GONÇALO JOSÉ DE SOUSA AZINHEIRA

AN AUTOMATION SYSTEM FOR PREDICTIVE MAINTENANCE OF ELECTRIC MACHINES APPLIED TO PUMPING SYSTEMS

UNIVERSIDADE DO ALGARVE
Instituto Superior de Engenharia
2024

GONÇALO JOSÉ DE SOUSA AZINHEIRA

AN AUTOMATION SYSTEM FOR PREDICTIVE MAINTENANCE OF ELECTRIC MACHINES APPLIED TO PUMPING SYSTEMS

Mestrado em Engenharia Eletrotécnica e de Computadores
Especialidade em Tecnologias de Informação e Telecomunicações

Trabalho efetuado sob a orientação de:
Professor Doutor Jorge Filipe Leal Costa Semião

UNIVERSIDADE DO ALGARVE
Instituto Superior de Engenharia
2024

AN AUTOMATION SYSTEM FOR PREDICTIVE MAINTENANCE OF ELECTRIC MACHINES APPLIED TO PUMPING SYSTEMS

Declaração de autoria de trabalho

Declaro ser o autor deste trabalho, que é original e inédito. Autores e trabalhos consultados estão devidamente citados no texto e constam da listagem de referências incluída.

I hereby declare to be the author of this work, which is original and unpublished. Authors and works consulted are properly cited in the text and included in the reference list.

(Gonçalo José de Sousa Azinheira)

Acknowledgements

This work was completed with the support of many, many people.

Chiefly among them are, of course, my supervisor, Dr. Jorge Semião, who went way above and beyond his duties as a supervisor. I am extremely grateful for all he's done and the graceful patience he has shown me.

Meriting all the praise that can be attributed to a non-supervisor, I'd like to especially thank Ricardo Veiga, whose experience, expertise, readiness and friendship helped me carry on through the work as it came, easy and hard.

André Pedro who was especially able and willing to put up with my ramblings and wild ideas, i am very grateful to him as well as to the rest of the laboratory mates I've had through the development of this thesis as well as all the professors in such labs who have always given me so much.

Sérgio Brito kindly lent me his master's thesis hardware and respective firmware, for which I am grateful.

Carlos Santos and Dr. Nelson Sousa, from the Mechanical Engineering Department, offered invaluable mechanical expertise and hands-on support.

António Costa, from the Electrical Engineering Department was a helpful assistant in lots of electrical procedures and in the moral support he always provided.

My partner in life, Beatriz Duarte, who has been with me for years, has helped me in ways that no one else could, for that my gratitude knows no bounds.

Abstract

This thesis presents the development and implementation of an automation system for predictive maintenance of electric machines in pumping systems. The work integrates Industry 4.0 technologies such as IoT devices, machine learning algorithms, and advanced sensor systems to enhance the reliability and efficiency of industrial operations. A complete test bench was developed, featuring an electric pump, water reservoir, automation and control board, energy meter, vibration sensors, pressure sensors, motorized valves, and temperature sensors, allowing automated test procedures including fault-injection behaviors through valve control. The research bridges the gap between theoretical models of predictive maintenance and their practical implementation in industrial environments, emphasizing the importance of automation and data-driven decision-making. The core achievements include the successful development and deployment of a system capable of real-time data acquisition, advanced vibration analysis, and fault prediction using machine learning. A detailed analysis was conducted to determine optimal data processing procedures prior to analysis, and automation-based sensors were integrated with electronic-based sensors in a unified predictive maintenance system. Machine learning algorithms demonstrated the feasibility of implementing predictive maintenance within standard automation pumping systems by successfully predicting faults induced in the test pump. The developed system not only reduces unexpected failures but also aligns with modern demands for sustainability and operational efficiency. By combining data acquisition, real-time

analysis, and predictive modeling, the research offers a comprehensive approach that can be adapted across various industries reliant on electric machines.

Keywords: Predictive Maintenance, Industry 4.0, Machine Learning, Automation Systems, Vibration Analysis, Electric Pumps, Industrial IoT

Resumo

Esta tese apresenta o desenvolvimento e implementação de um sistema de automação para manutenção preditiva de máquinas elétricas em sistemas de bombagem. O trabalho integra tecnologias da Indústria 4.0, como dispositivos IoT, algoritmos de aprendizagem automática e sistemas avançados de sensores para melhorar a fiabilidade e eficiência das operações industriais. Foi desenvolvida uma bancada de ensaios completa, incluindo uma bomba elétrica, reservatório de água, quadro de automação e controlo, contador de energia, sensores de vibração, sensores de pressão, válvulas motorizadas e sensores de temperatura, permitindo procedimentos de teste automatizados, incluindo comportamentos de injeção de falhas através do controlo das válvulas. A investigação estabelece uma ponte entre os modelos teóricos de manutenção preditiva e a sua implementação prática em ambientes industriais, enfatizando a importância da automação e da tomada de decisões baseada em dados. As principais concretizações incluem o desenvolvimento e implementação bem-sucedidos de um sistema capaz de aquisição de dados em tempo real, análise avançada de vibrações e previsão de falhas utilizando aprendizagem automática. Foi realizada uma análise detalhada para determinar os procedimentos ideais de processamento de dados antes da análise, e os sensores baseados em automação foram integrados com sensores eletrónicos num sistema unificado de manutenção preditiva. Os algoritmos de aprendizagem automática demonstraram a viabilidade de implementar manutenção preditiva em sistemas de bombagem com automação padrão, prevendo com sucesso falhas induzidas na bomba

de teste. O sistema desenvolvido não só reduz falhas inesperadas como também se alinha com as exigências modernas de sustentabilidade e eficiência operacional. Ao combinar aquisição de dados, análise em tempo real e modelação preditiva, a investigação oferece uma abordagem abrangente que pode ser adaptada a várias indústrias dependentes de máquinas elétricas.

