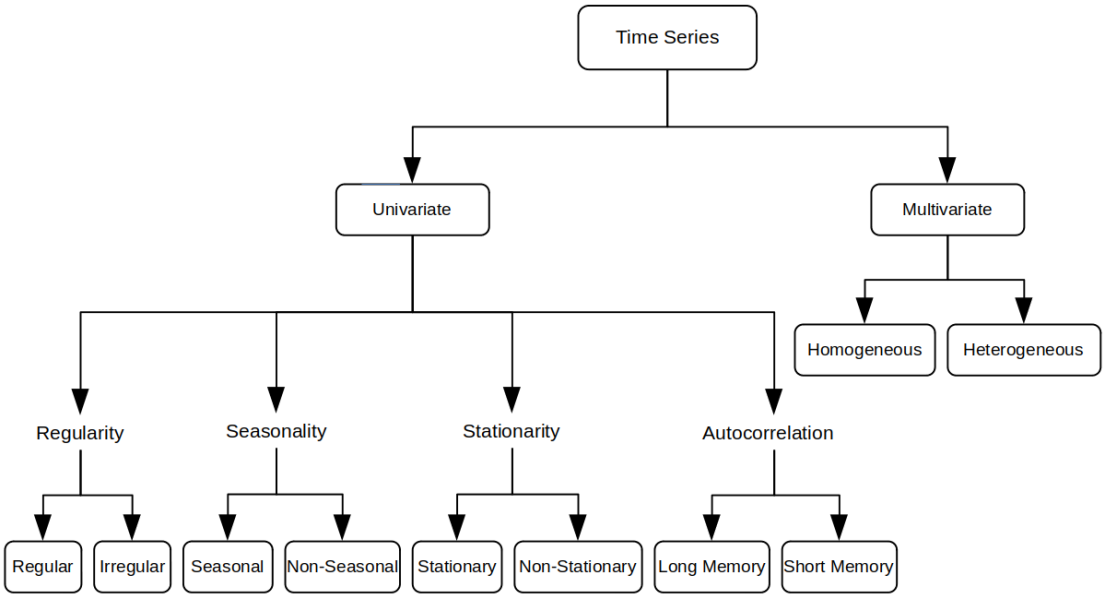


Figure 3.11: Classification of time series

3.2.4 Time Series Analysis

Time series data are becoming very valuable in modern organizations. With *Industry 4.0* there are more and more sensors capturing information of various equipment and it is extremely important to process this data and analyze it, both to understand what has happened and what will happen in the future. This is where time series analysis comes in. This subsection explains the objectives and processes of time series analysis.

Time series analysis is based on the fact that the values measured over time have some internal structure, such as, for example, autocorrelation, trend or seasonal variation [Fawumi 2015]. In this analysis, the methods used are not only aimed at extracting statistical properties from the data but also at forecasting. Forecasting is applying a model to predict future values based on the analysis of values measured in the past.

To perform time series analysis, it is necessary to create a hypothetical probability representation of the data [Seymour *et al.* 1997], in other words, it is necessary to create a model. After the model has been determined, it is possible to adjust the model to the data, adjust parameters and, if possible, use the model to predict the next data. This information has a very significant value, for example in the case of the manufacturing industry it may be possible based on the predicted data to verify that a problem will occur and to act in advance, this is the purpose of predictive maintenance.

Objectives of Time Series Analysis

The focus in time series analysis is to determine a model that can describe the data patterns. When this task is completed, it is possible to use the model in different ways,

such as:

- Identify in the observed data statistical and other relevant properties;
- Predict future values of the time series;
- In manufacturing industries using knowledge about future data in order to decide the maintenance date of the equipment.

The objective of creating a time series model is exactly the same as that of a generic predictive model, which is basically to decrease the error between the predictions of the values with the future values [Fawumi 2015]. However, there is a significant difference in the way the model is built. In the case of a time series model, the lag values of the target variable are used to predict the next values of that same variable, whereas in traditional models, other variables are used as predictors. The concept of lag value does not even apply to traditional models since the observations do not have a chronological order [Shumway and Stoffer 2010].

Time Series Models

In general, there is a pipeline for building a time series analysis and forecasting model [Fawumi 2015]:

1. Pre-process the data. Proceed with methods of data imputation, observation of data, determination of trends and seasonality, filtering noise from data, etc.;
2. Determine a model that fits the pre-processed data;
3. Fit the model parameters to the data. In this step, the model adjusts its parameters in order to reduce the error between predictions and future values;
4. Perform prediction and forecasting time series values.

There are several models for time series analysis, however, they can be categorized into three major classes: *Autoregressive* (AR) models, *Integrated* (I) models, and *Moving Average* (MA) models [Fawumi 2015]. These three classes of models depend linearly on previous data. There are other models, such as *Autoregressive Moving Average* (ARMA), *Autoregressive Integrated Moving Average* (ARIMA) and *Seasonal Autoregressive Integrated Moving Average* (SARIMA), that are basically a combination of the three classes mentioned above. In the following paragraphs, each of these models will be addressed.

In an AR model, the value of the time series at time t, Y_t, depends linearly on the previous values of the time series and on the random disturbance occurring at time t. The parameter p of AR(p) represents the number of lags that will be used to predict the value of Y_t. This model presents the assumption that the time series has to be stationary. In an AR model of order p, the model can be written according to the following expressions [Cang and Seetaram 2012]:

$$Y_t = \epsilon_t + \phi_1 \cdot Y_{t-1} + \phi_2 \cdot Y_{t-2} + ... + \phi_p \cdot Y_{t-p} \tag{3.9}$$

$$Y_t = \epsilon_t + \sum_{k=1}^{p} \phi_k \cdot Y_{t-k} \tag{3.10}$$

Where ϕ_1,ϕ_2 and ϕ_{t-p} are parameters coefficients, Y_{t-1}, Y_{t-2}, Y_{t-p} are previous values of the time series Y and ϵ_t is a random error term uncorrelated over time, typically called white noise.

The MA model is similar to the AR model, except that it uses the average of the most recent q values and instead of performing a regression using the values of the previous observations, it performs a regression using the residuals, that is, the difference between the average and the observed values [Baccar 2019]. Like the AR model, the MA model assumes that the time series to be analyzed has a stationary behavior. This model can be written according to the following equations [Cang and Seetaram 2012]:

$$Y_t = \epsilon_t + \theta_1 \cdot \epsilon_{q-1} + \theta_2 \cdot \epsilon_{q-2} + ... + \theta_q \cdot \epsilon_{t-q} \tag{3.11}$$

$$Y_t = \epsilon_t + \sum_{k=1}^{q} \theta_k \cdot \epsilon_{t-k} \tag{3.12}$$

Where θ_1, θ_2, θ_q are parameters coefficients, ϵ_t is white noise and ϵ_{t-1}, ϵ_{t-2} and ϵ_{t-p} are the residuals.

The ARMA(p, q) model is the combination of the models explained above, AR(p) and MA(q), mathematically it is simply the sum of the two expressions, as indicated by the following expression:

$$Y_t = \epsilon_t + \sum_{i=1}^{p} \phi_i \cdot Y_{t-i} + \sum_{i=1}^{q} \theta_i \cdot \epsilon_{t-i} \tag{3.13}$$

The disadvantage of ARMA is the assumption that the time series has to be stationary, since it results from two models that also have this limitation.

The ARIMA model emerged to circumvent the limitation of the previous models. An ARIMA model can be characterized by three parameters, order of the AR(p), order of the MA(q) and the number of differencing (d) required to make the time series stationary. The most important feature of the ARIMA model is that it can handle non-stationary time series thanks to parameter d. It can be said that the ARIMA model is the application of the ARMA model to the time series differentiated d times [Baccar 2019]. Although the ARIMA model can handle a greater range of time series than previous models, it still lacks the ability to deal with seasonality in the time series.

The SARIMA model is a generalization of ARIMA in which it adds four more parameters (P, D, Q, s) in order to deal with seasonal time series.

Choosing the type of model to analyze a time series requires a very thorough study of the time series given that each model depends on the underlying attributes of the time series.

Intentionally blank page.

Chapter 4

Proposed Solution

This chapter is divided into three sections and has the following objectives: enumerate which problems the proposed solution intends to solve, explain in general terms the proposed solution, and lastly, address the implemented solution indicating all the software and hardware used.

4.1 Problem Statement

The developed project arose from the need to solve three major problems. A description of these problems, as well as proposed solutions to address them, are presented below in the form of a list.

1. **Existence of several recipients for the data produced by IoT devices**

   With the increase of IoT Devices installed throughout the manufacturing facility, there is more and more data that has to be stored for further processing. One of the big problems with this data increase is the existence of several recipients, leading to a divergence in data storage. This divergence is often verified when there is some equipment data stored in a company database and other data are stored in local databases. This data distribution causes a major obstacle when trying to perform data analysis or data visualization due to the difficulty of gathering all data together.

   Another disadvantage of having multiple recipients is the desynchronization of timestamps. As there are parallel data flows, the entities that apply the timestamps are different, and there is the possibility of applying the timestamps relative to another temporal reference. Later, when performing a global data analysis, there are timestamps with different references, which makes the analysis difficult.

   In order to solve this problem, a system capable of converging data from all equipment is proposed. This system receives data in real-time from several IoT devices installed in different equipment, storing all data based on their origin. In this way, all data is in a given database, making it possible to access data from any equipment in the same place.

2. **A high number of unplanned stops at the *Haulick & Roos* press**

   In the last few years, the *Haulick & Roos* press has had several breakdowns which has a very significant financial impact, both in the repair of the equipment and in the downtime of producing parts to feed the production lines.

   In order to reduce the number of unplanned stops on the press, a predictive maintenance system is proposed, which runs in real-time on the server, sending alerts when there is an abnormal behavior of the equipment.

3. **Data visualization platform with several limitations**

   The data visualization platform currently implemented in the company has several limitations, such as low intuitiveness, difficulty in adjusting the time in which the data is to be viewed, few functionalities implemented, among others.

   This project also involves the development of a data visualization platform that allows, among others features, the visualization of all data from all equipment, checking the operating status (stopped or working) of all equipment, adjusting the time window visualization of the data in a simple and intuitive way and finally, the presentation of forecasts about the future operating status of the *Haulick & Roos* press.

4.2 Overview of the Proposed Solution

This project proposes a solution that solves the 3 problems described in Section 4.1. However, the main focus of this project is the development of a new approach for predictive maintenance systems, this new approach is based on the segmentation of time series. For the case of equipment that presents several changes of operation over time, for example, a mechanical press has periods in which it is operating normally, other periods in which it is stopped, other periods in which it is in the testing phase, among others. Each period has different operating patterns and, therefore, trying to extract patterns or detect anomalies using the data as a whole is a highly complex problem. The proposed approach focuses on the segmentation of time series in order to group the data in different states and only then proceed to an analysis of the data belonging to each state independently. In this way, it is possible to detect anomalies using much less complex algorithms and obtaining better results.

The general architecture of the proposed solution is shown in Figure 4.1.

The developed system presupposes the existence of several equipment properly equipped with sensors and an *Edge Device*. The *Edge Device* is responsible for extracting the measurements from the sensors and forwarding them in real-time to the developed system hosted on the server.

In Figure 4.1, it is observed that the first component of the server is *Data Reception*, this component has the function of receiving data from *Edge Devices*, processing the information received and sending it later to the next component, *Time Series Segmentation*.

In the component *Time Series Segmentation*, initially, the data received is separated based on the equipment to which they belong. Then, the data are processed in order to divide the time series into two segments, *Working* and *Stopped*. This segmentation occurs every t_{agg} seconds, with t_{agg} being the aggregation time performed, as will be

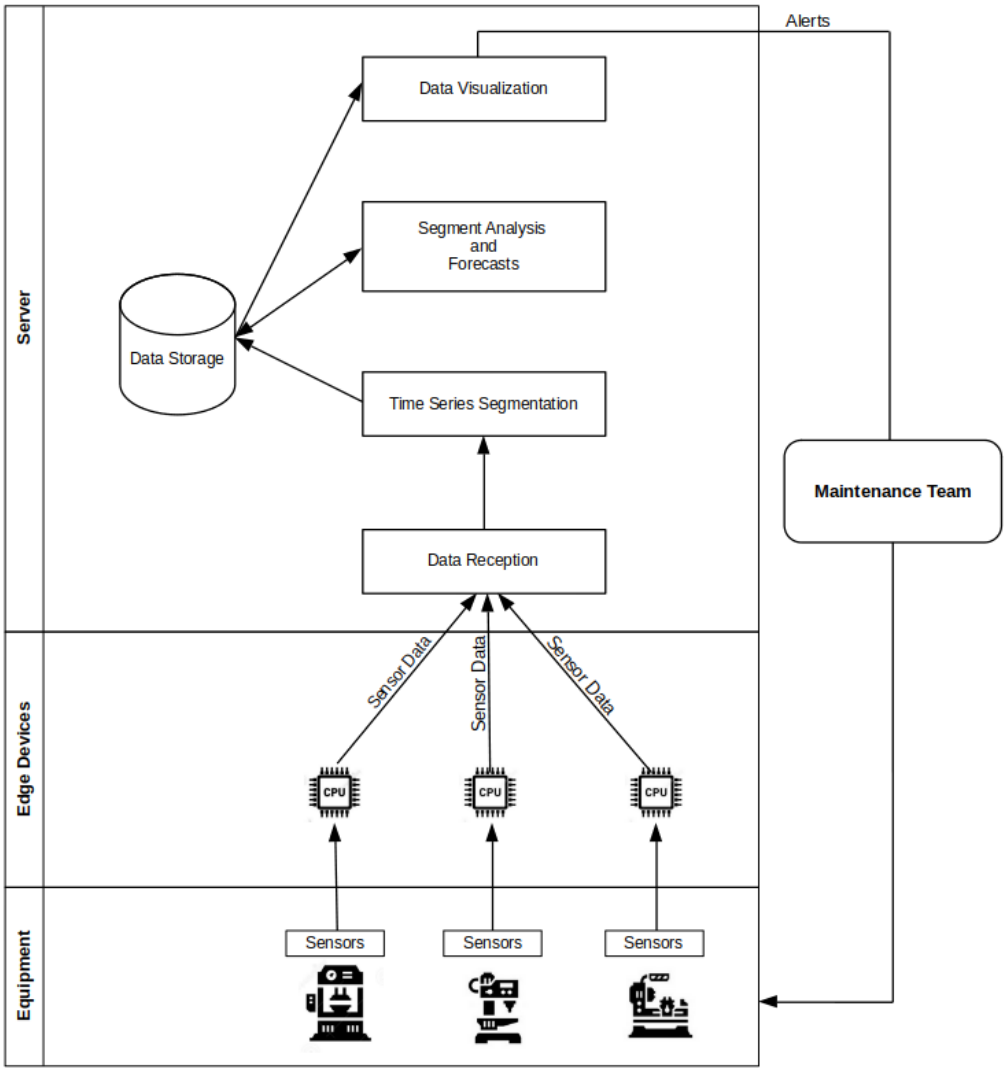


Figure 4.1: Architecture of the Proposed Solution

discussed in Chapter 7. At the end of t_{agg} seconds, all segmented time series are sent to a table in the database.

The component following *Time Series Segmentation* is *Segment Analysis and Forecasts*. This component runs every t_{for} minutes and performs the analysis of the segments marked as *Working* of each equipment. The objective is to detect anomalies in the last t_{for} minutes in order to anticipate a malfunction in a certain equipment. After t_{for} minutes, the information that characterizes the condition of the equipment in the last t_{for} minutes is sent to the database, being categorized as *Safe*, if less than $n_{ano}+1$ anomalies were detected, or *Warning*, if more than n_{ano} anomalies in the last t_{for} minutes were detected. n_{ano} is a value that must be defined a priori. In the event of anomalies, these are also recorded in the database.

A very important point in the *Segment Analysis and Forecasts* is the fact that the algorithms only search for anomalies in the *Working* segments. If anomalies were search

for in the entire time series, *Working* and *Stopped* segments, there would be many anomalies detected by the algorithm that do not correspond to reality, as many of these anomalies would correspond to changes from one segment to another.

The final component of the developed system is *Data Visualization*. This component corresponds to a dashboard that presents various information, such as:

- Number of equipment that are using the developed system;
- Number of equipment that are working and number of equipment that are stopped at the time of viewing;
- Time series of all sensors of each equipment;
- Characterization of the condition of each equipment in the last t_{for} minutes, using two possibilities, *Safe* or *Warning*;
- History of the characterization of the condition of each equipment, with increments of t_{for} minutes.

The developed visualization platform can be accessed anywhere in the company and must always be on a device near each equipment. Whenever a *Warning* alert is issued, operators who are close to the equipment are obliged to turn off the equipment and carry out a general overhaul of the equipment. If no signs of any damage are detected, they must turn on the equipment and proceed with production. If signs of a possible failure are detected, they must contact the maintenance team in order to act in advance, avoiding future problems.

Although Figure 4.1 represents three equipment, the developed system has the ability to process data from an indefinite number of equipment without having to change anything, the system was dynamically designed to adapt to different equipment and different sensors in each equipment.

4.3 Implementation

Figure 4.2 presents the architecture of the implemented solution, representing all the software and hardware used. The system was developed based on data from two different equipment, *bosch_prensa01* and *bosch_prensa02*. *bosch_prensa01* represents the *Haulick & Roos* press and it is the equipment on which the predictive maintenance system was performed. *bosch_prensa02*, although it is represented as a equipment, in reality, it is not a equipment, it was only used to evaluate the time series segmentation of data coming from several equipment at the same time and to test an alternative way of data ingestion, as will be explained in Chapter 5.

The data from the sensors belonging to each equipment are sent by the *Edge Devices* to a REST (*Representational State Transfer*) API (*Application Programming Interface*) developed using *Flask*. All functions performed by the REST API will be covered in detail in Chapter 6.