Palavras chave: Manutenção Preditiva, Indústria 4.0, Aprendizagem Automática, Sistemas de Automação, Análise de Vibrações, Bombas Elétricas, IoT Industrial

Contents

List of Tables

List of Figures

Acronyms, Abbreviations and Symbols

AC	Alternating Current
ADC	Analog to Digital Converter
AI	Artificial Intelligence
API	Application Programming Interface
AP	Access Point
AWS	Amazon Web Services
AdaBoost	Adaptive Boosting
CM	Condition Monitoring
CPS	Cyber-Physical System
CSV	Comma-Separated Value
CPU	Central Processing Unit
DAQ	Data Acquisition
DC	Direct Current
DFT	Discrete Fourier Transform
DT	Decision Tree
DoM	Design-out Maintenance
E-Q	Efficiency-Flow Curve

FBD	Function Block Diagram
FFT	Fast Fourier Transform
FTP	File Tranfer Protocol
H-Q	Head-Flow Curve
HTTP	Hypertext Transfer Protocol
IEC	International Electrotechnical Commission
IIoT	Industrial Internet-of-Things
IoT	Internet-of-Things
JSON	JavaScript Object Notation
KNN	K-Nearest Neighbors
LAN	Local Area Network
LED	Light Emitting Diode
LR	Logistic Regression
LSVM	Linear Support Vector Machine
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
NPSH	Net Positive Suction Head
NVMe	Non-Volatile Memory Express
OPC UA	OPC Unified Architecture
OS	Operating System
P-Q	Power-Flow Curve
PCA	Principal Component Analysis
PLC	Programmable Logic Controller
PM	Preventive Maintenance
PdMaaS	Predictive Maintenance as a Service

PdM	Predictive Maintenance
RF	Random Forest
RMS	Root Mean Square
RM	Reactive Maintenance
ROI	Return of Investmen
RS485	Recommended Standard 485
RSC	Remote Service Call
RUSBoost	Random Under Sampling Boosting
SD	Secure Digital
SSH	Secure Shell
SVM	Support Vector Machine
TCP	Transmission Control Protocol
THD	Total Harmonic Distortion
TLS	Transport Layer Security
TWE	True Weighted Efficiency
VFD	Variable Frequency Drive
VNC	Virtual Network Computing
rpm	rotations per minute
η	Efficiency
ρ	Fluid Density

— Il sentiero per il Paradiso inizia all'Inferno.

Dante Alighieri

1 Introduction

The integration of next-generation digital technologies like Artificial Intelligence (AI), Internet-of-Things (IoT), and big data analytics has recently reshaped industrial processes, a movement known as Industry 4.0 [1] and Industrial Internet-of-Things (IIoT). These advancements have made systems smarter and more interconnected, allowing for highly efficient and flexible operations. Not only have they revolutionized production methods, making them more efficient, adaptable, and accurate, but they've also improved decision-making across industries using real-time, data-driven insights. Organizations can now respond to changing conditions or challenges faster and more effectively.

One of the standout benefits of Industry 4.0 is its impact on maintenance strategies. Predictive Maintenance (PdM) [2], in particular, has become a game-changer, enabling

companies to extend the life of their critical infrastructure while significantly reducing unplanned downtime and the high costs of reactive maintenance [3].

For electric machines, especially those in pumping systems, the importance of PdM cannot be overstated [4]. Pumping systems are essential in industrial processes, ensuring smooth operations. When one of these systems fails, it can disrupt production, leading to costly delays. This potential for disruption highlights the importance of a strong maintenance strategy. Luckily, Industry 4.0's data-rich environment offers a solution. PdM systems continuously monitor machine performance and health in real-time, identifying early signs of wear or issues that might otherwise go unnoticed until a failure occurs. By detecting these problems early, companies can replace or repair components before any unexpected downtimes happen. Machine Learning (ML) algorithms that analyze both historical and real-time data enable PdM systems to accurately forecast potential equipment malfunctions, extending the operational life of electric machines and increasing overall system reliability.

This thesis focuses on developing and implementing an automated system to improve PdM for electric machines used in pumping systems. By incorporating advanced sensors, AI-driven diagnostics, and ML decision-making processes, this system can significantly improve traditional maintenance practices. It aims not only to optimize machine performance but also to revolutionize how maintenance is managed on industrial floors. Through automation, real-time data processing, and predictive analysis, the system promises to enhance operational efficiency, reduce unplanned downtimes, and lower maintenance costs. Ultimately, this will lead to increased productivity and sustainability for industries relying heavily on electric-powered pumping systems [5].

1.1 Background and Motivation

With increasing global demand for energy efficiency and the critical need to decrease maintenance costs and time taken for downtime, there is an increasing drilling on the optimization of industrial systems [3]. Pumping systems are a part and parcel of many

such processes, ranging from water treatment and agriculture, to oil and gas and general manufacturing.

The basic philosophy of these systems rests on electric machines, which eventually degrade with time and may fail. Any unscheduled downtime in pumping systems may lead to significant operational disruption, financial losses, and even possible safety hazards. In this regard, the need arises for an efficient and automated system for forecasting and preventing a machine failure. The traditional maintenance strategies are based on either purely reactive or time-based approaches [6], which turn out to be either too little, too late, or result in a waste of labor and resources. Reactive maintenance allows an operational failure to occur and then shuts down the system to perform repairs. Time-based maintenance, on the other hand, often results in functional components being changed well before the end of their life. Either approach leads to a loss in operational efficiency and increases costs [7]. However, PdM has become one of the recent effective methods. PdM can be enabled through leveraging data from sensors, algorithms, and automation systems that allow early detection of faults before degradation develops into an expensive failure [8].

The main motivation of this thesis is to develop an advanced automation system for PdM in the case of electric machines applied in pumping systems. The research will therefore help in bridging the gap between the theoretical PdM models and their application to real industrial environments. This project will integrate state-of-the-art technologies, including IoT, ML algorithms, and analysis of real-time data, to develop a holistic system that could monitor the condition of electric pumping systems autonomously at all times.

And this is even more relevant in the emergence of Industry 4.0 programs within industries. As industrial systems become more integrated and built upon smart technologies, PdM systems become indispensable to attain operational efficiency, sustainability, and competitiveness [5]. It is here that this potential contribution could lie in discussing how smart automation might be integrated with PdM models to further reduce unscheduled downtime and extend the life cycle of electric machines while op-

timizing overall pumping system maintenance costs. This thesis is, therefore, driven by the industrial needs of cost reduction, effective execution, and improvement in the reliability of electric machines employed at pumping systems. In this framework, the development of an automation system for PdM is presented as a concrete answer to one of the highest industrial priority needs, which opens toward efficient, inexpensive, and sustainable methods of carrying out maintenance.

In doing PdM in electric machines, it is best performed using vibration sensors and vibration analysis. Electric motors and pumps are the central parts of any pumping system and, as such, will naturally emit or generate vibrations during operations. On the other hand, erroneous patterns usually denote or symbolize the early signs of several mechanical issues such as misalignment, bearing wear, unbalanced rotors, or shaft damage. In this case, continuous monitoring of these kinds of vibrations makes it possible to detect potential faults long before they can cause significant damage or any failure in the system [4].

The vibration sensors, usually accelerometers, velocity sensors, or piezzoelectric transducers, are fitted to the electric machine at selective points to capture actual data in real time. Such a sensor would measure amplitude and frequency of the vibration; such a signature would indicate the condition of the operating machine. Transmitted to an automation system, this information on patterns, analyzed by advanced algorithms, identifies abnormal vibration signatures. Therefore, maintenance teams can identify exactly what is wrong with the equipment and predict when high levels of failure points will likely occur with quite good accuracy.

The literature induces that Vibration Analysis has long been highlighted as a key tool for Condition Monitoring (CM) (a type of PdM that involves using sensors to measure the status of an asset over time while it is in operation), and if linked with automation and ML, forms the backbone of PdM strategies [9]. Vibration analysis involves studying the time-domain as well as a frequency-domain characteristic of vibration signals in this respect:

- The time-domain analysis will tend to measure over-all vibration levels and any

spikes or changes in amplitude outside established baselines.

- Frequency-domain analysis is performed by using several tools, including the Fast Fourier Transform (FFT); it breaks the vibration signals down according to their constituent frequencies. It does this to allow the identification of a vibration pattern corresponding to a specific fault (a bearing defect or imbalance, for example) [10].

In this respect, an automation system can reach an unprecedented level of detail in focused, continuous vibration analyses in real-time by combining data from several sensors. This allows the identification of very small variations in machine behavior, which would otherwise be really hard to observe, neither by simple observation nor with conventional monitoring approaches. This predictive capability avoids costly downtime associated with catastrophic failures, as well as unnecessary maintenance interventions.

Moreover, integration of vibration analysis into the overarching automation system enables the free interaction of various components within the PdM architecture. Vibration sensor data may be integrated with other inputs relative to temperature, pressure, and electrical current, among other parameters, thereby offering a general perspective, not only on machine health, but also in other parts of the system. Consequently, aggregated data will be analyzed through ML systems, fault detection enhanced, and predictive models honed to realize more and more accurate and reliable maintenance predictions. This technique is quite priceless when considering pumping systems, as most of them have extreme operating conditions. Other factors include fluctuating load conditions, temperature changes, and demanding operations that put extreme stress on electric machines. Early fault detection is of the essence in order to prevent operational disruptions [11, 12].

This is a non-invasive, low-cost vibration analysis test that allows for continuous monitoring [13] without interfering with the operation of the machine, thus further increasing efficiency and practicality of the PdM system. This makes integration of vibration sensors and vibration analysis within an automated PdM system a potentially

powerful approach that could be considered to improve electric machine reliability and performance at pumping systems. By troubleshooting mechanical problems early, organizations can radically reduce unexpected downtime risks while improving maintenance costs and prolonging the life of critical equipment. It will explore the deployment of vibration sensors in an automated environment, how real-time data and ML algorithms can work in a complementary fashion to achieve optimized maintenance schedules that improve overall system performance.