For the implementation of the *Time Series Implementation* component, a *Message Oriented Middleware* (MOM) was used. A MOM allows the developer to focus on the development of the services themselves, instead of focusing on the implementation of

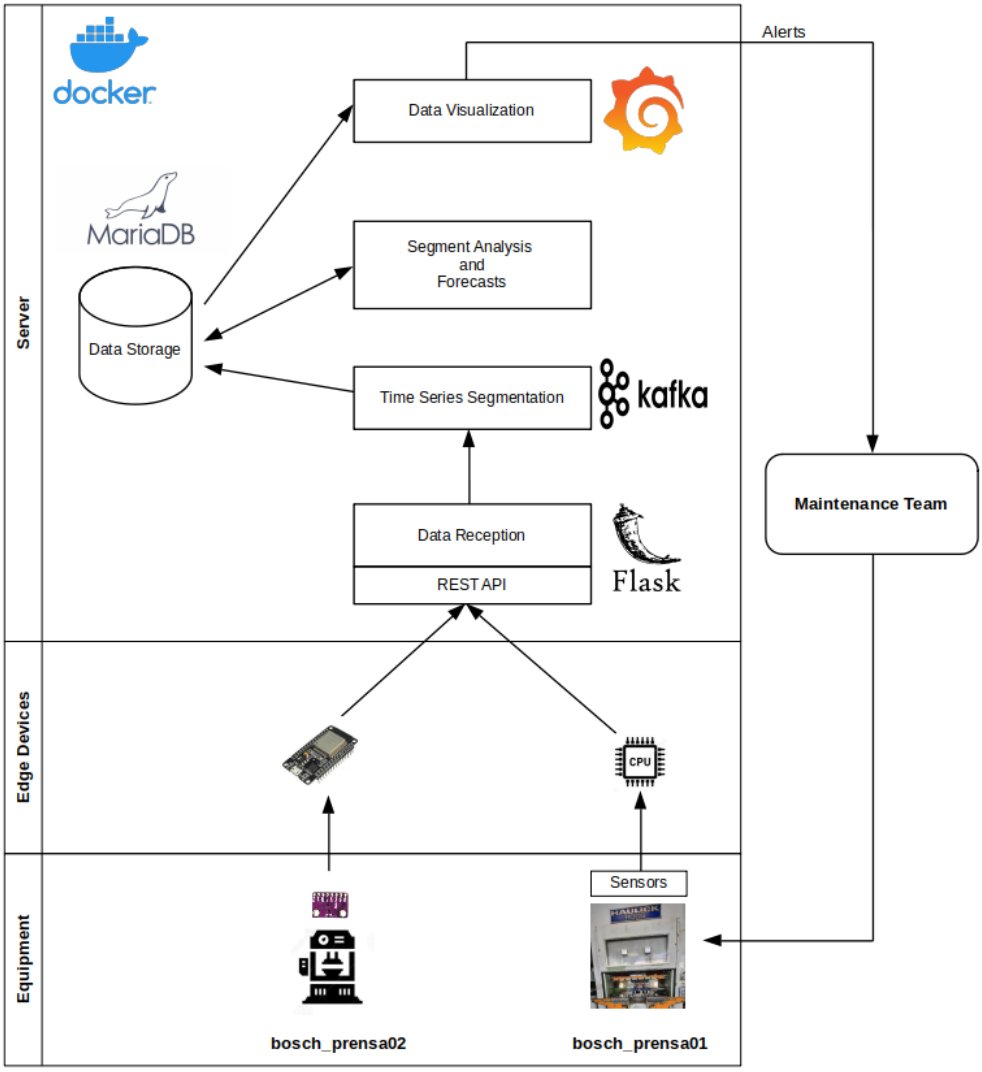


Figure 4.2: Architecture of the Implemented Solution

communication protocols between the various services. In this case, the MOM used was *Kafka*. Appendix B contains a brief explanation of some types of MOM, as well as the reasons why *Kafka* was used in the implemented solution. Chapter 7 covers all the steps implemented from the receipt of data related to sensors to the storage of the time series segmented in the database.

The *Segment Analysis and Forecasts* component includes the analysis of the time series segments in order to detect anomalies. For this purpose, a data processing pipeline was developed, covered in Chapter 8. It is important to note that only the time series segments from the equipment *bosch_prensa01* were analyzed, since *bosch_prensa02*, as explained above, does not correspond to a true equipment in which failures can be anticipated.

The database management system chosen to store all the data produced by the developed system was *MariaDB*. *MariaDB* is an open-source, multi-threaded, relation database management system. It is a popular fork of *MySQL* created by *MySQL*'s

original developers. Security, speed and efficiency were the three factors that contributed to the use of *MariaDB* instead of other database management systems.

Grafana was the chosen visualization platform. *Grafana* is an open-source visualization and analytic software. *Grafana* communicates directly with *MariaDB* through queries, immediately updating the implemented dashboards. All the information present in the developed dashboard will be presented over the next chapters.

To simplify the development and deployment of the system, *Docker* was used as a virtualization platform. All of the components explained previously were containerized using *Docker* containers. In order to enable the exchange of information between containers, a network was created with the name *sbroETL*, and this network was assigned to all containers. More information about *Docker* can be found in Appendix A.

The next four chapters cover the four stages of the developed system, *Data Ingestion*, *Data Reception*, *Time Series Segmentation* and *Segment Analysis and Forecasts*. Throughout each of these chapters, data pertaining to the equipment present in Figure 4.2 will be presented.

Chapter 5

Data Ingestion

The first step necessary to take advantage of the developed system is to send sensor data to the *REST API* hosted on the server. To do this, it is necessary to install sensors in the various equipment, extract their measurements using *Edge Devices* and send these measurements, in a normalized way, to the REST API.

As can be seen in the architecture of the implemented solution, Figure 4.2, two alternative ways of ingesting data on the server were used, however, only the way of ingesting data corresponding to *bosch_prensa02* can be used in production, the data ingestion of *bosch_prensa01* was an alternative way to be able to use the data from the *Haulick & Roos* press, without having to change the hardware and software currently installed on the press.

Figure 5.1 shows in more detail the data ingestion pipeline corresponding to each of the equipment used.

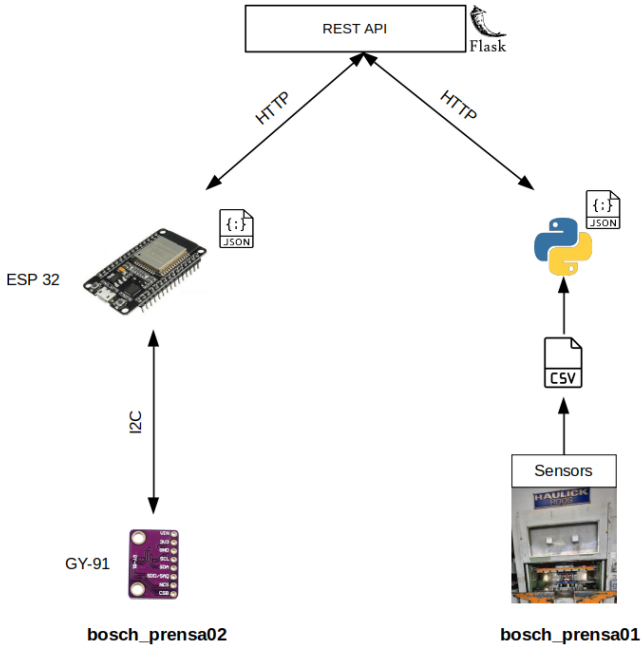


Figure 5.1: Data ingestion pipeline

Sections 5.1 and 5.2 discusses the *bosch_prensa02* and *bosch_prensa01* data ingestion pipeline, respectively.

5.1 *bosch_prensa02*

This section deals with a data ingestion pipeline that can be used in the production phase. As shown in Figure 5.1, in this case, there is no real equipment, only a data ingestion pipeline has been developed ready to be installed on any equipment.

The developed system requires the installation of two types of devices, a sensor, capable of making measurements on some property, and an *Edge Device*, capable of extracting the values measured by the sensors and sending this information to the server. The remainder of this section deals with a possible choice of these devices taking into account an eventual installation on a mechanical press.

5.1.1 Data Source

The first task is to study the functioning of the equipment and understand what are the physical properties that best characterize the functioning of the equipment. In the case of a stamping press, as discussed in Chapter 3, vibration analysis is the most suitable for predictive maintenance problems.

In order to analyze the equipment's vibrations, the GY-91 module, Figure 5.2, was chosen. This module contains two sensors, MPU-9250, which is an accelerometer and BMP 280, which is an atmospheric pressure gauge. Only the MPU-9250 was used, since atmospheric pressure, in this situation, was not relevant data for describing the equipment's operation. The sensor MPU-9250 has the ability to measure acceleration in 3 axis, which offers a great advantage, since it can provide a more complete information of the environment where it is inserted.

Figure 5.2: Module GY-91

In addition to choosing the sensor, it is also necessary to define the location of the equipment where the sensor will be installed. In a mechanical press, the main target on which maintenance systems are concerned are the crankshaft bearings. For example, excessive vibrations caused by damage on crankshaft bearings can damage the bearing's housing and other components causing serious damage to the press. Based on this, the location where the sensor would be installed would be next to the crankshaft bearing.

5.1.2 Edge Device

In addition to the implementation of sensors, it is necessary to implement an *Edge Device*. This device has the function to extract the measurements made by the sensors

and send this data in a normalized way to the server.

There are many devices capable of performing these operations, such as ESP32, ESP8266, *Raspberry Pi*, PLC, among others. In this case, ESP32, Figure 5.3, was chosen.

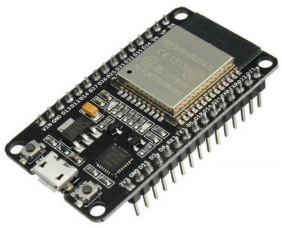


Figure 5.3: ESP32

ESP32 is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and bluetooth. The two functions performed by ESP32 are discussed below.

Data Acquisition

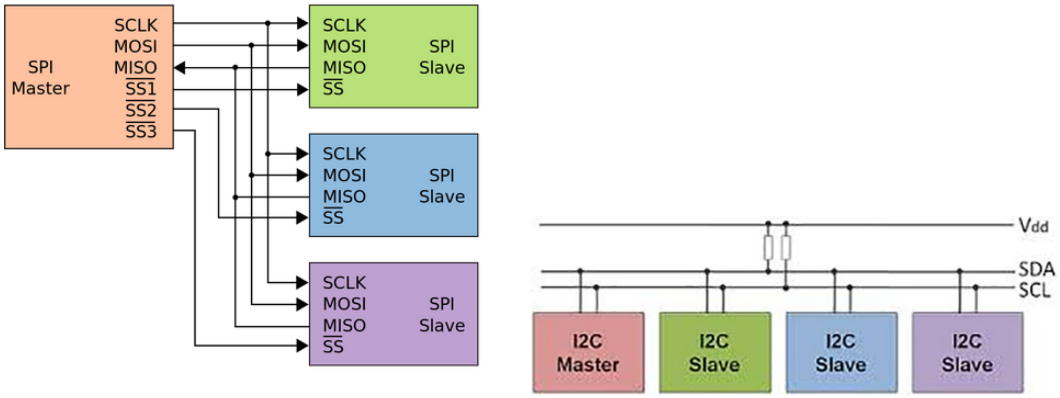
The first function performed by the *Edge Device* is to acquire the measurements made by the sensor, in this case, it is to acquire the three accelerations measured by the MPU-9250.

The MPU-9250 sensor captures the accelerations to which it is subjected and stores the value in one of its memory positions. This sensor has two communication protocols, *Serial Peripheral Interface* (SPI) and *Inter-Integrated Circuit* (I2C). We opted for the I2C protocol for two reasons, the first is its simplicity, there is only need 2 wires, *serial data* (SDA) and *serial clock* (SCK), for data transmission and synchronization, respectively, while the SPI protocol requires three wires, *Serial Data Out* (SDO/MOSI), *Serial Data In* (SDI/MISO) and *Serial Clock* (SCK). The second reason is the ease with which it is possible to add more slaves to the transmission, while in SPI it is necessary to add a *Slave Select* (SS) pin for each existing slave, in the I2C, to add a new slave it is possible to use the same bus but with the 7-bits that identify the new slave. Figure 5.4 shows a scheme that illustrates some of the differences between the two protocols.

ESP32 was programmed to read, based on the I2C protocol, the value of the accelerations in their memory positions, for this purpose the MPU-9250 datasheet was used in order to consult the address of the sensor, as well as the memory locations where the measurements were stored. The electrical circuit between the ESP32 and the GY-91 is shown in Figure 5.5. In Appendix D, more information about the I2C protocol is found, as well as the *MPU-9250* sensor datasheet.

Data Transmission

After ESP32 reads the measurements, it is necessary to serialize the information in a JSON file. As such, the first task is to create a JSON file that contains all fields listed in Appendix C, placing all fields empty except three:



(a) SPI [Suiu 2020] (b) I2C [Tulan 2015]

Figure 5.4: Schematic differences between SPI and I2C

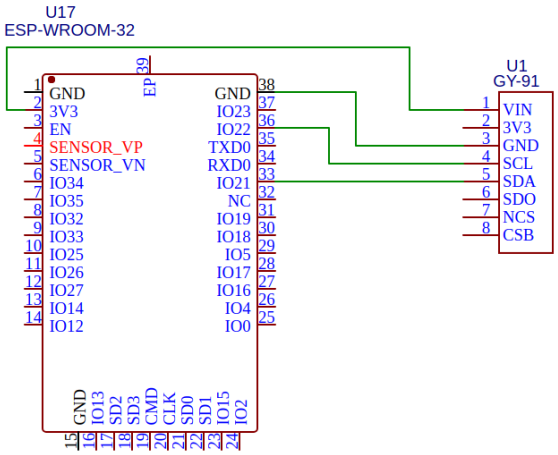


Figure 5.5: Electric circuit between ESP32 and GY-91

- ***location***;

  String that represents the location of the equipment where the *Edge Device* is installed.

- ***dim00***;

  String that represents the identification of the sensor from which the measurement was extracted.

- ***val00***.

  Float that represents the extracted value of the sensor identified with the field *dim00*.

At the end of JSON creation, it is necessary to establish an HTTP connection to the REST API on the server and send the JSON file using the POST method. If the

request is successful, the REST API sends the code 201, claiming that a new resource was created, otherwise, it sends the code 400, claiming a bad request, meaning that it was unable to understand the request made.

It is important to note that each measurement is sent independently, so in this case, as there are three measurements, it is necessary to create three JSON files and send each one in an independent POST.

Table 5.1 presents the name given to each measurement, the physical meaning of each measurement, the unit of the measurement and, finally, the frequency of the JSON files sent to the REST API for each measurement.

Table 5.1: Measurement information in *bosch_prensa02*

Sensor Name	**Meaning**	**Unit**	**Frequency**
ShaftVibrationX	Vibration measured on the X axis	g	142 Hz
ShaftVibrationY	Vibration measured on the Y axis	g	142 Hz
ShaftVibrationZ	Vibration measured on the Z axis	g	142 Hz

This pipeline developed for data ingestion only presents a possible solution, it is not mandatory to have these devices on all equipment, for example, on other equipment there may be other sensors and a *Raspberry Pi* instead of an ESP32. What is mandatory is for the *Edge Device* to send a JSON file normalized on an HTTP POST request.

5.2 *bosch_prensa01*

This equipment refers to the data from the sensors installed on the *Haulick & Roos* press, covered in Chapter 2. As mentioned in Section 4.1, one of the problems that this project aims to solve is the high number of unplanned stops in this press, for this, the first step is to send the data from the sensors installed in the press to the developed system.

The developed system, as previously mentioned, works in real-time, and all the sensor data related to the press were stored in the *Nexeed MES*, so it was necessary to implement a solution capable of simulating the streaming of the storage data in real-time.

The implemented solution consists of storing the data of all sensors in a CSV (*comma-separated values*) file, in which each row of the file corresponds to a measurement of a sensor. Using this file, a service was developed that reads each row in the file and sends a JSON file containing the information of the respective row to the REST API through the POST method. In order to simulate the streaming of data in real-time, the service developed respects the time interval between consecutive rows, so the REST API receives JSON files with the same frequency if the data had come directly from an *Edge Device*.

Table 5.2 contains all information regarding the sensors. Comparing the sensors present in Table 5.2 with the sensors identified in Chapter 2, it is possible to verify that not all sensors are found in the presented table. In fact, more sensors are present in the CSV file, however through an initial filtering, only these sensors were chosen to develop the anomaly detection system. Factors such as the amount of missing data and measurements with sudden changes in magnitude in a short period of time contributed to the removal of the remaining sensors.

As discussed in Chapter 2, the data acquisition system currently implemented in

Table 5.2: Measurement information in *bosch_prensa01*

Sensor Name	**Meaning**	**Unit**	**Frequency**
OilTemperature	Oil temperature in the hydraulic tank	°C	1 Hz
OilLevel	Oil level in the hydraulic tank	%	1 Hz
EngineActualSpeed	Current motor speed	rpm (revolutions per minute)	1 Hz
ShaftVibration	Average vibration of crankshaft bearings and connecting rod bearings	mg	1 Hz
PressureLeft	Hydraulic pressure on the left side of the hydraulic tank	bar	1 Hz
PressureRight	Hydraulic pressure on the right side of the hydraulic tank	bar	1 Hz

the press sends an XML file to *Nexeed MES* containing all the sensors every 1 second. Therefore, the REST API receives a JSON file for each sensor with a frequency of 1 Hz.

It is important to note that this form of data ingestion was not developed to be used in the production phase, it was just a solution to use the data from the *Haulick & Roos* press in the development phase. When the developed system goes into production, it is necessary to configure the PLC CX2020 to make HTTP requests to the REST API or to implement in parallel another *Edge Device* with the ability to send the data to the REST API using HTTP requests.

Chapter 6

Data Reception

This chapter deals with the first component of the architecture represented in Figure 4.2, *Data Reception*.

The purpose of this component is to receive the data from the *Edge Devices* and forward this information to *Time Series Segmentation*. As mentioned in Chapter 4, a REST API was developed to achieve these objectives.

As stated in Chapter 4, the component *Time Series Segmentation* performs data processing using *Kafka* as a message transfer protocol, so the initial solution was for the *Edge Devices* to be *Kafka* clients and send the values of the respective sensors directly to the *Time Series Segmentation* via a topic in *Kafka*. However, some *Edge Devices* such as ESP32 and ESP8266 lack the capacity of being *Kafka* clients. Hence, the need arose for the implementation of a REST API that receives HTTP requests from *Edges Devices* and generates a normalized *Kafka* message ready to enter the *Time Series Segmentation*. In this way, it is possible to use all the advantages of *Kafka*, even using devices with low computational power, such as *Edge Devices*.

Section 6.1 presents an overview of a REST API and Section 6.2 presents the implemented REST API.

6.1 Overview of REST API

REST API, also called RESTful API, is an application programming interface that follows the restrictions of the REST architecture. API is a set of definitions and protocols for creating and integrating programs. For an API to be considered of the RESTful type, there are some conditions to be fulfilled [Lange 2016]:

- **Client Server**;

  The first constraint is that the system must be made up of clients, servers and resources, with requests managed through HTTP requests.

- **Stateless**;

  The communication between client/server is stateless, that is, all information about the client's session is kept only on the client.

- **Cache**;

  Server responses must be marked as cacheable or noncacheable. The cache can reduce the number of client/server interactions, increasing the performance level of the system.

- **Uniform interface**;

- **Layered System**.

  The client is only aware of the layer that is immediately following, not being aware of the layers behind the server.

6.2 Implemented REST API

The REST API was developed using *Python* and *Flask*. Figure 6.1 shows schematically all operations performed by the REST API.