Regarding efficiency, efficient pumping system solutions are not new, which through installed controllers and sensors can automatically control the system, for example to regulate a constant temperature, or a constant water pressure, reducing the energy required by regulating the operating speed of the pump [14]. In most applications, the energy consumption of pumps can be substantially reduced by regulating the pump speed [15]. Still, a complete solution of an efficient management system with capability to infer deviations in behaviour in relation to the expectable (identified by past data statistics), and use that knowledge to improve the efficiency and the management decisions, is yet to be available in the market. This is also an important motivation for this work, as previous research work demonstrated that a cost-effective solution of a vibration-based sensor can produce good results on identifying operational errors [16]. If this solution is used in a complete automation system, it allows the optimization and management of pumping systems and also the monitoring and preventive identification of errors.

Furthermore, there are several operational optimization errors that can be identified by vibration, as happens with cavitation [12]. Therefore, it is important that information about the status of the pumps, error prediction and system performance can be available to the system that controls the operation of the pumps and the optimization of their operation and the energy consumed, with everything being framed within the same system and not different systems. For example, in a situation of predictable error, it may be preferable to reduce system performance, penalizing one of the normal requirements such as constant water pressure, to prevent the system from breaking

down and causing greater damage (complete stoppage, or premature damage to the system). Moreover, it is very important that any PdM solution for pumping systems can be compatible with the common industrial uses. In this regard, the monitoring strategy and the ML analysis should be developed using an automation system, with Programmable Logic Controller (PLC), because it is the common practice in industrial applications, and especially in industrial pumping systems.

1.2 Research Objectives

The goals of this MSc thesis focus on enhancing the field of PdM for pumping systems, by developing a new state-of-the-art Automation System. The new system should leverage Industry 4.0 tools like IIoT devices and ML algorithms to enable real-time condition monitoring and early fault detection based on vibrational analysis. This will involve the following objectives:

- Review of Maintenance systems, especially focusing on and PdM systems, and including sensor types and techniques;
- Review of electric pumps;
- Review of Automation pumping systems;
- Review state-of-the-art ML and AI algorithms, focusing on anomaly detection;
- Review state-of-the-art hardware systems for PdM in Academia and Industry applications;
- Analyse different vibration data collection techniques, to investigate the best procedures on system monitoring;
- Build an automation control panel and electric pumping system, to allow fault-injection in laboratory;
- Develop a test set-up with the automation pumping system to allow automated data collection for different system behaviours;
- Implement a ML data analysis to identify algorithms to allow predictive error-detection;

- Implement a PdM system for automation, to allow predictive error-detection;
- Validate the system's ability to actively predictively errors and predict maintenance needs as a proof of concept.

By meeting these objectives, the research done in this thesis seeks to contribute to the development of more efficient, reliable, and intelligent maintenance systems for industrial applications.

1.3 Thesis Structure

This thesis is organized into seven chapters.

Chapter 1 provides an introduction to the topic, outlining the background, motivation, and research objectives.

Chapter 2 reviews relevant literature, discussing electric pumping systems, their components, and problems associated with centrifugal pumps, as well as various approaches to maintenance and automation systems, with a focus on Predictive Maintenance and the use of data acquisition and Machine Learning.

Chapter 3 details the preliminary data acquisition and analysis, including the hydraulic system setup and the initial analysis performed on collected data.

Chapter 4 describes the data acquisition process within the automation system, focusing on the hardware used and the methods of gathering the necessary data.

Chapter 5 covers the data analysis, including classification techniques and testing results, to evaluate the system's performance.

Chapter 6 presents the experimental results obtained from the implementation of the system.

Finally, Chapter 7 concludes the thesis with an analysis of the work completed and suggestions for future research.

— Mit dem Wissen wächst der Zweifel.

Johann Wolfgang von Goethe

2

Literature Review

PdM has emerged as a transformative approach in industrial and mechanical systems, offering significant improvements in operational efficiency, reliability, and cost-effectiveness. Unlike traditional reactive and preventive maintenance strategies, PdM leverages real-time data and advanced analytics to anticipate and mitigate equipment failures before they occur. This approach is particularly critical for electric machines, which serve as the backbone of various industrial applications, including pumping systems that operate under demanding conditions.

This chapter presents the literature review for the knowledge surrounding PdM, with a focus on its application to electric machines used in pumping systems. It begins by exploring the fundamental concepts and principles underpinning PdM, followed by an analysis of the role of electric machines in industrial operations. Furthermore, the

chapter reviews state-of-the-art automation systems and their integration with PdM frameworks. Key challenges, limitations, and emerging trends in this field are also discussed to provide a comprehensive understanding of the research landscape.

By synthesizing insights from prior studies, this chapter establishes a foundation for the subsequent development of an innovative automation system for PdM. It highlights gaps in current methodologies and identifies opportunities to enhance the reliability and efficiency of electric machines in pumping systems.

2.1 Overview of Predictive Maintenance Techniques

2.1.1 Introduction to Maintenance and Predictive Maintenance

Maintenance is a group of procedures and strategies that range from inspection to repair, and are performed to keep equipment in good working order. Most authors in literature identify two big main types of interventions ([6]), reactive and proactive (Figure 2.1). Regarding the proactive strategies, they can be divided in preventive and PdM strategies.

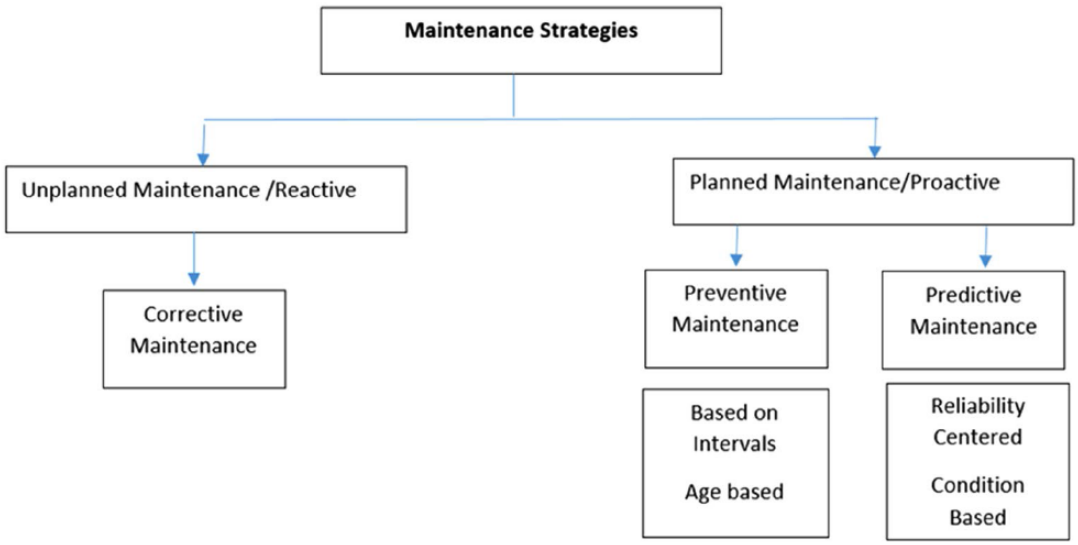


Figure 2.1: Hierarchy of maintenance strategies (from [6].)

In order to compare and understand the differences between these maintenance strategies, let us resume their main characteristics:

- Reactive Maintenance (RM) as a procedure is the repair or substitution performed after a failure has occurred, and is the most expensive type of maintenance [7]. As a strategy, it consists using equipment until it fails. Requiring no real *a priori* planning of maintenance, there's still a wide adoption of RM [17].

  Another problem of RM is that it can lead to a domino effect of failures. When a component fails, it can cause other components to fail, as is the case with excessive vibration, heat or when components break.

- Preventive Maintenance (PM), sometimes referred to as Periodic or Planned Maintenance, is a proactive strategy performed at regular intervals. It comprises scheduled inspections and replacements to try and minimize the probability of failures or unexpected breakdown. It is less expensive than RM [7], but is, by definition, over-preventive, as it is performed at regular intervals, regardless of the condition of the equipment, which leads to unnecessary costs [18]. If efforts are made to mitigate this, reducing the frequency of PM, the risk of unexpected RM increases.

- Predictive Maintenance (PdM) is a proactive strategy performed based on the condition of the equipment, and is the least expensive type of maintenance, as it is purposely used to optimize the balance of costs between proactive and RM procedures (Figure 2.2). The condition of the equipment is evaluated using real-time data. This method makes it easier to implement timely interventions in order to stop failures before they happen, using as little resource mobilization as possible. PdM plays a key role in reducing the costs associated with over-preventive maintenance while, at the same time, avoiding unexpected RM [18].

Interestingly, other authors refer to a different strategy, the Design-out Maintenance (DoM) [19]. This DoM strategy has a different approach when compared with the previous, because it aims at eliminating or reducing the root causes of recurring maintenance issues through design improvements. It focuses on modifying or redesigning equipment, systems, or processes to eliminate inherent problems that cause