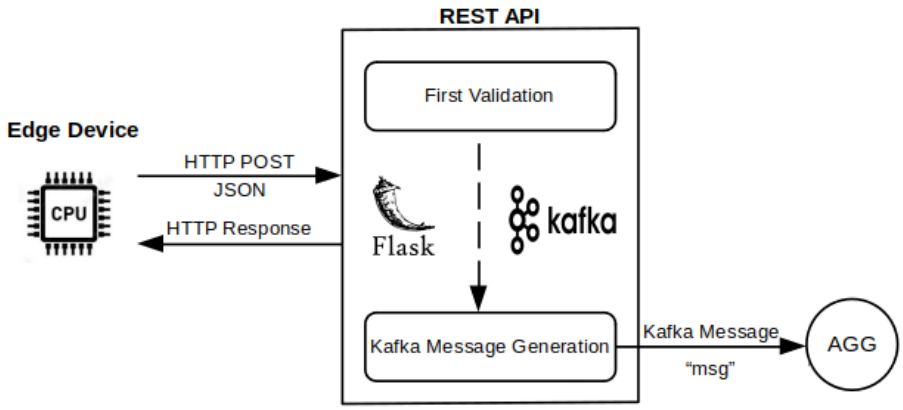


Figure 6.1: Operations performed by the REST API

The REST API service receives a POST request from the *Edge Devices*, extracts the JSON file and performs the first validation. In this validation, the REST API verifies that the file sent is a JSON, and presents all the necessary keys, Appendix C. If any of these conditions are not met, the REST API sends an HTTP Response with code 400, claiming that the client's request is incorrect. Otherwise, the JSON is considered valid, and the REST API proceeds to create a *Kafka* message with the fields present in the JSON file, sending the message to the topic AGG. If the message is sent successfully, an HTTP Response with code 201 is sent, claiming that the request was successful and that it led to the creation of a resource.

As discussed in Section 4.1, the entity responsible for assigning the timestamp to the *Kafka* messages is the REST API and not each *Edge Device*, thus ensuring that there is only one time reference.

As all components of the system were containerized using *Docker*, when the developed system is in production and there are many *Edge Devices* sending HTTP requests to the REST API, it is possible to scale the container containing the REST API, ensuring that no HTTP request is lost, and that the efficiency of the system is not compromised by the increase of *Edge Devices*.

Chapter 7

Time Series Segmentation

This chapter addresses the most important component of the developed system, *Time Series Segmentation*. This is the element that differentiates the proposed approach for predictive maintenance systems from the others. The general objective of segmentation is to reduce the complexity of the algorithms to be applied to predict failures by dividing the equipment data according to the equipment operating phase.

The implemented segmentation initially groups the data depending on the equipment to which they belong. Then, after some data transformation, the data of each equipment is inserted into two segments, *Working* segment or *Stopped* segment. The *Working* segment contains the equipment data when it is in the operating period, the *Stopped* segment contains data when the equipment is stopped or when it is undergoing initiation tests, for example, when testing whether the maintenance performed has resolved the problem or whether the problem still persists. In case the equipment is in fact stopped, this stop can be either planned or unplanned, and it is this second type of stop that is to be avoided.

It is important to note that the two types of segments created were designed taking into account the data from the equipment *bosch_prensa01* (*Haulick & Roos*). Through a visualization of the data of the *Haulick & Roos*, it is possible to verify that there are periods in which the data frequency is not *1 Hz* and these periods do not correspond to a normal operation of the equipment, but to phases in which the equipment is being tested. In this way, with the segmentation of the time series, it is possible to identify these data and not consider them when applying the algorithms at the time of data analysis, since they would cause noise and, therefore, would increase the complexity of the analysis to be performed.

Although in this case the time series have been divided into *Working* or *Stopped* segments, the implementation of segmentation offers the possibility of using other states, such as heating state and steady state. To do so, it is only necessary to parameterize these new states and the rest of the developed system adapts to the existing states.

Section 7.1 covers the entire *Kafka* pipeline implemented, explaining in detail all the transfer of messages between topics. Section 7.2 presents an example of the segmentation carried out over a period of time, relative to the data of *bosch_prensa01*.

7.1 Kafka Pipeline

As mentioned before, *Kafka* was used to perform the time series segmentation. Figure 7.1 shows the pipeline developed using *Kafka*, from the reception of the *Kafka* message from the REST API to the storage in the database of the segmented time series.

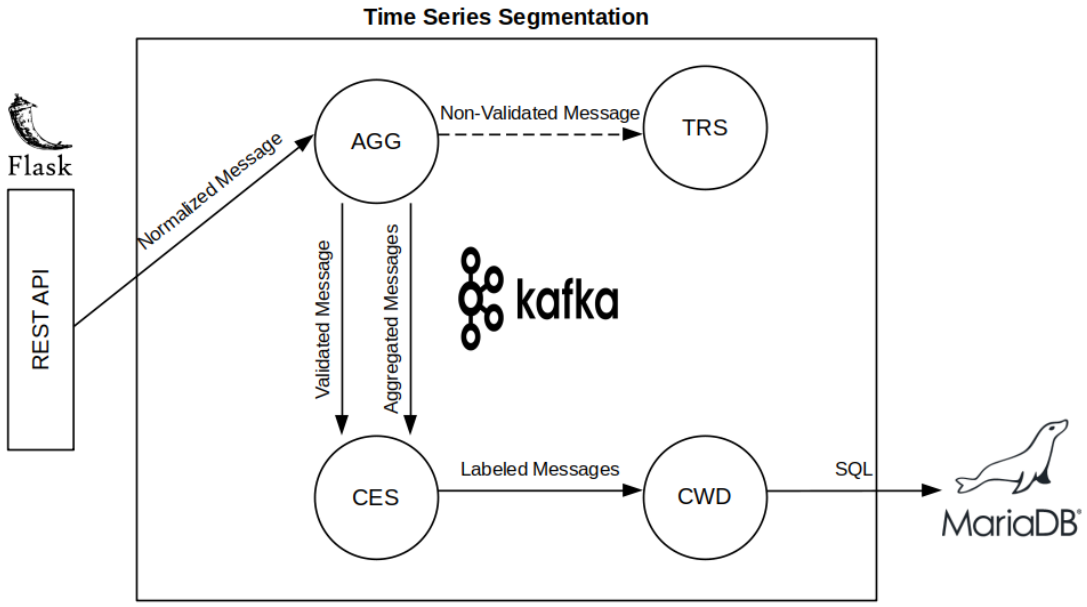


Figure 7.1: Kafka pipeline

The rest of this section discuss in detail all the transformations made throughout the pipeline.

7.1.1 Topic AGG

The topic AGG is the first topic in the *Kafka* pipeline. AGG stands for aggregation since it is the main task performed by the services that manage the messages that reach this topic. Figure 7.2 presents a diagram that illustrates all the tasks performed by these services. In addition to aggregation, the services also perform the *Second Validation* on incoming messages. Below, each of these two tasks is described.

Second Validation

As soon as a message arrives in the topic AGG, the second validation is performed. This validation focuses on the sensor name described in the *Kafka* message.

The parameter *dim00*, defined by the *Edge Device*, is a parameter that is important to standardize, otherwise, it is possible to have two identical sensors in two different equipment with a different name, or have a sensor name that does not clearly identify the property that is being measured. In order to promote the standardization of the name of the sensors, regular expressions were used that define a priori the way of writing the name of the sensor.

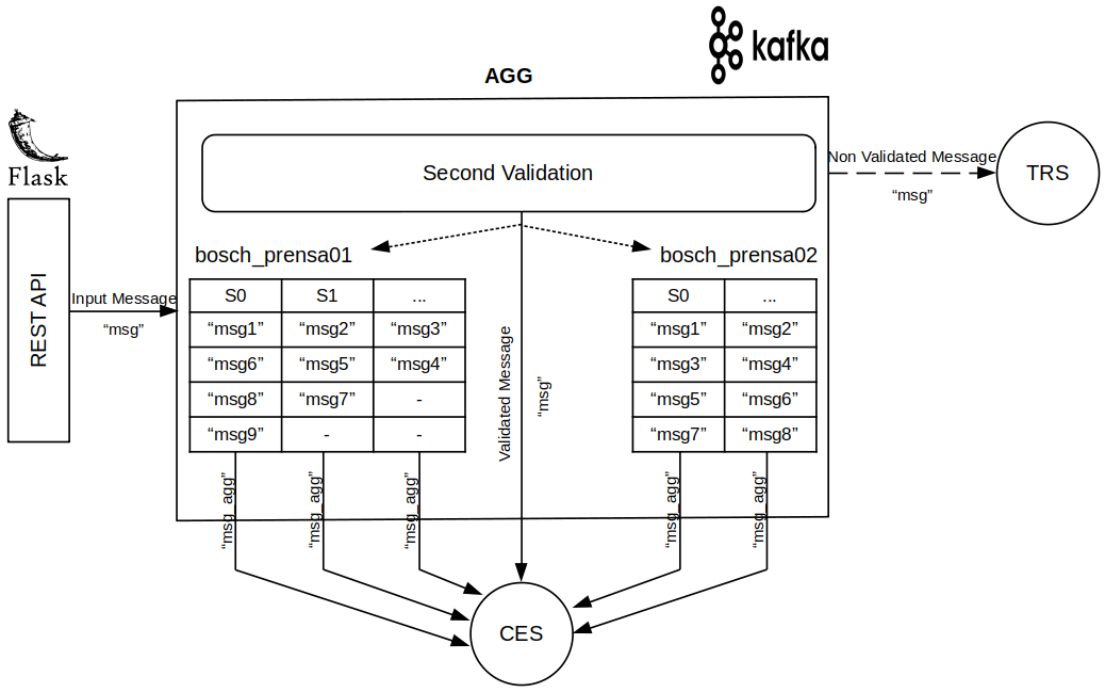


Figure 7.2: Operations performed on messages that reach the topic AGG

A *Regular Expression* is a string that allows the creation of patterns to be detected in another string. As an example, below are one of the regular expressions implemented in this validation.

- **ˆOil[\w-]+$**
  - ˆOil
    The string must start with "Oil";
  - [\w-]+
    After Oil, the character can be any lowercase, uppercase character, any digit, "_" or a "-". The "+" symbol allows the previous character to appear once or more;
  - $
    This character forces the string to end there.

This expression accepts sensor names like "OilPressure", "OilTemperature" and "OilLevel", but rejects names like "PressureOil" and "Oil:Temperature". Using this methodology, it is possible to reduce the probability of having two sensors measuring the same property with different names, and at the same time, it offers the freedom to install more sensors on any equipment without having to change the developed system, it is only required to follow the patterns described by the implemented regular expressions.

Each message that reaches the validation goes through all the regular expressions implemented, and if it does not present the patterns defined by any regular expression, it is considered a non-validated message and is sent directly to the topic TRS (Trash).

Otherwise, the message is considered valid and is copied immediately. The copy of the message is sent directly to the topic CES, called *msg* in Figure 7.2, and the original message goes to the second operation carried out in this phase, the aggregation.

The topic TRS is a topic where all non-validated messages are stored to check the problems with them.

Aggregation

As mentioned in Chapter 3, data aggregation consists of grouping data over a period of time, extracting important properties. This was the method chosen to enable the detection of the operating status of each equipment, that is, from the aggregation of several *Kafka* messages belonging to an equipment, it is possible to check if a equipment is working or is stopped. Based on the knowledge of the operating status of each equipment, it is possible to segment the time series, since it is only necessary to insert the data that arrives into the segment that corresponds to the active state.

The time according to which the data aggregations are performed is a pre-defined value, throughout the document this value is defined as t_{agg}, and this value is measured in seconds.

As explained in *Second Validation*, what arrives at this stage is a *Kafka* message from a certain sensor installed in a given equipment. This message is processed, extracting both the equipment and the sensor to which this message refers. It is subsequently stored in a data structure together with messages belonging to the same sensor and the same equipment, as shown in Figure 7.2.

Every t_{agg} seconds an aggregation is elaborated for each existing sensor, and several aggregation properties are calculated, such as number of messages, mean, minimum, maximum, sum, standard deviation, among others. For each aggregation performed, a *Kafka* message is created that contains all the properties of the aggregation performed, *msg_agg* in Figure 7.2. Then, these aggregation messages are sent to the topic CES where they will be processed in order to extract the operating status of each equipment.

It is important to note that, t_{agg} does not have to be global, that is, the aggregations of each equipment can have different times. For example, equipment that behaves more dynamically may have aggregations with a shorter time interval in order to better characterize all variations of the behavior.

7.1.2 Topic CES

CES stands for *Calculate Equipment Status*. This topic receives two types of *Kafka* messages from the topic AGG, receives messages that only contain the measurement of a sensor, *msg*, and receives aggregation messages containing a set of aggregation properties, *msg_agg*.

As in the topic AGG, there are two major tasks performed by the services that manage the messages that reach this topic, *Update States* and *Message Labeling*. Figure 7.3 shows a schematic overview of the processing carried out by these services.

Below is an explanation of the two major tasks performed.

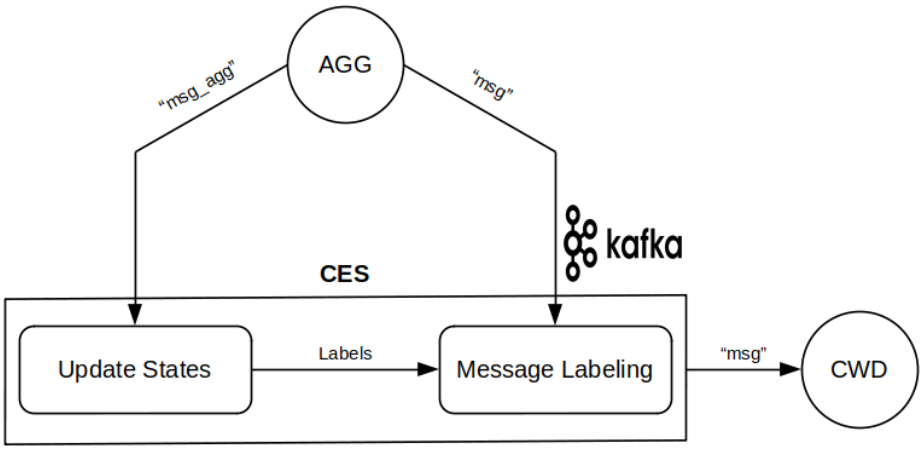


Figure 7.3: Operations performed on messages that reach the topic *CES*

Update States

In this task, based on the information in the aggregation messages, the status of the equipment is updated, and can only take the value of *Working* or *Stopped*.

This task depends on the equipment to be approached. In the case of the two equipment covered, *bosch_prensa01* and *bosch_prensa02*, when the equipment is not in operation there is no sending of data and therefore, by analyzing the number of messages present in the aggregation, it is possible to verify whether the equipment is stopped or not. However, as previously mentioned, the equipment *bosch_prensa01* has test periods and the objective is to place the data referring to this period also in the *Stopped* segment. Through an analysis of the equipment test periods, it was found that the frequency of data sending in this situation is always much lower than *1 Hz*, so a threshold, n_{mes}, was defined, which concerns the minimum number of messages in the aggregation necessary to activate the *Working* status.

The value of n_{mes} is directly related to t_{agg}, since the higher t_{agg}, the more messages are found in the aggregation. The objective is to play with these values so that if the sending frequency is close to *1 Hz*, the *Working* status is activated, if it is much lower, the *Stopped* status is activated. A pair of admissible values for the case of *bosch_prensa01* would be $t_{agg} = 10$, and $n_{mes} = 5$, so it is guaranteed that the *Working* status is only activated if in the last 10 seconds at least 5 messages from a sensor have been sent.

Besides being necessary to define n_{mes}, it is also necessary to define for each equipment if all sensors are taken into account to define the status or if there is a master sensor. In this case, it was considered that to activate the *Working* status, all aggregations referring to all sensors must have more than n_{mes} messages.

Message Labeling

The purpose of this task is to associate the status of the equipment defined in the previous stage with each message. For this, only the message *msg* is received, and the active status is inserted in one of the message fields.

At the end of the message labeling process, the message *msg* is sent to the last topic

in the *Kafka* pipeline, the topic CWD, while the message *msg_agg* is deleted since it has already fulfilled its function.

7.1.3 Topic CWD

CWD stands for *Communication with Database*. This topic is the last in the *Kafka* pipeline. It receives a *Kafka* message from the topic CES, and the the services that manage the messages that reach this topic are responsible for processing that message and sending all the necessary information to the database used, *MariaDB*.

Figure 7.4 presents a schematic of the operations performed by these services.

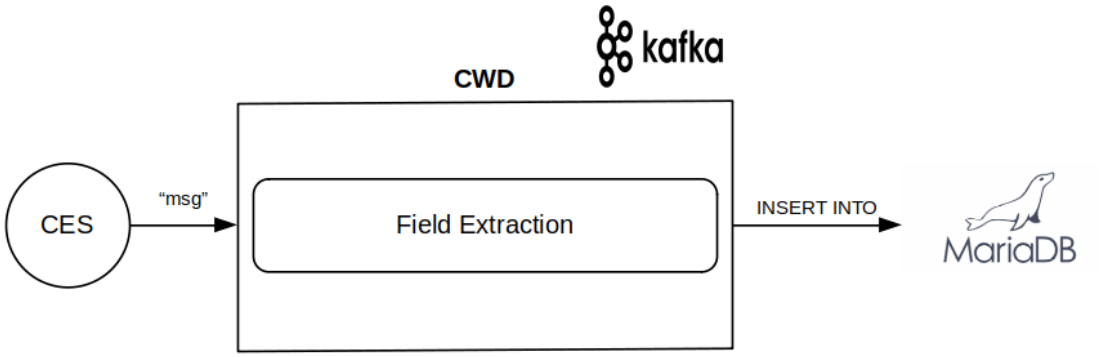


Figure 7.4: Operations performed on messages that reach the topic CWD

When a message arrives at this topic, the first task to perform is the extraction of the relevant fields, which are:

- Date and Time;
- Message timestamp;
- Equipment Name;
- Equipment Status;
- Sensor Name;
- Sensor Measurement.

Then, these values are sent to a table belonging to the database.

Since each message that arrives is associated with the status of its respective equipment, when visualizing the time series of a given sensor it is possible to identify several segments over time.

7.2 Visualization

In this section, the objective is to present the result of all the processing carried out in the previous section. Table 7.1 shows the values used for the parameters discussed above.

Table 7.1: Time series segmentation parameters used

Parameter	Value
t_{agg}	10
n_{mes}	5

Figure 7.5 presents an image of the part of the developed dashboard that presents the segmentation of the time series. As can be seen in the upper-left corner, these data belong to the equipment *bosch_prensa01*. In the lower-left corner, it is possible to notice that the time series shown belongs to the *OilLevel* sensor since it has a yellow color. All time series are segmented, however only this one is shown to simplify the visualization.

Figure 7.5: Time series segmentation example

In Figure 7.5, there are clearly two segments along the time series, the green segment corresponds to the *Working* segment, and the red segment corresponds to the *Stopped* segment, for example, from 18:34:14 until 19:05:29 the equipment was stopped, having then been running from 19:05:29 to 19:33:11. At around 18:49:00, it is possible to verify the effect of n_{mes}, those data are considered in the *Stopped* segment because they have a frequency lower than expected, it is possible to verify that they have a spacing between them greater than the others. Those data correspond to a period in which the equipment was being tested and therefore it makes no sense to analyze this data.

In addition to the representation of the segmented time series, some statistical data about the equipment is also presented in the developed dashboard, Figure 7.6. *Number*

of Equipment represents the number of equipment that are using the developed system, *Working* represents the number of equipment that is currently working and finally, *Stopped* represents the number of equipment in the stop period.