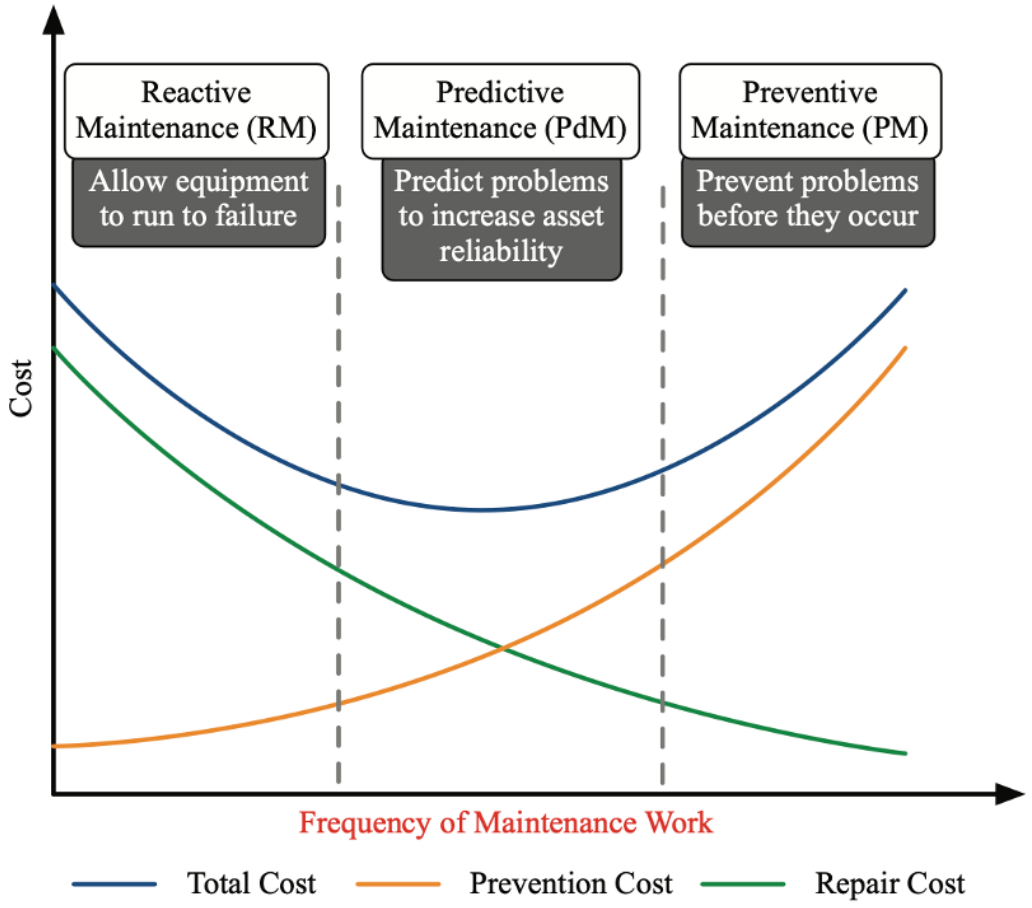


Figure 2.2: PdM as a prevention/repair cost equilibrium (from [18]).

failures, inefficiencies, or high maintenance costs. As it prevents future failures, it can be considered as a proactive strategy. However, as design improvements are chosen based on failure statistics, it can also be considered as a reactive strategy, or both proactive and reactive.

Another aspect that maintenance may impact directly is the efficiency of the equipment. As an example, While Johnson et al. [20] were measuring True Weighted Efficiency (TWE) (Equation 2.1) in 10 large centrifugal pumps over 13 months, one of the pumps (pump 9) underwent maintenance. After this procedure an increase to the pump's TWE of about 7.1% was measured: from 78.8% to 84.9%, which the authors calculated to represent 130–200 MWh yearly [20]. This clearly shows how much of an impact maintenance can have on the efficiency of a pump - and how much of an impact it can have on the energy bill of a facility.

TWE is the proportion of hydraulic energy output compared to total energy input, as given by:

$$TWE = \frac{\sum P_n t_n}{\sum \frac{P_n}{\eta} t_n} \tag{2.1}$$

where: P_n = power input;
t_n = time interval;
η = pump efficiency.

Regarding PdM, it represents a paradigm shift in equipment management, emphasizing the prevention of failures rather than reacting to them. It leverages data-driven approaches to predict when a machine is likely to fail, allowing timely interventions that prevent downtime and costly repairs. This approach contrasts with traditional maintenance strategies such as RM, where repairs occur after equipment failure, and PM, which follows a predetermined schedule without considering the actual condition of the machinery. The key objective of PdM is to maximize the uptime of equipment while minimizing maintenance costs. This is achieved by continuously monitoring equipment conditions through sensors and analysing the data to identify patterns indicative of wear, malfunction, or failure. By providing actionable insights, PdM enables businesses to make informed decisions about when and how to maintain their machinery, reducing unnecessary interventions and optimizing resource allocation.

In industrial settings, PdM is particularly critical for electric machines, which often serve as core components of complex systems. For instance, in pumping systems, the failure of an electric motor or related components can disrupt operations, leading to significant productivity losses. PdM offers a proactive solution by identifying early warning signs of issues such as misalignment, bearing wear, or electrical imbalances. As industries increasingly adopt automation and advanced analytics, PdM has become a cornerstone of Industry 4.0 initiatives. As it will be explained later on, the integration of technologies such as the Internet of Things (IoT), artificial intelligence (AI), and

machine learning has further enhanced the accuracy and reliability of PdM systems. These advancements are transforming the way organizations approach maintenance, shifting the focus from reactive measures to predictive and preventive strategies that ensure optimal performance and longevity of equipment.

2.1.2 Key Components of Predictive Maintenance Systems

PdM systems rely on a combination of advanced technologies and methodologies to monitor equipment performance, analyze data, and predict potential failures. These systems integrate hardware and software components designed to work cohesively, ensuring the accurate detection of anomalies and timely implementation of maintenance actions. These components are typically sensors, data analytics, real-time monitoring, and automation, and the effectiveness of PdM systems lies in the seamless integration of these components, because they work together to provide actionable insights, enabling organizations to transition from reactive to proactive maintenance practices. The continuous advancement of technologies such as IoT, edge computing, and AI further strengthens the capabilities of PdM systems, making them indispensable in modern industrial applications. The key components of PdM systems are briefly described below.

Sensors and Data Acquisition

Sensors form the backbone of PdM systems, providing the raw data necessary for monitoring and analysis. Various types of sensors are used based on the specific parameters to be monitored: Vibration Sensors, to Detect imbalances, misalignments, and bearing faults; Temperature Sensors, to monitor overheating of motors and other components; Current and Voltage Sensors, to measure electrical parameters to detect irregularities in electric machines; Ultrasound Sensors, to identify early signs of leaks or material fatigue; and Data Logging and Transmission, as it is necessary to collected data in real-time and transmit it to subsequent analysis via wired or wireless networks. Emerging technologies such as IoT enable seamless communication between sensors

and data storage systems, enhancing efficiency and accessibility.

Data Analysis and Algorithms

Once data is acquired, it is processed and analysed using advanced algorithms to identify patterns and predict failures. Typically, these algorithms include Signal Processing Techniques, to extract meaningful features from raw sensor data (for example, Fast Fourier Transform and wavelet analysis are commonly used in vibration analysis), and/or Machine Learning and AI Models, to detect correlations between equipment conditions and failure patterns (techniques such as supervised learning, unsupervised clustering, and anomaly detection are frequently applied).

Real-Time Monitoring and Alerts

Real-time monitoring systems allow operators to track the health of equipment continuously. These systems employ dashboards and visualization tools to present critical data in an accessible format, like Threshold-Based Alerts, where predetermined thresholds trigger alerts when monitored parameters exceed safe levels, or Predictive Notifications, where advanced systems predict the time remaining before a failure, enabling preemptive maintenance scheduling.

System Integration and Automation

Integration of PdM systems with broader industrial automation frameworks enhances their functionality. First of all, automation is normally present in the operation of industrial applications, and in this case, maintenance can be an added functionality to the main system. Moreover, one example of an automation framework is the use of SCADA Systems (Supervisory Control and Data Acquisition systems) to facilitate centralized monitoring and control of equipment. Another example is the use of Edge Computing, to process data at the edge device (near the source), which reduces latency and enhances system responsiveness. Another example with increased interest nowadays is Cloud Integration, where cloud platforms enable large-scale data storage and advanced analytics using AI tools. Regardless of the framework used, it is important to stress that the integration of maintenance procedures and tasks in the normal system

operation frameworks is very important, to facilitate proceadures and reduce costs.

2.1.3 Commonly Used Techniques in Predictive Maintenance

PdM employs a variety of techniques tailored to monitor and assess equipment health. These techniques focus on detecting early signs of wear or malfunction to prevent system failures. Each PdM technique offers unique advantages and is suited for specific types of equipment and conditions. The choice of the technique(s) to use depends on factors such as the nature of the machinery, operational environment, and the criticality of the system. Also, advanced PdM systems often integrate multiple techniques, combining their strengths to achieve comprehensive monitoring and fault detection capabilities.

Among the most effective techniques is vibration analysis, which relies on the principle that mechanical issues such as imbalance, misalignment, or bearing faults produce distinctive vibration patterns. This approach is particularly applicable to rotating machinery like electric motors and pumps, where it provides detailed diagnostic insights without interrupting machine operations. By identifying anomalies such as loose components or misaligned shafts, vibration analysis ensures the efficient functioning of equipment.

Thermal imaging and temperature monitoring represent another critical approach in PdM. Utilizing infrared cameras or temperature sensors, this technique identifies abnormal heat patterns. These anomalies often indicate problems such as overheating due to friction, electrical faults, or insufficient lubrication. Thermal analysis is especially useful in electric machines and pumping systems, monitoring motors, bearings, and electrical panels, where excessive heat can be a precursor to critical failures. This method offers a safe and non-contact means of detecting issues, allowing it to be applied in high-risk environments, especially for both mechanical and electrical systems.

Ultrasound analysis is another valuable technique for Predictive Maintenance (PdM) that captures high-frequency sound waves emitted by equipment during operation. It is particularly adept at identifying leaks in pipelines or valves and monitoring lubri-

cation levels in machinery. Also, it is versatile enough to be applied to both fluid and mechanical systems, allowing for the early detection of anomalies such as cavitation or bearing wear. This method is highly sensitive, capable of detecting subtle irregularities, and can be applied to both mechanical and fluid systems.

Electrical signature analysis (ESA) is another widely used method, focusing on monitoring electrical parameters such as current and voltage to detect issues like rotor defects, winding problems, or load imbalances. This non-invasive approach identifies mechanical or electrical faults, including rotor defects or load imbalances, offering a cost-effective solution for maintaining electric machines and pumping systems.

Oil analysis complements these techniques and is used extensively in heavy machinery. It complements other techniques by examining the composition of lubricants to detect contamination, metal particles, or signs of degradation. This method helps in diagnosing wear-related issues and ensuring proper lubrication practices, which are essential for extending equipment lifespan. By detecting problems at an early stage, oil analysis extends equipment longevity and ensures reliable performance.

Lastly, acoustic emission analysis detects sound waves generated by material deformation, cracks, or stress in machinery. This technique is particularly effective in monitoring the structural integrity of components such as turbines and pressure vessels. It allows for continuous monitoring during operation, providing an efficient means of detecting internal defects and preventing catastrophic failures.

Each of these techniques contributes uniquely to the PdM ecosystem, offering a range of diagnostic capabilities that cater to diverse industrial needs. By integrating multiple methods, organizations can achieve comprehensive monitoring and analysis, ensuring timely interventions and optimizing equipment performance. The effectiveness of these techniques lies in their ability to provide actionable insights, enabling maintenance teams to address issues proactively and reduce the risk of unexpected downtime.

2.1.4 Benefits and Challenges of Predictive Maintenance Techniques

As already stated, PdM represents a proactive strategy in equipment management, offering significant advantages in operational efficiency, cost-effectiveness, and risk mitigation. By predicting potential failures before they occur, PdM minimizes unplanned downtime, ensuring that machines remain operational when needed. This capability is particularly beneficial in industries requiring continuous operations, such as manufacturing and utilities, where unexpected equipment failures can lead to substantial losses. Moreover, PdM contributes to considerable cost savings by addressing issues before they escalate into major problems, optimizing maintenance schedules, and making efficient use of resources and personnel.