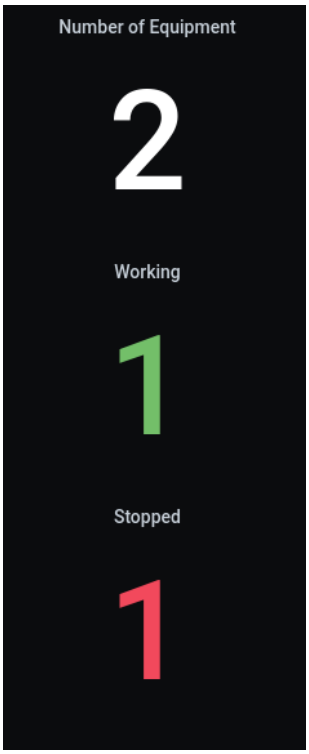


Figure 7.6: Statistical data on equipment

Chapter 8

Segment Analysis and Forecasts

This chapter deals with the analysis of the time series segmented, explaining all the necessary steps to transform the segments addressed in Chapter 7 in forecasts of equipment failures.

Contrary to what is normally used in predictive maintenance systems, Chapter 3, no supervised system was used, so there is no need to provide labeled data for the model to learn. In this case, the objective is to predict the failures through the detection of anomalies in the past data.

An anomaly or outlier according to [Hawkins 1980] is "an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism". [Barnett and Lewis 1994] indicate that "an outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs". In this case, an anomaly can have two causes, a malfunction of the equipment due to problems, or it can simply be a measurement error by some sensor.

Although the time series segmentation is a process that is running continuously, the segment analysis process takes place every t_{for} minutes, issuing alerts for *Grafana* separated by t_{for} minutes. The messages displayed by *Grafana* can be of two types, *Safe* or *Warning*. *Safe* is when less than $n_{ano} + 1$ anomalies were detected in the last t_{for} minutes, *Warning* is when more than n_{ano} anomalies were detected in the last t_{for} minutes. When a *Warning* is issued to *Grafana* it just means that there are anomalies in the operation of the equipment, the equipment may malfunction after 1 minute, 10 minutes, 20 minutes or not at all. However, the most cautious option is to stop production and check if there is a problem with the equipment.

The developed system allows implementing a different analysis of segments for each equipment, for example, in one equipment an anomaly detection algorithm may have better results and in another equipment it may have to be another. However, in the implemented system, only the analysis of the data from *bosch_prensa01* was performed, since, as the data relating to *bosch_prensa02* are not from any equipment in operation, failures cannot be predicted.

Section 8.1 presents the developed pipeline, explaining all the steps until the forecast is reached, and Section 8.2 shows a part of the dashboard where the output of this analysis is presented.

8.1 Pipeline

Figure 8.1 shows the pipeline implemented for the analysis of segments.

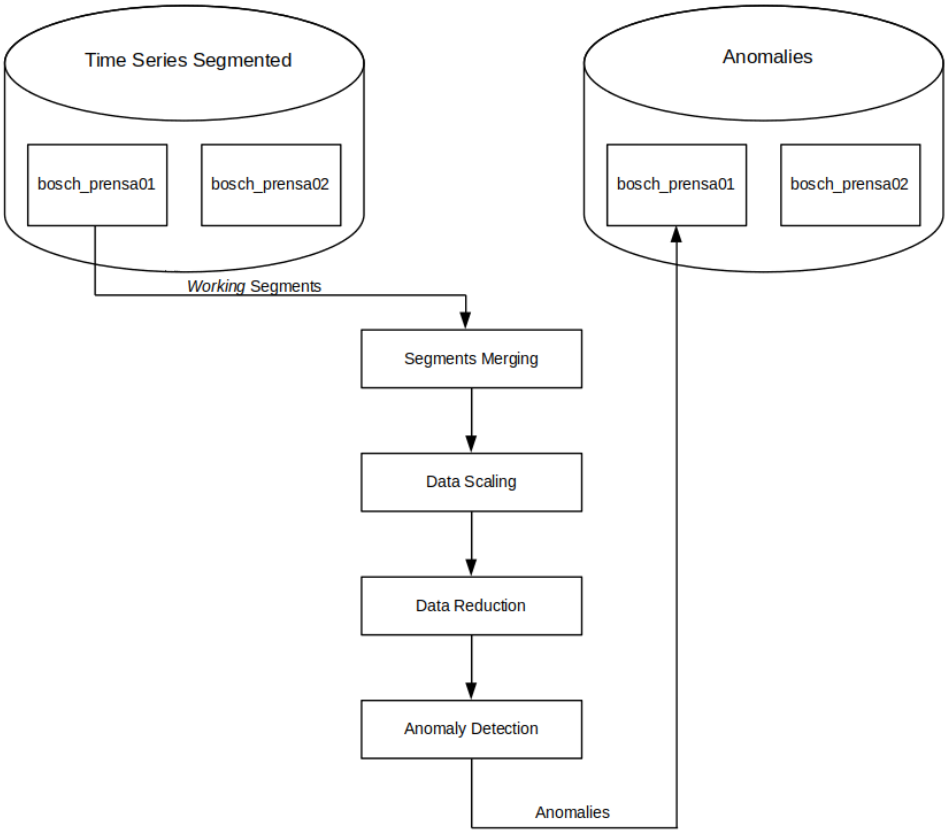


Figure 8.1: Segment analysis pipeline

The first step is the acquisition of the *Working* segments of the segmented time series stored in the database for the equipment to be analyzed, in this case, *bosch_prensa01*. The *Stopped* segment is not used because it represents periods of the press when it is not in operation or is being tested. Then the segments are merged, as only the *Working* segment was extracted, the time series left empty spaces between the data, hence the need to join them. After the merge, data scaling is performed to normalize all time series, since some series have higher values than others. Next, a data reduction is performed, more precisely a *Dimensionality Reduction* technique to reduce the number of variables that are being analyzed. Finally, an anomaly detection algorithm is applied in multidimensional data, in order to ascertain the need to stop or not the equipment.

In the rest of the section, all the components illustrated in Figure 8.1 will be discussed in detail.

8.1.1 Segments Merging

As shown in Figure 8.1, only data belonging to the *Working* segment are analyzed. Thus, all data analysis is performed with data belonging to the same segment.

One of the consequences of working only with time series segments, Figure 7.5, is the existence of time periods in which there is no data. The purpose of the *Segments Merging* is to precisely eliminate these time periods.

After receiving the segmented time series, the segments are merged in order to eliminate empty spaces. This merging creates a new time base, called *Virtual Time Base*. Figure 8.2 shows the data related to the sensor *OilLevel* after passing the *Segments Merging*. Once again, all the time series belonging to the sensors of the *bosch_prensa01* equipment are used, however, only the data related to this sensor are presented.

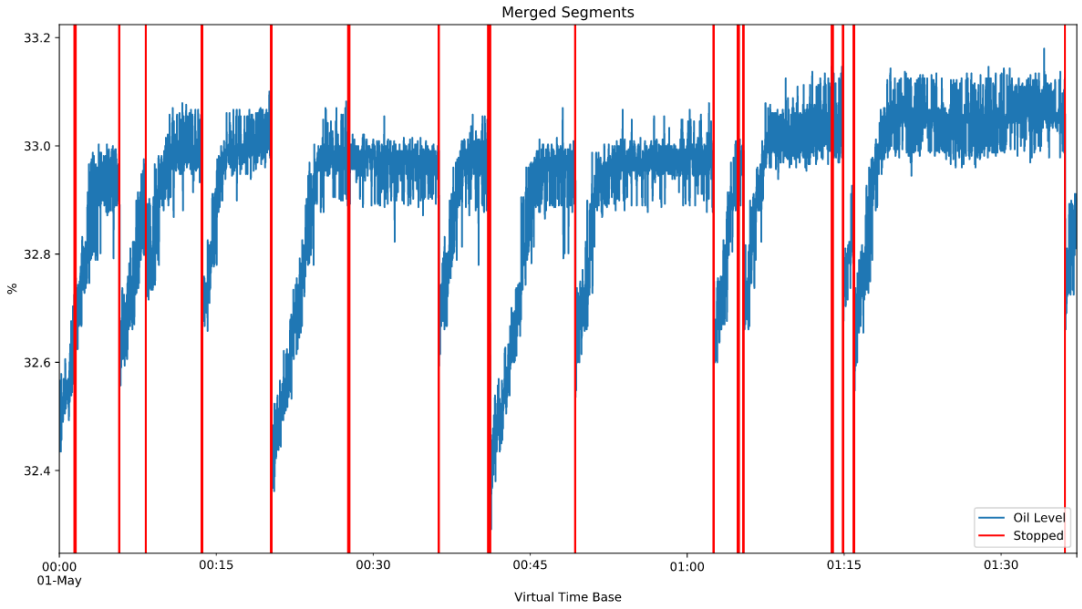


Figure 8.2: Merged segments in *OilLevel*

In Figure 8.2, the red vertical lines correspond to the time when two segments were joined, that is, when there was a stop, and their thickness is proportional to the duration of the stop. It is important to note that not all stops correspond to malfunctions, some simply correspond to meetings at the beginning of a shift, intervals, among others. On the x-axis, there is the *Virtual Time Base*, which always starts on 01/05/2021 at 00:00:00. The analysis of the segments takes place entirely based on the *Virtual Time Base*, and when anomalies are detected in the *Virtual Time Base*, it is necessary to convert the virtual time instants in which the anomalies were identified to the *Real Time Base*.

8.1.2 Data Scaling

After all time series are in the *Virtual Time Base* the data scaling process follows.

In the specific case of *bosch_prensa01*, there are 6 time series, each series having values with orders of magnitude quite different, for example, *ShaftVibration* presents values around 4 *mg*, and *PressureRight* presents values in the order of 430 *bars*. This difference in orders of magnitude may result in the algorithms giving more weight to some time series than to others.

The purpose of this component is to transform the time series in such a way that they all have the same range of values, thus eliminating the possibility that some time

series have more weight than others in data analysis. For this, we used the scaling to minimum and maximum method, applying the Equation 3.6. In this way, all values of the time series present values between 0 and 1.

In Figure 8.3, the *OilLevel* time series is shown after the data scaling has been applied. As can be see through a comparison between Figure 8.2 and Figure 8.3, only a scale of all values was carried out, all patterns remained unchanged.

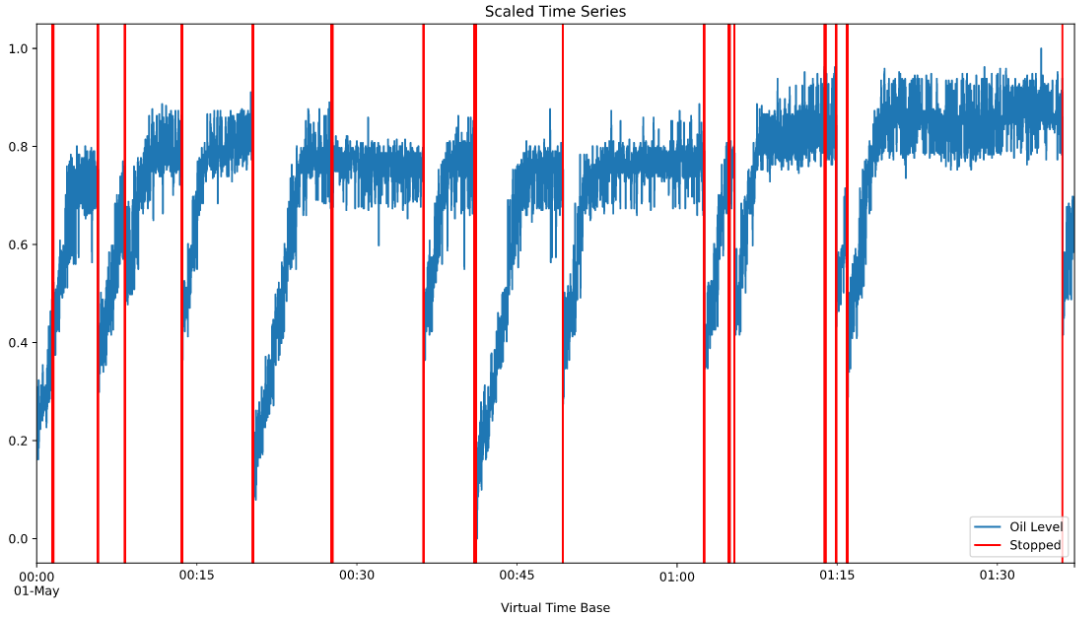


Figure 8.3: Scaled *OilLevel*

Following this phase is the *Data Reduction*, where the importance of the *Data Scaling* is noted.

8.1.3 Data Reduction

In this phase, the input are the scaled time series, in the case of *bosch_prensa01* there are 6 series, but there may be equipment that has hundreds of sensors, which generates an enormous amount of data and most of them may be redundant. *Data Reduction*, as explained in Chapter 3, aims to reduce the amount of data, maintaining all initial data patterns.

In this case, a *Dimensionality Reduction* strategy was used in order to avoid the curse of dimensionality. When there are few features, in this case, time series, there may be low results due to the lack of variety of data, however, when there are many features, there is the possibility that any existing observation would appear different from the rest due to the high number of comparison features.

Dimensionality Reduction eliminates some features of the dataset and creates new features that contain the necessary information about the initial dataset. In this case, the *Principal Component Analysis* (PCA) method was used.

PCA, as explained in Chapter 3, is a statistical technique that converts high dimensional data to low dimensional data, creating new features, *Principal Components*, which present the greatest variance. The previous step, *Data Scaling*, is a critical step when

applying PCA because PCA tries to create features that exhibit maximum variance, and as for features with higher magnitudes the variance is greater, PCA assigns more weight to features with a higher magnitude.

To use the PCA it is necessary to choose the number of *Principal Components* a priori. For this purpose, data from one of the times that the analysis of the segments was performed was used and all the *Principal Components* were calculated, as well as the variance explained by each component. Having obtained the results shown in Table 8.1. The *% of Variance* column gives the ratio, expressed as percentage, of the variance explained by each component to the total variance in all of the variables. The *Cumulative %* column gives the percentage of variance explained by the first m components, where m is the index of the *Principal Component* in question.

Table 8.1: Total variance using *Principal Components*

Number of *Principal Components*	**% of Variance**	**Cumulative %**
1	77.89	77.89
2	14.71	92.60
3	4.13	96.73
4	2.08	98.68
5	1.308	99.988
6	0.012	100

Through the analysis of Table 8.1, it is possible to verify that using 2 *Principal Components* it is possible to obtain 92.60% of the variability contained in the original 6 features. Thus, it is possible to reduce the complexity of the data by losing only 7.4 % of information. It is important to note that these values were obtained by simulating the data analysis at a given moment, however, as the analysis of the segments occurs every t_{for} minutes, each time the analysis is carried out these percentages are different.

Figure 8.4 shows the two *Principal Components* used. The next step is to detect anomalies using these two time series (features).

8.1.4 Anomaly detection

This is the last stage of the analysis of the segments, the objective is, based on the two *Principal Components*, to check if the condition of the equipment is within the normal, *Safe*, or if the condition of the equipment shows indications of a possible failure in the future, *Warning*. For this purpose, an anomaly detection is carried out on the data relating to *Data Reduction*.

This anomaly detection has 3 restrictions that are covered below.

- **Maximum virtual time in anomaly detection**;

  Not all segmented data is included in this search, there is a pre-defined maximum limit, t_{max}. t_{max} corresponds to the maximum duration, in minutes, of the dataset in the virtual time base. If after doing the merging segments the duration of the dataset is greater than t_{max} minutes, only the last t_{max} minutes of information are used. This limit on the duration of the dataset to be exploited is intended to

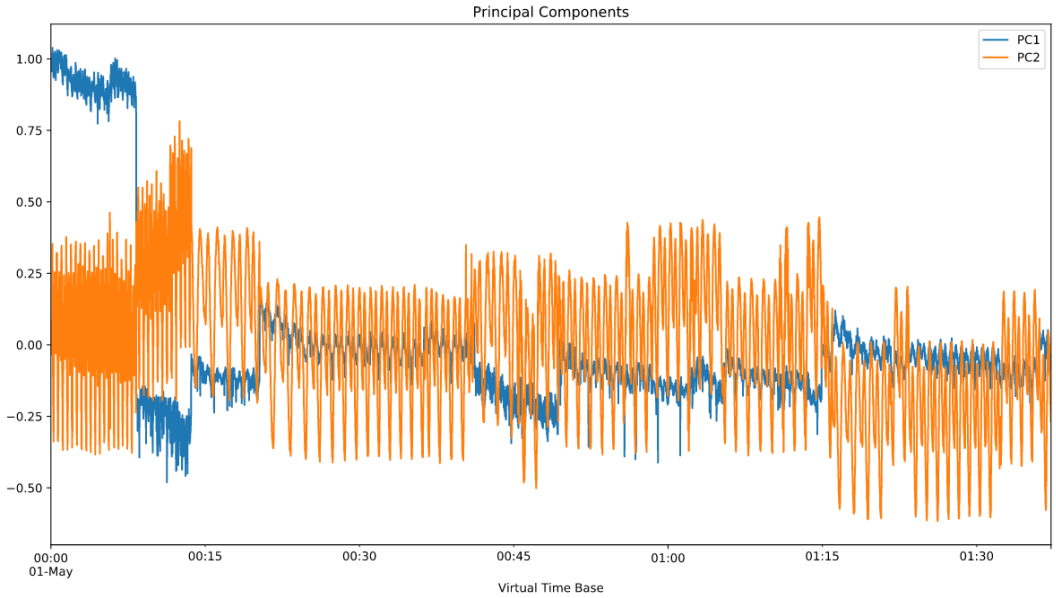


Figure 8.4: *Principal Components* used

reduce the processing time for anomaly detection, for example, if the equipment is running for consecutive days, it does not make sense to use all data in the anomaly detection.

- **Minimum virtual time in anomaly detection**;

  In the same way that there must be a maximum virtual limit of the dataset to be used, there must also be a minimum limit, t_{min}. In the beginning, when the developed system starts, there is not enough data to perform anomaly detection with decent results. Thus, if the virtual duration of the dataset is less than t_{min} minutes, the segment analysis is not carried out and, consequently, the forecast is not processed.

- **Useful anomaly detection period**.

  Although the last t_{max} minutes are used to search for anomalies, it makes no sense to consider the anomalies that occurred more than t_{for} minutes ago. As segment analysis occurs every t_{for} minutes, if anomalies are being detected more than t_{for} minutes ago, there is a case of counting the same anomaly twice. These t_{for} minutes, in contrast to t_{max}, relate to real-time.

  This restriction also guarantees that if the equipment is stopped for more than t_{for} minutes, anomalies will never be counted since only the anomalies detected in the last t_{for} real minutes are counted.

After counting the number of anomalies detected in the last t_{for} minutes, in the *Real Time Base*, it is necessary to compare this number with the threshold defined a priori, n_{ano}. If in the last t_{for} minutes more than n_{ano} anomalies are detected, the label *Warning* is sent, otherwise, the label *Safe* is sent. When in fact there is a problem with

the equipment, there are several anomalies in a short period of time, while when there is only one, the most likely cause is some poor measurement by a sensor.

For the detection of anomalies, an *Unsupervised Anomaly Detection* technique was used. *Unsupervised anomaly detection* techniques detect anomalies in an unlabeled dataset, assuming that most of the observations in the data are normal, looking for the observations that least resemble the rest of the observations, classifying them as anomalies.

In the development phase, several anomaly detection algorithms were tested in order to verify which algorithms best fit the existing data. Below is a list of the 3 best algorithms, explaining their concept in general.

- **Angle Based Outlier Detection (ABOD)**;

  ABOD in addition to analyzing the distance between observations, which is what most anomaly detection algorithms do, it also analyzes the angle between the distance vectors between two observations. One of the premises of the ABOD algorithm is that comparing the angles between pairs of distance vectors with other observations allows to distinguish between a standard observation or an anomaly [Kriegel *et al.* 2008].

- **Stochastic Outlier Selection (SOS)**;

  SOS uses the concept of affinity to quantify the relationship between two observations in a dataset. Affinity is proportional to the similarity between two observations, so an observation is considered an anomaly when the remaining observations have insufficient affinity [Janssens *et al.* 2012].

- **Copula-Based Outlier Detection (COPOD)**.

  COPOD is an anomaly detection algorithm inspired by copulas to model multidimensional data. COPOD initially builds an empirical copula, using it to predict tail possibilities of each observation to determine its "extremeness" level [Li *et al.* 2020].

In order to choose which algorithm was best suited to the data, two tests were performed. In the first test, data from the equipment *bosch_prensa01* were used in which it was already known that after a few minutes there would be a failure, and anomalies were detected in the previous 10 min using each of the three algorithms described above. In the second test, the procedure was the same, but in this case, the *bosch_prensa01* would not have any problems in the near future.

Table 8.2 shows the results of the two tests. All the algorithms used need a parameter that is the proportion of anomalies in the dataset, a_f, this parameter was determined using the trial and error method until the best result for each algorithm was obtained. By coincidence, all the algorithms had better results using this parameter at 0.005.

In the failure test, the objective was for the algorithms to detect the largest possible number of anomalies in the last 10 minutes, and in this perspective the algorithm with the best result was COPOD, detecting 4 anomalies. On the other hand, in the test without failure, the objective was not to detect any anomalies because the equipment had no failure in the future, and once again COPOD was the algorithm that showed the best results. Based on this analysis, it was concluded that COPOD is the algorithm that

Table 8.2: Number of anomalies detected in the last 10 minutes

Algorithm	**N° of anomalies in test with failure**	**N° of anomalies in test without failure**
ABOD	1	1
SOS	0	1
COPOD	4	0

best fits the data from the equipment *bosch_prensa01*. In addition to these tests, more tests were subsequently carried out, using more time to search for anomalies instead of 10 minutes, using other periods with failure and without failure, and COPOD always obtained the best results.

In Figure 8.5 and Figure 8.6, it is possible to view the test result with and without failure using the COPOD algorithm, respectively. The anomalies were superimposed on the two *Principal Components* because it was based on them that COPOD performed the anomaly detection. In Figure 8.5, as Table 8.2 shows, 4 anomalies were detected in the last 10 minutes, while in Figure 8.6 it is possible to verify that no anomalies were detected in the last 10 minutes.

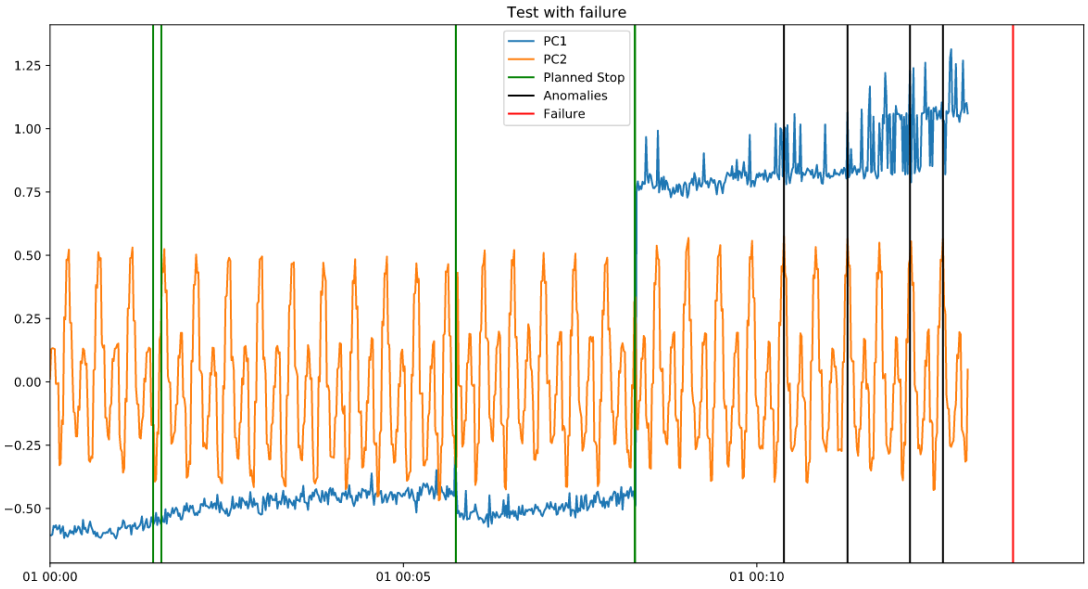


Figure 8.5: Test with failure

After COPOD detects anomalies, it is necessary to convert the date and time of the anomalies from the *Virtual Time Base* to the *Real Time Base*. Then, the real date and time of the anomalies are compared with the current time and all anomalies detected more than t_{for} minutes ago are eliminated. Finally, the remaining anomalies are counted, compared to the predefined threshold, n_{ano}, and the *Safe* or *Warning* forecast is sent to the database, which will subsequently go to *Grafana*.

When there is a problem in the equipment and the forecast *Warning* is sent, as soon as the equipment stops to perform the repair, the forecast *Warning* is replaced by *Safe*

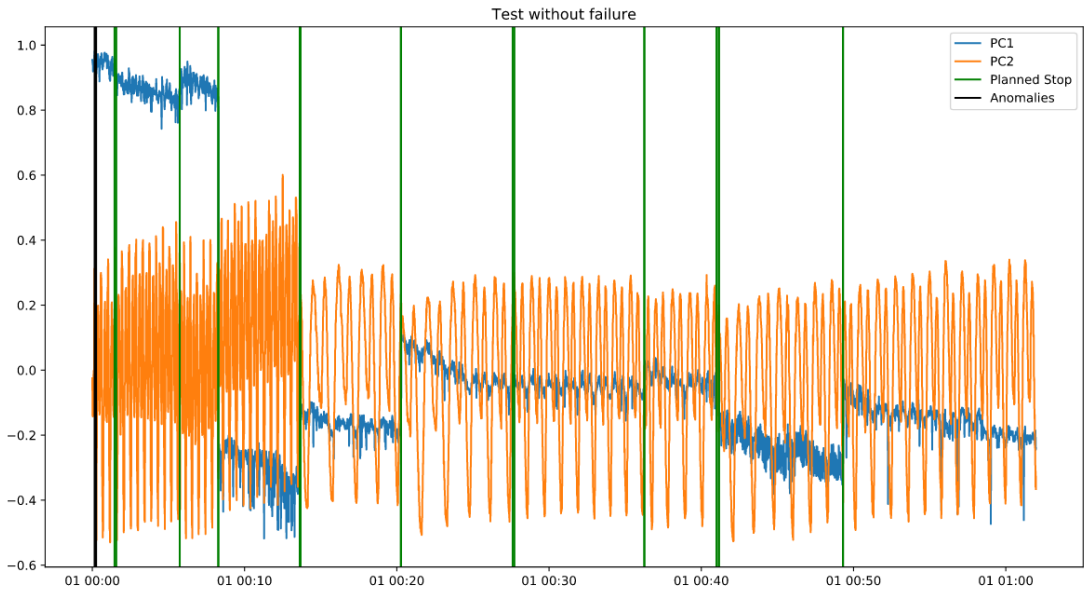


Figure 8.6: Test without failure

since in the last t_{for} real minutes, no new anomaly was detected.

It is important to mention that, as the anomaly detection algorithm is applied to the *Principal Components*, it is impossible to know which sensor allowed the detection of the anomaly, the anomaly could even have been caused by a combination of small variations in several time series. This means that when an anomaly is detected it is not possible to do *Reverse Engineering* to find the cause of the anomaly.

8.2 Visualization

This section presents the panels of the developed dashboard related to the forecast of the condition of the equipment *bosch_prensa01* as well as the visualization of the detected anomalies. This information is only available on the *bosch_prensa01* tab, because as previously mentioned, it is not possible to perform predictive analysis on the equipment *bosch_prensa02*.

Table 8.3 shows the values used for the parameters discussed above. These parameters were determined by trial and error and it is not possible to claim that this is the best combination of parameters.

In the panel of the dashboard illustrated in Figure 8.7, it is possible to view the time series related to the sensor *OilLevel* with an indication of when the anomalies occurred. The detected anomalies are represented by vertical purple lines.

The panel of the dashboard represented in Figure 8.8, displays the current forecast, in this case, *Safe*, indicating that in the last t_{for} minutes, in this case, 10 minutes, the number of anomalies detected was less than $n_{ano} + 1$, in this case, less than 4.

Finally, in the panel of the dashboard represented in Figure 8.9, a history of the forecasts made is presented. Thus, it is possible to compare this history with the equipment failure data to analyze the efficiency of the developed system.

Table 8.3: Parameter Values

Parameter	Value
t_{agg}	10
n_{mes}	5
t_{for}	10
n_{ano}	3
t_{max}	90
t_{min}	1
a_f	0.005

Figure 8.7: Overlapping time series segmented with detected anomalies

Figure 8.8: Current prediction

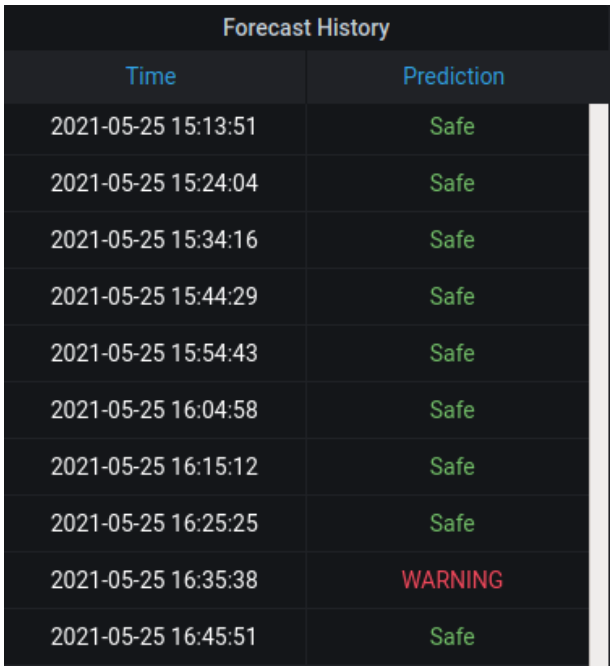


Figure 8.9: Forecast history

Intentionally blank page.

Chapter 9

Performance Analysis

In this chapter, the objective is to present an evaluation of the developed system.

Data relating to the two equipment used in Chapter 5 were used. Considering *bosch_prensa01*, a CSV file with data corresponding to the day 18/08/2020 was used, which contained data from the press from 06:16:12 to 21:31:56. The sending of data in real-time was made using a service that sends each measurement of the sensor simulating the work performed by an *Edge Device*, as was discussed in Chapter 5. As for the *bosch_prensa02* equipment, the module GY-91 was not placed on any equipment, having proceeded to the acquisition and sending of data to the system when the sensor was placed on a static table.

As the developed system works with data in real-time, the system was left to run for 15:15:44, from 25/05/2021 at 13:51:47 to 26/05/2021 at 05:07:31 receiving data in real-time from the two equipment covered. During that period, a simulation of the system in production was carried out. The parameters of the developed system used are those found in Table 8.3.

In the sections below, the results obtained are presented, first analyzing the developed dashboard and then analyzing the forecasts made.

9.1 Dashboard Developed

This section presents the dashboard developed after all the information has been sent and processed. Figure 9.1 and Figure 9.2 show the dashboard for *bosch_prensa01* and *bosch_prensa02*, respectively. In the upper-right corner of the dashboard, it is possible to change to any equipment that is using the developed system.

Statistical data relating to equipment, *Number of Equipment*, *Working* and *Stopped* do not vary depending on the selected equipment as they are related to all equipment. At the end of the simulation, there was one equipment stopped and one working, this information can also be verified through the visualization of the active segment at the end of the simulation in Figure 9.1 and Figure 9.2. In the case of *bosch_prensa01*, the red segment is active, meaning that the equipment was stopped and in the case of *bosch_prensa02*, the green segment is active, symbolizing that, at the end of the simulation, the equipment was working.

Taking a closer look at Figure 9.1, it is possible to verify that the time series presented in the panel *Time Series Segmentation* concerns the sensor *OilLevel*. This was chosen in order to give continuity to what was shown throughout the document. In addition to

the segmented time series, on the panel *Time Series Segmentation*, it is also possible to view vertical purple lines identifying the anomalies detected.

The panel *Forecast History* shows all the forecasts issued by the system, making it possible, through its comparison with the equipment's fault record, to evaluate the system developed with regard to predictive maintenance. Through the analysis of the time increment between successive forecasts it is possible to confirm that the time interval in which the analysis of the segments was carried out, t_{for}, is in fact 10 minutes, as shown in Table 8.3.

Approaching Figure 9.2 in detail, it is possible to verify that the time series relates to the measurement *ShaftVibrationX*. As expected, there are no major variations given that the sensor has been on a static table all the time. The red segments, which indicate that the equipment is stopped, were intentionally caused by cutting off the ESP32 power.

It is possible to verify that there are no anomalies in the panel *Time Series Segmentation*, nor any information related to the predictions made, since in this equipment, as previously mentioned, segment analysis was not carried out.

Considering everything mentioned in this section, it is possible to conclude that problems 1 and 3 mentioned in Section 4.1 have been solved. As for problem 1, the developed system receives data from multiple equipment, stores and processes the data separately and finally, creates a dashboard that contains the data of all the equipment. As for problem 3, the implemented dashboard presents a pleasant interface, ease of adjusting the time window in which the data is to be visualized and the possibility of viewing all the equipment in one place.

Figure 9.1: Dashboard related to *bosch_prensa01*

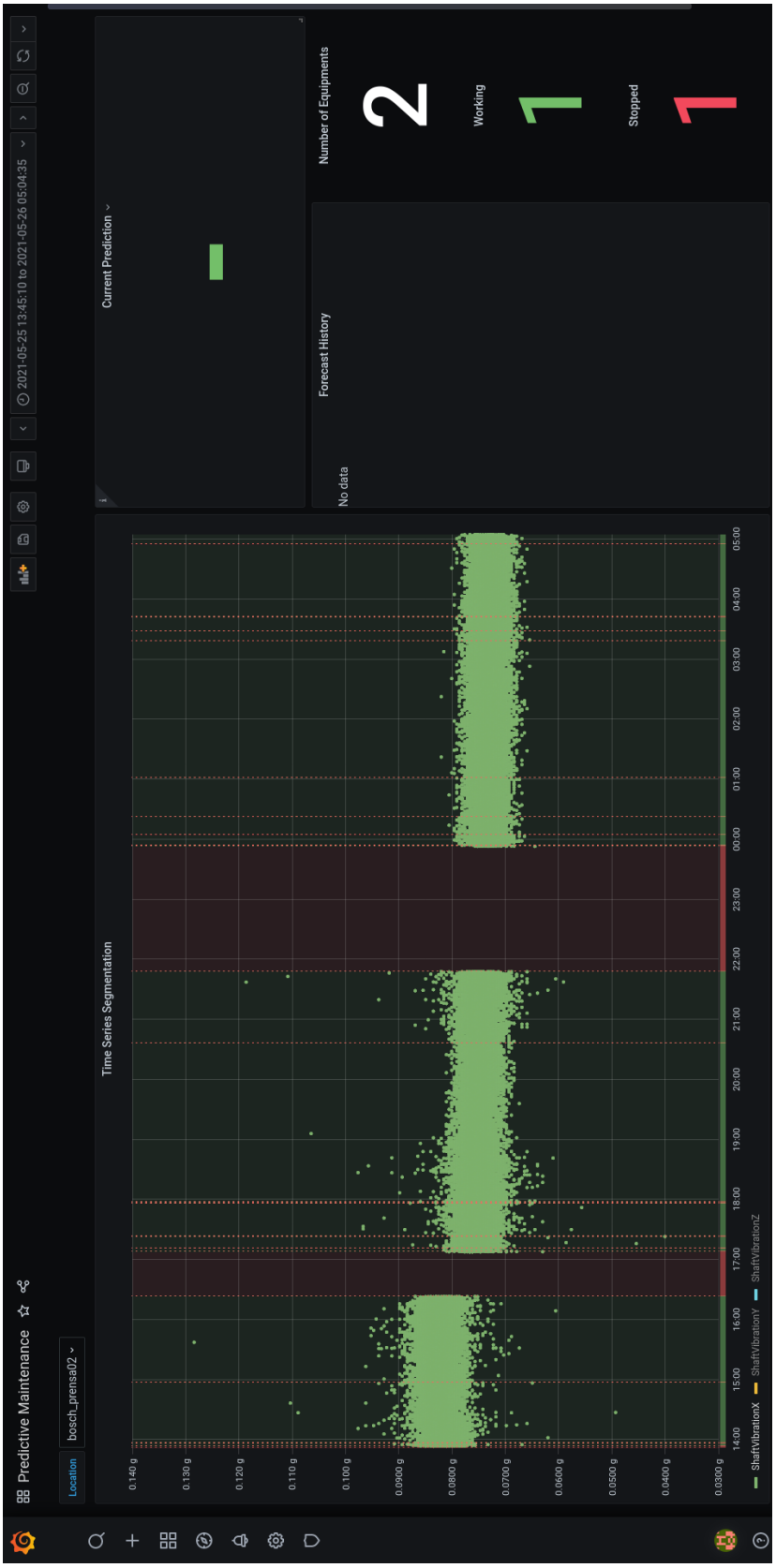


Figure 9.2: Dashboard related to *bosch_prensa02*

9.2 Forecast Analysis

In this section, the forecasts issued for the *bosch_prensa01* during 15:15:44 are evaluated.

In order to analyze the quality of the predictions made, in addition to the history of the forecasts made, present in the panel *Forecast History* of Figure 9.1, it is also necessary to know the description of each stop of the equipment, allowing the comparison between the forecasts made and the description of each stop. For this, a shift book was consulted that contains descriptions of all the stops of the day 18/08/2020. The shift book consulted contains the description of the stops on the date and time they occurred, however, the forecasts made were issued at the time of the simulation, causing a discrepancy in the dates. To resolve this discrepancy, a mapping of the dates and times in the shift book was made to match the simulation dates. In this way, as both the descriptions of the stops and the forecasts made were in the dates and times of the simulation, it was possible to carry out the comparison.

Table 9.1 presents a summarized version of the shift book already adapted for the dates on which the developed system was simulated. The column *Index* is used to associate each stop with a number to facilitate the identification of each stop. The column *Description* contains a brief explanation of the description for each stop. In this case, there were 5 different types of stops, below is a brief explanation of each one.

- **Container Exchange**;

  Whenever the container that receives the parts produced by the press is full, the press stops automatically for the container to be exchanged.

- **Ingrown Sheet**;

  Interlocking of the metal sheet in the die.

- **Interval**;

  Break for workers.

- **Mechanical Breakdown**;

  Oil leakage.

- **Die Exchange**.

  Die exchange to produce new parts.

The columns *Begin*, *End* and *Duration* correspond to the start date of the stop, end date of the stop and duration of the stop, respectively. Finally, the column *Failure* identifies whether the stop was caused by a failure or not, as previously mentioned, not all stops are necessarily caused by failures.

Although Table 9.1 records only 12 stops, by consulting the segments, it is possible to verify that in reality, there were 33 stops of the press. The main reason for this discrepancy is the existence of many short stops that have not been reported by the workers. Of the 12 stops reported there were only 2 stops that were caused by equipment malfunctions. During all simulation, 89 forecasts were issued, of which 6 were *Warning* and the rest *Safe*.

For the relationship between each stop and the forecasts issued, the methodology exemplified in Figure 9.3 was adopted. For the analysis of stop p, the forecasts that

Table 9.1: Shift book adapted to the day of the simulation

Index	Description	Begin	End	Duration	Failure
0	Container Exchange	25/05/2021 13:59:32	25/05/2021 14:02:54	00:03:22	No
1	Ingrown sheet	25/05/2021 14:25:56	25/05/2021 15:36:05	01:10:09	Yes
2	Interval	25/05/2021 15:46:43	25/05/2021 16:28:24	00:41:41	No
3	Container Exchange	25/05/2021 16:59:15	25/05/2021 17:21:26	00:22:11	No
4	Container Exchange	25/05/2021 17:32:52	25/05/2021 17:37:57	00:05:05	No
5	Container Exchange	25/05/2021 17:55:49	25/05/2021 18:04:08	00:08:19	No
6	Interval	25/05/2021 18:07:38	25/05/2021 18:15:04	00:07:26	No
7	Container Exchange	25/05/2021 18:27:46	25/05/2021 19:05:53	00:38:07	No
8	Mechanical Breakdown	25/05/2021 19:33:11	26/05/2021 00:48:08	05:14:57	Yes
9	Interval	26/05/2021 01:35:17	26/05/2021 01:47:56	00:12:39	No
10	Container Exchange	26/05/2021 02:30:17	26/05/2021 02:39:18	00:09:01	No
11	Die Exchange	26/05/2021 02:42:34	26/05/2021 03:57:40	01:15:06	No

are temporally between stop *p-1* and *p* are used, this forecast period will be referred to throughout the document as useful forecast period. For example, a successful case is when *stop p* was caused by a malfunction and at least one of the forecasts from *F3* to *F9* was a *Warning*.

Table 9.2 presents the results of the forecasts associated only with the stops that were caused by malfunctions, stops *1* and *8*. The column *Failure Detected?* identifies whether, in the useful period of the forecasts, there was at least one *Warning* issued, the column *Failure detection time* indicates, in case the fault has been detected, what was the time between the first *Warning* issued and the stop, finally, the column *N° of anomalies detected* indicates, in case the fault has been detected, the number of anomalies detected in the useful forecast period.

As Table 9.2 indicates, the two stops caused by equipment malfunctions have been successfully detected. Stop 1, which lasted about 1 hour, 10 minutes and 9 seconds was detected 4 minutes and 23 seconds before it occurred and stop 8, which lasted about 5 hours, 14 minutes and 57 seconds was detected 23 minutes and 51 seconds before the stop occurred.

Table 9.3 shows the results of the forecasts associated with the stops that were not caused by malfunctions. The table structure is exactly the same as Table 9.2.

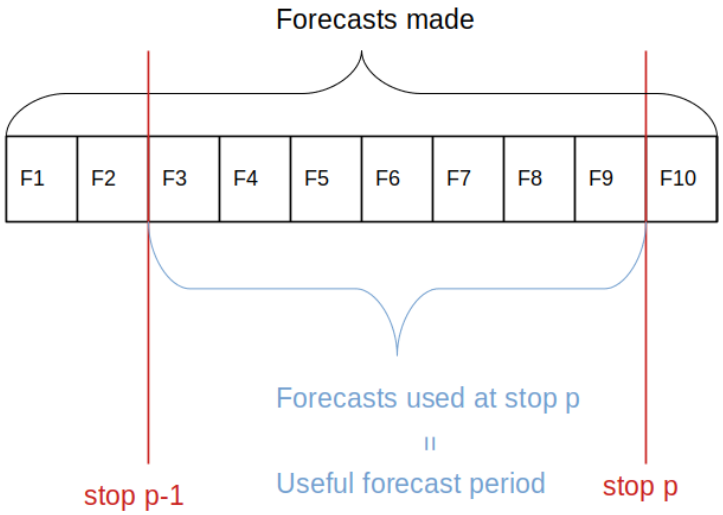


Figure 9.3: Methodology of analysis of each stop

Table 9.2: Analysis of stops caused by failures

Index	Failure detected?	Failure detection time	N° of anomalies detected
1	Yes	00:04:23	5
8	Yes	00:23:51	17

As shown in Table 9.3, *Warnings* alerts were issued before 2 stops that were not caused by malfunctions. These two situations can be classified as false positives, since the system warned that the behavior of the equipment was abnormal, and the next stop was not caused by any problem. It is important to note that although the next stop was not caused by a malfunction, it is possible that, in fact, the equipment's behavior was abnormal, and that if the stop was not so early, there could have been a bigger problem.

Regarding all stops that were not reported in the system, the developed system did not issue any *Warning* alerts.

To evaluate the performance of the predictions in a more objective way, a *Confusion Matrix* was elaborated. The *Confusion Matrix* compares the actual values with the predictions made, allowing to obtain a more complete view of the performance of the forecasts. Figure 9.4 shows the *Confusion Matrix* in general, indicating the name of each of its elements.

Table 9.3: Analysis of stops not caused by failures

Index	Failure detected?	Failure detection time	N° of anomalies detected
0	No	-	-
2	No	-	-
3	Yes	00:23:37	4
4	No	-	-
5	No	-	-
6	No	-	-
7	No	-	-
9	No	-	-
10	Yes	00:29:35	10
11	No	-	-

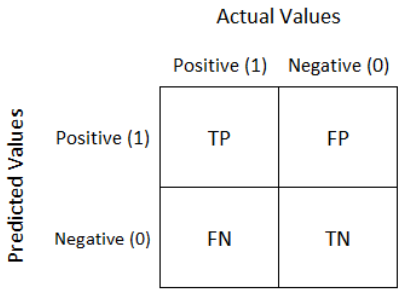


Figure 9.4: *Confusion Matrix* in general

In the list below, each of the constituent elements of the *Confusion Matrix* is explained, taking into account the current problem.

- **TP - True Positive**;

  The model predicts a positive value and it is true. In this specific case, the stop is caused due to a malfunction in the equipment, and the model issued at least one *Warning* in the useful forecast period.

- **TN - True Negative**;

  The model predicts a negative and it is true. In this specific case, the model did not issue any *Warning* in the useful forecast period, and the stop is not caused by any malfunction.

- **FP - False Positive (Type 1 Error)**;

  The model predicts positive and it is false. In this specific case, the model issued at least one *Warning* in the useful forecast period, however, the stop is not caused by malfunctions.

- **FN - False Negative (Type 2 Error)**.

  The model predicts negative and it is false. In this specific case, the model did not issue any *Warning* in the useful forecast period, however, the stop is caused by a malfunction.

A *Type 2 Error* is more serious than the *Type 1 Error*. While in the *Type 1 Error*, it is only necessary to stop the equipment for a general overhaul, in the *Type 2 Error*, there is an undetected fault, resulting in economic losses both in downtime for maintenance and in the maintenance itself. Therefore, although both situations are considered wrong forecasts, for a correct evaluation of the forecasts made, these two errors cannot have the same weight.

Table 9.4 shows the obtained confusion matrix. In this case, in addition to the reported stops, unreported stops were also considered, considering 33 stops.

Table 9.4: *Confusion Matrix* obtained

Actual / **Predicted**	**Positive (1)**	**Negative (0)**
Positive (1)	2	2
Negative (0)	0	29

Through the analysis of Table 9.4, it is verified that there were 2 true positives, 2 false positives, 0 false negatives and 29 true negatives. Based on this information it is possible to calculate various metrics used to classify the performance of forecasts, including *Accuracy*, *Recall*, *Precision* and $F_\beta - Score$. Depending on the problem, there are more appropriate metrics than others.

Accuracy is defined as the ratio between the correct forecasts and the total number of forecasts. Using the elements present in the *Confusion Matrix*, it can be calculated using Equation 9.1 [Galdi and Tagliaferri 2018].

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{9.1}$$

Precision is defined as the ratio between correct positive forecasts and the total number of positive forecasts. Using the elements present in the *Confusion Matrix*, it can be calculated using Equation 9.2 [Goutte and Gaussier 2005]. This metric focuses on the accuracy of the model with respect to positive predictions.

$$Precision = \frac{TP}{TP + FP} \tag{9.2}$$

Recall is the ratio between correct positive forecasts and the number of total actual positive. Equation 9.3 [Goutte and Gaussier 2005] concerns the way of calculating this metric using the elements present in the confusion matrix. This metric focuses on the accuracy of the model to detect the actual positives.

$$Recall = \frac{TP}{TP + FN} \tag{9.3}$$

$F_\beta - Score$ is the weighted average of *Precision* and *Recall*, Equation 9.4 [Goutte and Gaussier 2005]. β is the number of times that the recall is more important than the precision.

$$F_\beta - Score = (1 + \beta^2) \cdot \frac{Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall} \tag{9.4}$$

In most cases of predictive maintenance, there is an unbalanced dataset, and this case is no different. The press stops much more often for reasons unrelated to damage than the other way around, hence the cause of the unbalanced dataset. Based on this reason, and the difference in weights between a false positive and a false negative (a false negative means that there was a malfunction not detected whereas a false positive only implies a review of the equipment's operation), it is concluded that the metric *Accuracy* is not the most recommended to measure the performance of the predictions made. *Precision* and *Recall* are quite interesting as they focus on false positives and false negatives, respectively. However, it is incorrect, in this case, only to take into account either false positives or false positives, because although false negatives are more serious than false positives, a false positive still involves stopping the equipment to check whether there is or not a problem and therefore there are also losses in this situation. That said, it is concluded that the most appropriate metric to classify the performance of the forecasts made is the $F_\beta - score$ with $\beta = 3$, evaluating in this case, both false negatives and false positives, giving 3 times more weight to false negatives.

Table 9.5 shows the values obtained using the 4 metrics explained above.

Table 9.5: Evaluation of forecast performance

Accuracy	**Precision**	**Recall**	$\boldsymbol{F_\beta - Score}$
0.9394	0.5	1.0	0.9091

The *Accuracy* obtained was 93.94%, which means that of the forecasts made, 93.94% were correct, as mentioned above, this value does not correctly assess the efficiency of the forecasts. As for *Precision*, the result was only 50%, that is, of all the times the system predicts that the equipment will stop due to a malfunction, it only gets it right half the time. Regarding *Recall*, an efficiency of 100% was obtained, meaning that all stops caused by breakdowns were correctly predicted. Finally, as for the $F_\beta - Score$, 90.91% was obtained, which is a very satisfactory result.

The results presented in Table 9.5 are quite satisfactory, especially the metric $F_\beta - Score$, which is the most relevant for the problem in question, as explained previously. However, through the analysis of the *Confusion Matrix*, it is possible to verify that the total number of observations is reduced and that in order to better test the developed system, more data from the press should be used, namely more data containing stops caused by malfunctions.

Despite a reduced set of data, it is possible to conclude that problem 2, described in Section 4.1, was successfully solved, because if on day 18/08/2020 the developed system was in production, the two stops that occurred due to malfunctions would not happen, since before stops occur, *Warning* alerts would have been issued that would allow equipment to be stopped and maintained.

9.3 Economic Analysis

In this section, we intend to evaluate the performance of the developed system at an economic level, analyzing in a first phase the cost of implementing the developed system and, in a second phase, the economic benefits. This analysis is only done on the equipment *bosch_prensa01*, since the equipment *bosch_prensa02* has no economic benefits, it only served to test the time series segmentation using data from multiple equipment.

Regarding the cost of implementing the developed system, the economic costs are reduced. The equipment *bosch_prensa01* already has all the necessary sensors and also has a PLC to serve as *Edge Device*. It would only be necessary to change the way the PLC sends the data, instead of sending the sensor data in XML files to the *Nexeed MES*, it is necessary to send the sensor data via HTTP requests to the REST API, as explained in Chapter 5. The cost associated with this change could be accounted for in hours when the equipment is stopped and in hours paid to engineers to reprogram the code of the PLC, however, this cost, in addition to being difficult to quantify, is relatively low.

With regard to the benefits of implementing the developed system, not all features are quantifiable, such as the dashboard developed and the centralization of data from different equipment. However, it is possible to quantify the predictive maintenance system developed for the equipment *bosch_prensa01*, and this is where this analysis focuses. Two types of economic benefits can be considered, the cost of corrective maintenance that would no longer be necessary and the cost associated with the time that the equipment is not producing. As for the first factor, the cost depends on the malfunction itself, and it was not possible to obtain the economic value of the two corrective maintenance (stop 1 and 8) that took place on 18/08/2020. As for the second factor, Bosch Thermotechnology, S.A., like other companies, uses a flexible production plan. This plan consists of sending operators home when the equipment is out of order, with the operators being responsible for working overtime to compensate for the time the equipment has not been in production, including weekends. In this way, it is difficult to quantify the cost associated with the time the equipment does not produce, as these hours will be recovered later. Therefore, the economic analysis will focus more on the time that could have been "gained" on 18/08/2020, than on monetary values.

Concerning the time the equipment is not producing, it is necessary to take into account two factors, on the one hand, it is necessary to consider the time "gained" with the fact of avoiding equipment failures, on the other hand, it is necessary to count the time that is "lost" whenever there is a *Warning* alert, since only 50% of the times a *Warning* is issued there is actually a failure in the equipment, Table 9.5.

For the purpose of accounts, it is considered that whenever a *Warning* is issued, the machine stops for 15 minutes for the maintenance team to check the status of the equipment. In the case where, in addition to the check up, it is also necessary to make some correction in the equipment, a stop time of 30 minutes is considered. These values were obtained through an analysis of the shift book where all maintenance performed on the equipment is recorded.

As can be seen in Table 9.4, the two stops caused by malfunction were detected with some antecedence, so it can be said that the 1 hour, 10 minutes and 9 seconds associated with the duration of the stop 1 and the 5 hours, 14 minutes and 57 seconds associated with the duration of the stop 8, could have been avoided. In total, if the system had been in production, the equipment could have worked another 6 hours, 25 minutes and

6 seconds, without considering the time spent on checkup and maintenance. As for the time spent on check up and maintenance, at stops 1 and 8, it is considered that the equipment was stopped for 30 minutes each (check up and maintenance) and at stops 3 and 10, it is considered that the equipment was stopped 15 minutes each (check up). In total, the equipment would be stopped due to *Warnings* alerts for 1 hour and 30 minutes. Combining the time "gained" and "lost", it is concluded that if the developed system was in production on that day, the equipment would work another 4 hours, 55 minutes and 6 seconds.

Although only an analysis based on time was made, it is possible to verify the positive impact that the implementation of the developed system would have, not only on the equipment *bosch_prensa01*, but also on any equipment.

Chapter 10

Conclusion and Future Work

The presented work proposes a system capable of converging data from multiple equipment and performing predictive maintenance on each one of them. The work developed offers a solution of the entire process, from the communication between the sensor and the *Edge Device* to the sending of alerts to a dashboard in order to alert the maintenance team in case any intervention is needed.

This project proposes a new approach with regard to predictive maintenance systems. This new approach is based on the segmentation of time series from different sensors in different states. In this way, in the analysis phase, only data belonging to the same state are compared, making it possible to extract patterns and detect anomalies using less complex algorithms. This new approach rivals the more traditional form of predictive maintenance systems that use the data as a whole and use complex ML algorithms, such as *Recurrent Neural Network* (RNN), to infer patterns in the data. With the proposed approach, it was possible, using an anomaly detection algorithm, to predict future breakdowns on a mechanical press based on the created segments with a $F_\beta - Score$ of 90.91%. As mentioned in Chapter 9, the amount of data used to test the predictions made has been reduced, both in duration, 15 hours, 15 minutes and 44 seconds, and in the number of existing failures, 2. Thus, it is difficult to accurately assess the efficiency of the results obtained, however, the results obtained show very promising indications, allowing us to conclude that the proposed approach is a viable solution for predictive maintenance systems.

Although in this project, the segmentation focused on the separation of the time series in *Working* and *Stopped* segments, this is only one of the many possibilities. The objective is to divide the time series into segments to subsequently analyze and compare data belonging to the same segment. In the case of a mechanical press, segmenting the data in the warming period and stable period would also be a good idea as it would be possible in the analysis phase to detect anomalies by comparing data only belonging to the warming period or only belonging to the stable period. Certainly, using this way it would be much easier to detect patterns than to use the data all together.

The developed system presents several parameters that have to be defined a priori for each of the existing equipment, namely, t_{agg}, n_{mes}, t_{for}, n_{ano}, t_{max}, t_{min} and a_f. The purpose of these parameters is to offer versatility to the developed system, allowing the segmentation and analysis of data from equipment with different characteristics. A good example is the parameter t_{for}, if a equipment performs several operations per second it makes sense to decrease t_{for} to issue more alerts, however, if the equipment only

performs one operation per minute, it makes sense to increase t_{for}. In the case of the equipment *bosch_prensa01*, these parameters were chosen by trial and error. Even the type of algorithm used to detect anomalies can be chosen depending on the equipment, in the case of the equipment *bosch_prensa01*, *COPOD* was chosen because it was the one that, on the one hand, best detected anomalies before failures, and on the other, it was little susceptible to false positives.

The present work arose from the need to solve three problems, namely, the existence of several recipients for the data produced by IoT Devices, a high number of unplanned stops at the *Haulick & Roos* and data visualization platform with several limitations. The first problem was solved based on the pipeline developed from the sensor to the database, Chapter 5, Chapter 6 and Chapter 7 addresses all the steps that led to the resolution of this problem. The second problem was the problem that triggered the proposed approach and Chapter 7 and Chapter 8 presents the necessary steps to solve this problem. Finally, the third problem was solved by creating a dashboard that presents all the required requirements, as can be seen in Section 9.1.

10.1 Future Work

The work developed is by no means completed. Although all initial goals were successfully completed there is still room for improvement.

The parameters of the developed system present in Table 8.3, in addition to offering more versatility to the system, also increases the complexity of implementing the system. In the future, it would be useful to develop an optimization algorithm capable of selecting the best parameters for each equipment, namely, in this case, for the equipment *bosch_prensa01*. Thus, instead of wasting time testing some combinations of parameters, as happened in this project, the algorithm would select the combination that maximizes the efficiency of a specific metric, namely the $F_\beta - score$.

Another useful improvement would be the use of more data referring to the *Haulick & Roos* press in order to verify if the results obtained are, in fact, representative and not just a consequence of the reduced size of the data used.

Moreover, it would be interesting to evolve the segmentation performed into two segmentation phases, the first phase segmenting the time series into *Working* and *Stopped*, and the second phase segmenting the *Working* segments into *Warming* and *Stable* segments. Thus, it will be expected that the anomaly detection algorithms have a greater efficiency than the proposed solution.

It would also be important to adapt the way of sending data from the *Haulick & Roos* press, so that instead of sending XML files to *Nexeed MES*, send HTTP requests to the REST API developed, Chapter 6. In addition, there would be advantages to installing the data acquisition and sending pipeline related to *bosch_prensa02*, Section 5.1, in the *Haulick & Roos* press, because the cadence obtained is much higher than the current one.

In the future, it would also be pertinent, in addition to receiving data from the *Haulick & Roos* press, to use data from other real equipment to test the behavior of the developed system.

Bibliography

[Agrawal and Ratnadip 2013] R.K Agrawal and Adhikari K Ratnadip. An Introductory Study on Time Series Modeling and Forecasting Ratnadip Adhikari R. K. Agrawal. *arXiv preprint arXiv:1302.6613*, 1302.6613:1–68, 2013.

[Arvind 2019] Arvind. Docker Architecture: Why is it important? https://www.edureka.co/blog/docker-architecture/, 2019.

[Baccar 2019] Yacine Ben Baccar. Comparative Study on Time Series Forecasting Student : School Supervisor : cois ROEUFF Erwann LePennec Company Supervisor : Department of Information Technologies. (November):1–92, 2019.

[Barnett and Lewis 1994] V Barnett and T Lewis. *Outliers in Statistical Data*. Wiley, 1994.

[Bosch 2020] Bosch. Bosch em Portugal. https://www.bosch.pt/a-nossa-empresa/bosch-em-portugal, Retrieved in 2020.

[Cang and Seetaram 2012] Shuang Cang and Neelu Seetaram. Time series analysis. *Handbook of Research Methods in Tourism: Quantitative and Qualitative Approaches*, (June):47–70, 2012.

[Cayetano and Alonso 2017] Rubén Cayetano and Díaz Alonso. Software Containerization With Docker. 2017.

[Chappell 2004] D Chappell. *Enterprise Service Bus*. O'Reilly Media, 2004.

[Cho 2019] Young Cho. Implementing Predictive Maintenance Forward-. 2019.

[Chou and Yao 2009] Ying Chieh Chou and Leehter Yao. Automatic diagnosis system of electrical equipment using infrared thermography. *SoCPaR 2009 - Soft Computing and Pattern Recognition*, pp. 155–160, 2009.

[Chris Coleman *et al.* 2017] Chris Coleman, Satish Damodaran, Mahesh Chandramouli and Ed Deuel. Making maintenance smarter. *Deloitte University Press*, 2017.

[Cloudera 2019] Cloudera. Apache Kafka Guide. 2019.

[Cochrane 2005] John H Cochrane. Time series for macroeconomics and finance. *Manuscript, University of Chicago*, (773):1–136, 2005.

[DB-Engine 2021] DB-Engine. DB-Engines Ranking of Time Series DBMS. https://db-engines.com/en/ranking/time+series+dbms, 2021.

[Diez-Olivan *et al.* 2019] Alberto Diez-Olivan, Javier Del Ser, Diego Galar and Basilio Sierra. Data fusion and machine learning for industrial prognosis: Trends and perspectives towards Industry 4.0. *Information Fusion*, 50(July 2018):92–111, 2019.

[Dobbelaere and Sheykh Esmaili 2017] Philippe Dobbelaere and Kyumars Sheykh Esmaili. Kafka versus RabbitMQ. *arXiv*, (September), 2017.

[Efthymiou *et al.* 2012] K. Efthymiou, N. Papakostas, D. Mourtzis and G. Chryssolouris. On a predictive maintenance platform for production systems. *Procedia CIRP*, 3(1):221–226, 2012.

[Fawumi 2015] Kehinde Fawumi. Design of an Interactive and Web-Based Software for the Management, Analysis and Transformation of Time Series. 2015.

[Filipe and Gameiro 2020] Carlos Filipe and Teixeira Gameiro. Predictive Maintenance of Electrical Grid Assets. 2020.

[Forjan 2019] James Forjan. Time-series Data vs. Cross-sectional Data. https://analystprep.com/cfa-level-1-exam/quantitative-methods/time-series-data-vs-cross-sectional-data/, 2019.

[Futter 2017] Kim Futter. Time Series Analysis and Modeling with the Air Passengers Dataset. https://rpubs.com/kimnewzealand/311446, 2017.

[Galdi and Tagliaferri 2018] Paola Galdi and Roberto Tagliaferri. Data mining: Accuracy and error measures for classification and prediction. *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics*, 1-3(January):431–436, 2018.

[Galli 2020] Soledad Galli. Python Feature Engineering Cookbook. Packt Publishing, 1 edition, 2020.

[Gonçalves Da Silva and Ferreira 2019] João Pedro Gonçalves Da Silva and João Carlos Ferreira. A Predictive Maintenance Approach based in Big Data Analysis. 2019.

[Goutte and Gaussier 2005] Cyril Goutte and Eric Gaussier. A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation. *Lecture Notes in Computer Science*, 3408(April):345–359, 2005.

[Guo *et al.* 2017] Liang Guo, Naipeng Li, Feng Jia, Yaguo Lei and Jing Lin. A recurrent neural network based health indicator for remaining useful life prediction of bearings. *Neurocomputing*, 240:98–109, 2017.

[Han *et al.* 2011] Jiawei Han, Micheline Kamber and Jian Pei. Data mining: Data mining concepts and techniques. 2011.

[Hawkins 1980] D Hawkins. *Identification of Outliers*. Chapman and Hall, 1980.

[Hochreiter 1997] Sepp Hochreiter. Long Short-Term Memory. 1780:1735–1780, 1997.

[Hussien and Alla 2018] Mohammed Hussien and Abd Alla. Predictive Maintenance by Using Ultrasound Technique for Rotating Equipments in Thermal Power Plants. (Batch 10), 2018.

[J. M. Wetzer *et al.* 2000] J. M. Wetzer, C. J. Cliteteur, W. R. Rutgers and H. F .A Verhaart. Diagnostic-space and condition assessment techniques for condition based maintenance. *Conference on Electrical Insulation and Dielectric Phenomena*, pp. 47–51, 2000.

[Janssens *et al.* 2012] J.H.M. Janssens, F. Huszar, E.O. Postma and H.J. van den Herik. Stochastic Outlier Selection. 2012.

[Jimenez-Cortadi *et al.* 2020] Alberto Jimenez-Cortadi, Itziar Irigoien, Fernando Boto, Basilio Sierra and German Rodriguez. Predictive maintenance on the machining process and machine tool. *Applied Sciences (Switzerland)*, 10(1), 2020.

[Komorowski *et al.* 2016] Matthieu Komorowski, Dominic C Marshall, Justin D Salciccioli and Yves Crutain. Secondary Analysis of Electronic Health Records. *Secondary Analysis of Electronic Health Records*, (September):1–427, 2016.

[Krar 1994] S. Krar. The importance of maintenance. Manufacturing Automation, 1994.

[Kriegel *et al.* 2008] Hans Peter Kriegel, Matthias Schubert and Arthur Zimek. Angle-based outlier detection in high-dimensional data. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 444–452, 2008.

[Kumar 2020] Atul Kumar. Docker Architecture: A Complete Docker Introduction. https://k21academy.com/docker-kubernetes/docker-architecture-docker-engine-components-container-lifecycle/, 2020.

[Kwak and Kim 2017] Sang Kyu Kwak and Jong Hae Kim. Statistical data preparation: Management of missing values and outliers. *Korean Journal of Anesthesiology*, 70(4):407–411, 2017.

[Lange 2016] Kenneth Lange. THE LITTLE BOOK ON REST SERVICES by Kenneth Lange. 2016.

[Lessing 2020] Marlese Lessing. Containers vs VMs: Key Differences. https://www.sdxcentral.com/containers/definitions/containers-vs-vms/, 2020.

[Li *et al.* 2020] Zheng Li, Yue Zhao, Nicola Botta, Cezar Ionescu and Xiyang Hu. COPOD: Copula-based outlier detection. *Proceedings - IEEE International Conference on Data Mining, ICDM*, 2020-November(September):1118–1123, 2020.

[Mack *et al.* 2018] Christina Mack, Zhaohui Su and Daniel Westreich. Managing Missing Data in Patient Registries: Addendum to Registries for Evaluating Patient Outcomes: A User's Guide, Third Edition. pp. 7–8, 2018.

[Martin *et al.* 2015] N. Martin, M. Swennen, B. Depaire, M. Jans, A. Caris and K. Vanhoof. Batch Processing: Definitionand Event Log Identification. *Research Gate*, , 2015.

[Mayeen 2015] S. Mayeen. Bosch improves sales in all business sectors. https://news.cision.com/torque/r/bosch-improves-sales-in-all-business-sectors,c9765873, 2015.

[Mikolov *et al.* 2010] Tomaš Mikolov, Martin Karafiát, Lukaš Burget, Cernocky Jan and Sanjeev Khudanpur. Recurrent neural network based language model. *Proceedings of the 11th Annual Conference of the International Speech Communication Association, INTERSPEECH 2010*, (September):1045–1048, 2010.

[Narkhede *et al.* 2007] N Narkhede, G. Shapira and T. Palino. *Kafka: The Definitive Guide Real-Time Data andStream Processing at Scale, 1st ed.* O'Reilly Media, 2007.

[Nubera 2021] Nubera. Docker. https://www.nubera.eu/docker, Retrieved in 2021.

[Ostashchuk 2017] Oleg Ostashchuk. Time Series Data Prediction and Analysis Oleg Ostashchuk. (May), 2017.

[Perera and Alwis 2017] S. Perera and R. Alwis. Machine Learning Techniques for Predictive Maintenance. https://www.infoq.com/articles/machine-learning-techniques-predictive-maintenance, 2017.

[Rienstra and Hall 2002] Allan Rienstra and James Hall. Ultrasonic Detection. 3(2):1–4, 2002.

[Santiago 2019] A. Santiago. *Predictive Maintenance Mechanisms for Heating Equipment.* University of Aveiro, Portugal, 2019.

[Santos 2012] D. Santos. *Certificação Energética de Prensas Mecânicas.* University of Aveiro, Portugal, 2012.

[Seymour *et al.* 1997] Lynne Seymour, Peter J. Brockwell and Richard A. Davis. Introduction to Time Series and Forecasting., Vol. 92. 1997.

[Sharma 2019] Nitin Sharma. Getting Started with MQTT-Part 1. https://nitin-sharma.medium.com/getting-started-with-mqtt-part-1-a3c365e3a488, 2019.

[Shumway and Stoffer 2010] Robert Shumway and David Stoffer. Time Series Analysis and its Applications, Vol. 92. 2010.

[Silva 2019] A. Silva. *ThermoDroid: Android based solution for temperature physiological assessment.* University of Aveiro, 2019.

[SmartBear 2021] SmartBear. AMQP Testing. https://support.smartbear.com/-readyapi/docs/testing/amqp.html, 2021.

[Sommer *et al.* 2018] P. Sommer, F. Schellroth, M. Fischer and J. Schlechtendahl. Message-oriented Middleware for Industrial Production Systems. *IEEE International Conference on Automation Science and Engineering*, 2018-August:1217–1223, 2018.

[Suiu 2020] Alice Suiu. SPI (Serial Peripheral Interface). https://ocw.cs.pub.ro/courses/pm/lab/lab0xc0-6, 2020.

[Tallberg 2020] Sebastian Tallberg. A Comparison of Data Real-Time Stream Processing Pipelines. 2020.

[Tank 2021] P. Tank. Everything about Data Processing — Definition, Methods, Types Application, 2021.

[Tran Anh *et al.* 2018] D. Tran Anh, K. Dabrowski and K. Skrzypek. The predictive maintenance concept in the maintenance department of the "industry 4.0" production entreprise. *Foundations of Management*, **10**(1):283–292, 2018.

[Tulan 2015] Tulan. I2C(Inter Integrated Circuit)/TWI(Two Wire Interface). https://www.engineersgarage.com/electronic-projects/i2cinter-integrated-circuit-twitwo-wire-interface/, 2015.

[Zhu *et al.* 2013] Junda Zhu, Jae M. Yoon, David He, Yongzhi Qu and Eric Bechhoefer. Lubrication oil condition monitoring and remaining useful life prediction with particle filtering. *International Journal of Prognostics and Health Management*, 4(SPECIAL ISSUE 2), 2013.

Intentionally blank page.

Appendix A

Docker

Docker is a platform with several components that simplifies many of the laborious tasks that users need to do when developing and deploying an application. *Docker* enables users to separate their applications from their infrastructure in order to be able to deliver the application faster. Quoting [Nubera 2021], "(...) wraps a piece of software in a complete filesystem that contains everything needed to run: code, runtime, system tools, system libraries.", making the environment where the application runs the same, regardless of the location where it is run. The possibility of creating software that behaves in exactly the same way in spite of running on different hardware, configurations and operating system, is a feature that appeals to developers [Cayetano and Alonso 2017].

A.1 Docker Containers vs. Virtual Machines

Virtual Machines (VM) are another type of virtualization, however there are some significant differences between them. A host can run multiple containers at the same time since the containers are lightweight [Cayetano and Alonso 2017]. In general, a host can run more containers than VMs. This feature of the containers allows the user to separate each component of the application into several containers and then link them together using tools from the *Docker* platform, such as *Docker Compose*, which allows launching several containers from different images, connecting the containers using a specific network, mapping ports between the container and *Docker* host, mapping volumes, etc. In the case of the VM, all components are placed within a single VM.

Figure A.1 shows the main architectural differences between *Docker* and VMs. It can be seen that a VM has its guest operating system above the host operating system, which makes VM heavy. On the other hand, *Docker* containers share the host operating system, and that is one of the main reasons for being so lightweight. In the case of VMs, there is a great isolation in the host kernel, which means greater security compared to containers. A container can have some security risks as it shares the kernel host. As for the performance level, for all the reasons mentioned above, the docker containers has some advantages, for example, it has a much lower boot time, it does not need to allocate permanent resources to the containers and scaling up the containers is quite an easy task when compared to VMs.

The table A.1 presents a summary of the differences between *Docker* containers and VMs.

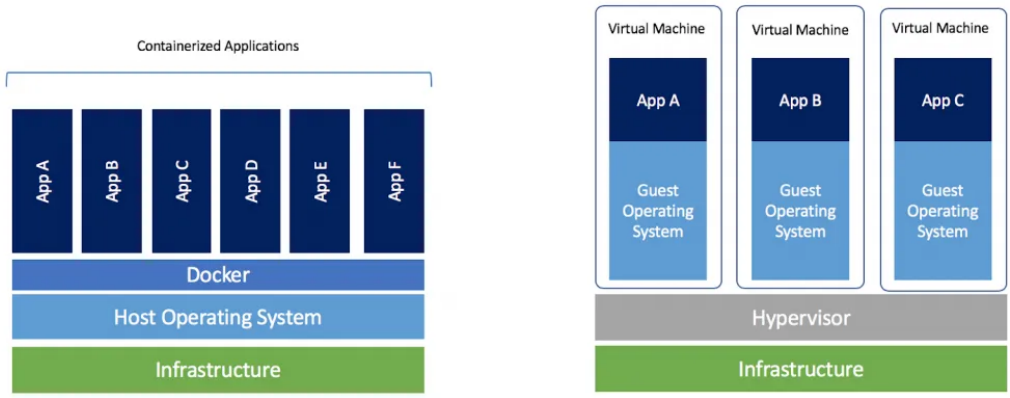


Figure A.1: *Docker* vs. VM architecture [Lessing 2020]

Table A.1: Differences between VMs and *Docker*

Virtual Machine	**Docker Container**
Each VM has a separate OS	All containers share the host OS
Boots in minutes	Boots in seconds
VM occupies a few GBs	Containers are lighweight (few MBs)
Hard to find ready-to-use VMs	Ready-to-use containers are frequent
VMs takes a considerable time to create	Containers can be created in seconds
More resource usage	Less resource usage

A.2 Docker Platform Components

A.2.1 Docker Engine

The main component of the *Docker* platform is the *Docker* Engine. This component consists of three distinct parts, a server, a REST API and a command line interface. The server, is a daemon process (program running for a long time doing some specific task) that controls the various types of *Docker* objects, such as containers, images, volumes, networks, etc. The REST API serves as a link between the client and the server. The command line interface allows the user to communicate with the server through the REST API using a set of *Docker* commands in the terminal [Cayetano and Alonso 2017].

Figure A.2 presents a schematic representation of the constituents of the *Docker* engine. There is a separation between the three components, this separation allows the user to decide where he wants to place the *Docker* server. Normally, the *Docker* client and *Docker* server are on the same machine, however there is the possibility to place the *Docker* server on other equipment and access it remotely.

A.2.2 Docker Objects

There are four main *Docker* objects, images, containers, networks and storage. An explanation of each of these objects will be given below [Arvind 2019].

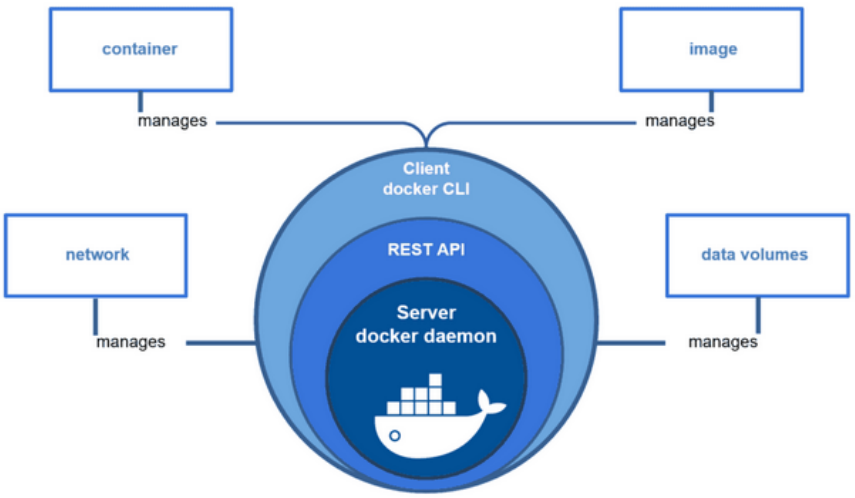


Figure A.2: *Docker* engine components [Kumar 2020]

Images The images are a read-only binary model that can create containers. They contain metadata that describes all the characteristics of the container. The images are used to store and send the developed applications. The images are stored in *Docker's* Registries. *Docker* registries are services where it is possible to store and download images.

Containers Containers, as explained earlier, function as isolated environments in which applications run. A container is created from an image, so all its features and functionality are defined in the image. However, there are some configurations that the user can provide, such as port mapping, volume mapping, setting environment variables, networks, etc.

Networks *Docker* networking is the way that isolated containers use to communicate. The main networks are bridge, host, overlay, macvlan, none. Bridge is the default network driver, by default all containers created use this network and can therefore communicate with each other. Host removes the network isolation between the container and the *Docker* host. Overlay connect multiple *Docker* daemons together. Macvlan allows the assignment of a MAC address to a container, making it appear that the container is a physical device on the network. Lastly, none disable all networking.

Storage By default, all files created in a container are stored in the container itself, this implies that when the container ceases to exist, the information is deleted. *Docker* offers two options for storing files on the host machine, volumes and bind mounts. Volumes are stored in a folder managed by *Docker*, bind mounts are stored in a folder anywhere on the host system.

Intentionally blank page.

Appendix B

Message Oriented Middleware

Message Oriented Middleware (MOM) is a concept that involves the transfer of data between applications using communication channels that carry units of information (messages) [Chappell 2004]. The concept of middleware is used since a layer is created between software application and system resources.

A MOM system creates an interface for sending messages between distributed services. An advantage of this interface is the possibility for the user to focus on the implementation of the task and use a MOM to communicate with other processes instead of implementing all the communication technology between processes [Sommer *et al.* 2018].

There are several types of MOM and the process for choosing the MOM that best suits the problem in question is not always simple. However, there are five features that make an objective differentiation between all types of MOM, namely: Communication patterns (CP), Payload types, Quality of service for message delivery, security and Implementation and services (I+S) [Sommer *et al.* 2018].

Below is an explanation of each of these features.

- **Communication patterns (CP)**

  There are two forms of message transfer, request/reply and publish/subscribe (Pub/Sub). Request/reply guides the classic communication of a client/server model, a service sends a request in the form of a message, the receiver reads the message and, if necessary, responds with another message. Publish/subscribe is designed to distribute message streams. In this case, publishers send messages to a specific channel (topic), creating a stream of messages. Subscribers subscribe to certain channels, in order to receive messages sent by publishers. In this case, it is irrelevant to the publisher whether there are any subscribers ready to receive the messages.

- **Payload types**

  Data sent via MOM is called a message payload. Before the payload is sent it is necessary to proceed with a serialization, as the payload is sent to the network as bits. Serialization can either be implemented by MOM or must be the responsibility of the user.

- **Quality of Service for message delivery (QoS)**

  The reception of sending messages is expressed as quality of service (QoS) and can be of three types: at least once, at most once and exactly once. At least once allows a message to be sent more than once to the consumer. At most once ensures that a message is never sent to the consumer more than once. The exactly once type, ensures that each message is sent exactly once to the consumer. In addition, there are also service possibilities for load balancing and clustering. A MOM can be executed in a cluster with several servers and the message load can be balanced between the different servers.

- **Security**

  There are three types of security supported by MOM, namely: encryption, authentication and authorization. Encryption ensures that no participant in other networks has access to messages. Authentication allows only equipment with proven identity to establish connection to the network. Finally, authorization allows the possibility to grant rights to certain resources only to certain clients.

- **Implementation and services (I+S)**

  A feature that differentiates the types of MOM is their type of implementation, they can have a free or paid implementation. Usually clients are created in different programming languages in order to facilitate the implementation of MOM, the more languages are available for each type of MOM, the more freedom the user has.

According to [Dobbelaere and Sheykh Esmaili 2017], the three most popular types of MOM are: *Advanced Message Queuing Protocol* (AMQP), *Message Queuing Telemetry Transport* (MQTT) and *Kafka*. Below is an explanation of the main characteristics of each of these types.

B.1 AMQP

The AMQP distinguishes between two types of layers, the functional layer and the transport layer. The transport layer defines the communication between the clients and the broker and the data encoding. The function layer defines the queue and exchange entities. The queue stores the messages in a specific sequence and forwards the messages to the respective consumers. Exchanges consume messages from producers and forward them to queues. Figure B.1 shows an example of publishing and subscribing messages using the AMQP protocol.

The standard implementation of AMQP does not present any type of security, however, its implementation with RabbitMQ supports the requirements of the industry [Sommer *et al.* 2018]. AMQP is widely used and has an active community and commercial support.

B.2 MQTT

MQTT was developed with the objective of allowing communication for devices with low computational capacity. Uses the Pub/Sub pattern with only one broker. The standard

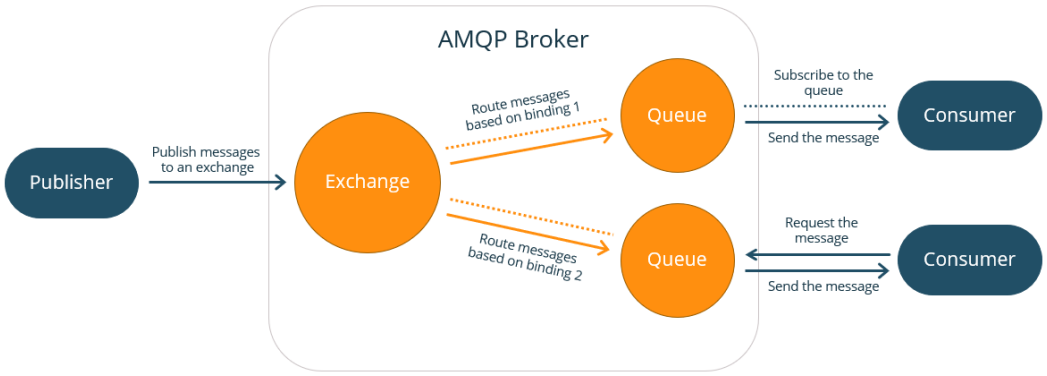


Figure B.1: Message transfer using AMQP [SmartBear 2021]

MQTT uses the TCP protocol as a method of transferring data between clients and the broker [Sommer *et al.* 2018].

The possibility of running an MQTT client on a device with low bandwidth network and with low computational resources is what makes this MOM quite suitable for IoT devices. An example of this suitability is shown in Figure B.2.

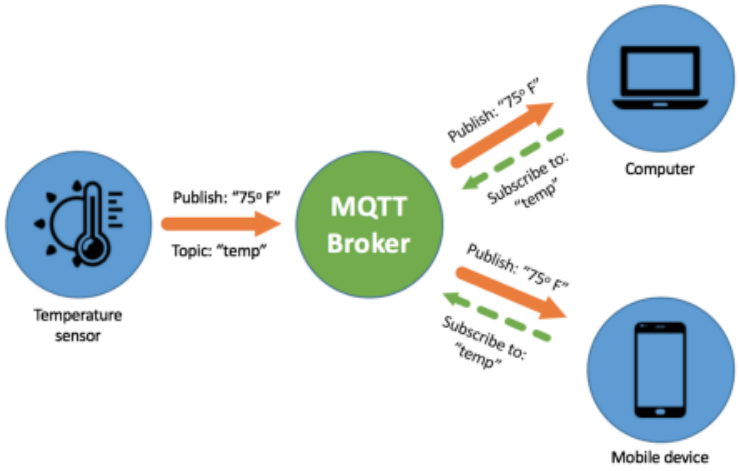


Figure B.2: MQTT in temperature sensing [Sharma 2019]

B.3 Apache Kafka

Apache Kafka is an open source message distribution system based on a broker. *Kafka* cluster typically consists of multiple brokers in order to maintain load balance [Tallberg 2020]. *Kafka* supports only the Pub/Sub messaging type.

For reasons of availability, parallelization and scalability, each topic is divided into partitions, and the partitions are spread across the cluster. The concept of partitions guarantees higher throughput.

The same partition can also be replicated in several brokers offering redundancy,

that is, if a broker goes down, the information is never lost [Silva 2019]. This feature is called *Replication*. However, when the same partition is present in more than one broker, there is always a *Leader* and one or more *Replicas*. The partition *Leader* is the one that is actually being consumed by clients, the *Replicas* are just partitions with the same information as the *Leader* however, in normal operation, they are not consumed. If, for some reason, the broker where the partition leader is housed goes down, there is an entity (*ZooKeeper*) that search for the closest *Replica* partition and promotes it to *Leader*. This way it is guaranteed that there is always a functioning partition.

A schematic of a *Kafka* cluster with two brokers is shown in Figure B.3.

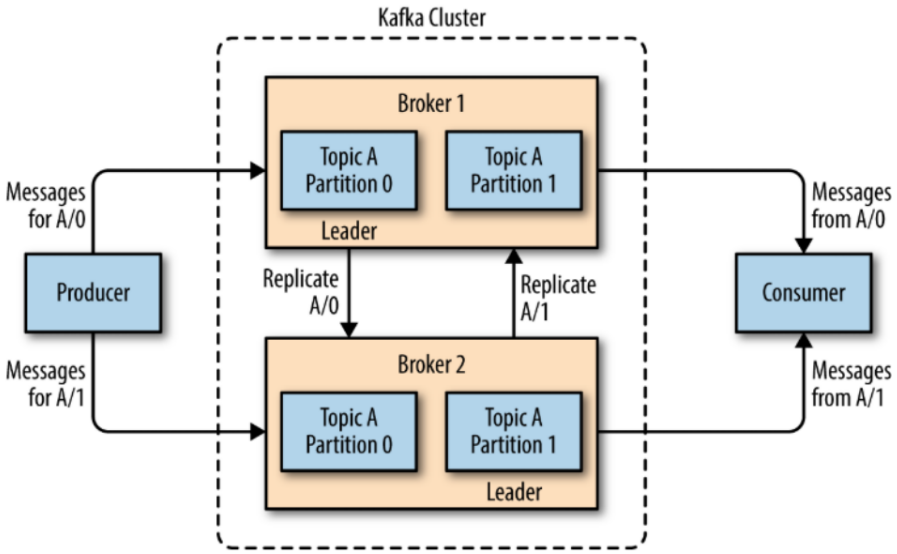


Figure B.3: *Kafka* cluster with two brokers [Narkhede *et al.* 2007]

One of the big differences from other MOM is that *Kafka* keeps each message on disk. Each partition is stored in multiple log files with fixed sizes, whenever the producer sends a new message, the broker appends the new log entry to a file. This message can be stored for a predefined period of time regardless of whether a consumer has received it or not. Each consumer has a reading offset, so the consumer can read all the information that is stored in the broker, and if the consumer is disabled for some period of time, the consumer can read the information again from the position where he was thanks to the offset. For this reason, *Kafka* can also be used as a database [Sommer *et al.* 2018].

There is a fundamental and absolutely necessary component to allow the operation of *Kafka* services, that component is *ZooKeeper*. *Zookeeper* works as a centralized service and is used to provide flexible and robust synchronization of distributed services, such as *Kafka*. There are four main functions performed by this component, namely [Cloudera 2019]:

- **Controller election**;

  It is responsible for maintaining the leader-follower relationship across all partitions. If a node goes down, it is ZooKeeper that informs other *Replicas* to function as partition *Leaders* in order to allow the partition to remain available.

- **Configuration of topics**;

  It is responsible for the configuration of all topics, number of partitions, location of *Replicas*, etc.

- **Access control lists (ACLs)**;

  Zookeeper maintains the ACLs for all topics, that is, it maintains a list of who is allowed to read/write on each topic.

- **Membership of the cluster**.

  Zookeeper also maintains a list of all active brokers at any given time.

B.4 Choice of the type of MOM used

Table B.1 was designed to summarize some of the differences between the three types of MOM addressed.

Table B.1: Differences between the three types of MOM, adapted from [Sommer *et al.* 2018]

	MQTT	**AMQP**	**Kafka**
CP: Messaging types	Pub/Sub	Pub/Sub Request/Reply Point-to-point	Pub/Sub
CP: Technical communication realization	Broker	Broker	Broker
QoS: Message delivery	At least once At most once Exactly once	Exactly once	At least once At most once Exactly once
Security: Authentication	SASL PLAIN	SASL with challenge response	SASL PLAIN and Kerberos
Security: Encryption	TSL	TSL	SSL
Security: Authorization	ACL	ACL	ACL

Based on the characteristics of the three types of MOM discussed above and using table B.1, it was considered that, for the realization of a system that achieves the established objectives, the protocol to be used is *Kafka*.

The characteristics that weighed more, in choosing *Kafka* in relation to the others, were the following:

- **High scalability**;

  Kafka is a distributed system, which allows it to be scaled easily without the need for downtime.

- **High durability**;

  Kafka stores the messages on the disk, making it a durable messaging system.

- **High reliability**;

  Kafka replicates the data in different brokers, ensuring that if for some reason one broker goes down, there is another one with the data.

- **High performance**.

  Kafka delivers high throughput in the process of sending and receiving messages, and presents constant performance levels, even when dealing with many terabytes of storage messages [Sommer *et al.* 2018].

Appendix C

JSON keys sent by Edge Devices

```
version: int
appSID: str
msgSID: str
msgDT: int
location: str
eventDT: int
eventSID: str
eventTP: int
gposSID: str
hdsc: str
flags: int
dim00: str
dim01: str
dim02: str
dim03: str
dim04: str
dim05: str
dim06: str
dom07: str
dim08: str
dim09: str
val00: float
val01: float
val02: float
val03: float
val04: float
val05: float
val06: float
val07: float
val08: float
val09: float
time_start: int
time_end: int
val00_wind: int
val00_num: int
val00_sum: float
val00_min: float
val00_max: float
val00_norm: float
val00_mean: float
val00_std: float
val00_q2: float
val00_uout3z: float
val00_lout3z: float
val00_cdiff: float
val00_pdiff: float
val00_sadiff: float
val00_sddiff: float
cmd: str
result: str
errorcode: int
nexttopic: str
store: str
dataTP: str
dataPR: str
dataCL: int
```

Figure C.1: Necessary JSON keys sent by Edge Devices

Intentionally blank page.

Appendix D

Programming the I2C protocol

According to the I2C protocol, the sending and receiving of bytes by the *Master* follows a set of rules.

Regarding the sending of a byte by the *Master*, the set of necessary steps to be fulfilled is illustrated in Figure D.1.

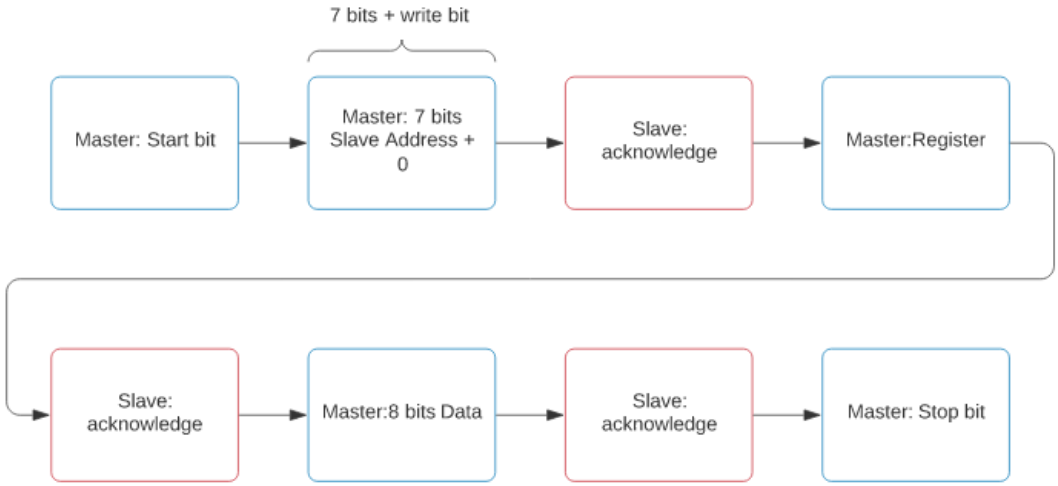


Figure D.1: Steps to send a byte

Initially, the *Master* has to send the start bit, then send the seven bits corresponding to the *Slave's* I2C address, followed by the eighth bit with the value 0 since it is intended to write. Then the *Slave* responds with acknowledgment. Upon receiving the acknowledgment, the *Master* sends the corresponding bits of the *Slave* register, that is, the memory position where the data will be written. The *Slave* responds with an acknowledgment, then the 8 bits are sent by the *Master*. At the end, the *Slave* sends the acknowledgment and the *Master* sends the stop bit, ending the process of sending a byte.

Regarding the process of receiving a byte from the *Master*, the process is slightly different, as shown in Figure D.2.

The differences between sending and receiving only start from the second acknowledgment by the *Slave*. Instead of the *Master* sending the eight data bits, it sends a restart

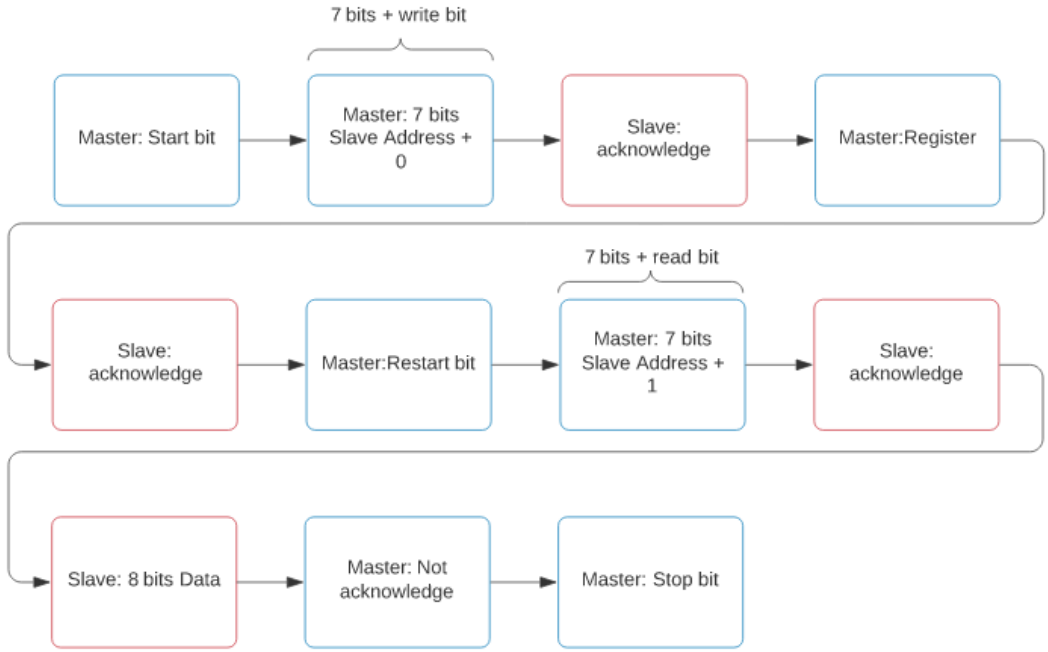


Figure D.2: Steps to read a byte

bit. Subsequently, the *Slave's* seven address bits are sent again, however this time, the eighth bit is a 1 since it is intended to read and not write. The *Slave* corresponds first with an acknowledgment and then sends the 8 bits of data that were in the memory position specified by the *Master*. Finally, the *Master* sends a non-acknowledgment bit and sends the stop bit, ending the process of receiving a byte.

In order to communicate between the *ESP32* and the MPU-9250 by I2C, it was necessary to import the *Wire* library.

As can be seen in Figure D.1 and Figure D.2, it was necessary to know the address of the MPU-9250 and the register that will be read. For this, it was necessary to consult the MPU-9250 sensor data sheet, `https://invensense.tdk.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf`.