Another advantage lies in the ability of PdM to enhance the lifespan of equipment. Early detection of wear and tear prevents prolonged operation under suboptimal conditions, thereby extending the life of components and maintaining optimal performance. This proactive approach reduces overall stress on machinery and ensures its reliability over time. Safety is also a crucial benefit, as PdM helps prevent hazardous failures by identifying faults early, protecting both personnel and equipment, and reducing the likelihood of accidents caused by malfunctioning systems. Additionally, the data-driven nature of PdM provides actionable insights, enabling informed decision-making regarding maintenance priorities and resource allocation. Energy efficiency is further improved by detecting inefficiencies, such as motor misalignment or improper lubrication, which helps reduce unnecessary energy consumption.

Despite these compelling benefits, implementing PdM comes with its own set of challenges. One major obstacle is the high initial investment required, which includes costs for sensors, analytics software, and integration with existing systems. This financial burden can be particularly daunting for small and medium-sized enterprises, making it difficult to justify the upfront expenses. Another challenge lies in the complexity of integrating PdM systems with legacy equipment and industrial automation frameworks, which often requires significant technical expertise to ensure seamless communication between sensors, analysis platforms, and control systems.

Data management also poses a challenge, as continuous monitoring generates vast amounts of data that require robust storage and analysis capabilities. Extracting meaningful insights from this data often necessitates advanced analytics tools and expertise. Moreover, the effective implementation of PdM relies on skilled personnel who are capable of installing sensors, analysing data, and interpreting the results. Training employees to use these systems efficiently can be both costly and time-consuming. The accuracy of predictions is another concern, as the reliability of PdM systems depends on the quality of the data and the algorithms used. False positives or missed faults can lead to unnecessary maintenance or unexpected failures, undermining the effectiveness of the system.

Finally, resistance to change within organizations can hinder the adoption of PdM practices. Operators and managers may prefer traditional maintenance strategies due to their familiarity and perceived simplicity, slowing the transition to more advanced methods.

While PdM offers substantial benefits in terms of cost savings, enhanced safety, and operational efficiency, its implementation demands careful planning to overcome these challenges. Organizations must weigh the costs and complexities against the potential long-term value of adopting PdM systems. By addressing these obstacles effectively, businesses can unlock the full potential of PdM, driving sustainable growth and achieving improved system performance.

2.1.5 Emerging Trends and Future Directions

PdM is undergoing rapid evolution, propelled by technological advancements and the increasing demand for more efficient and reliable systems in industrial settings. The integration of cutting-edge technologies such as artificial intelligence (AI), the Internet of Things (IoT), and digital twins is reshaping the landscape of maintenance practices, making PdM systems more accurate, scalable, and accessible. These innovations are cementing the role of PdM in the Industry 4.0 era, where interconnected, intelligent systems drive operational efficiency.

One of the most transformative trends is the integration of IoT and edge computing. IoT devices enable real-time monitoring by connecting sensors and equipment across facilities, facilitating seamless data collection and communication. This interconnected network enhances the efficiency and responsiveness of PdM systems. Edge computing complements IoT by processing data closer to the source, thereby reducing latency and bandwidth requirements. This capability is particularly critical for time-sensitive applications where immediate responses are essential to prevent equipment failures.

Advancements in AI and machine learning have also significantly improved PdM capabilities. AI-powered algorithms analyse complex datasets to identify subtle patterns and anomalies, enhancing the precision of fault predictions. Machine learning models continuously refine their accuracy by adapting to new data, making the systems more reliable over time. Deep learning techniques, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), are further pushing the boundaries of what PdM can achieve. These methods are especially effective for analysing unstructured data, such as audio, images, and video from equipment monitoring systems, enabling advanced fault detection and diagnostics.

The emergence of digital twins is another groundbreaking development in PdM. Digital twins are virtual replicas of physical assets, simulating their behaviour and conditions in real-time. By integrating data from sensors, digital twins offer a dynamic and detailed view of equipment performance. They allow operators to conduct scenario analyses and test maintenance strategies virtually before implementing them, improving decision-making and optimizing schedules. Additionally, digital twins help identify structural and operational inefficiencies, contributing to better equipment design and operation.

PdM as a Service (PdMaaS) is an innovation that leverages cloud-based solutions to make advanced maintenance accessible and scalable. By reducing the need for on-premises infrastructure, PdMaaS enables smaller organizations to adopt PdM systems at a lower initial cost. Subscription models associated with PdMaaS provide flexible pricing options, broadening the appeal and accessibility of these solutions across di-

verse industries.

Sustainability and energy efficiency are increasingly becoming focal points in PdM. By optimizing equipment usage, reducing waste, and minimizing energy consumption, PdM aligns with global sustainability goals. It encourages the adoption of eco-friendly practices and materials, helping industries reduce their carbon footprints. Efficient maintenance schedules prevent unnecessary operations, further contributing to environmental conservation.

The integration of autonomous systems is paving the way for self-healing machines that can diagnose and repair minor issues without human intervention. Robotics is also being increasingly deployed in maintenance tasks, particularly in hazardous or hard-to-reach environments, enhancing safety and precision.

Looking forward, the future of PdM will be shaped by ongoing innovations and the evolving needs of industries. Standardization and interoperability will become essential to ensure compatibility across diverse systems and equipment. Enhanced cybersecurity measures will be critical as reliance on IoT and cloud platforms increases, protecting sensitive data from potential threats. As technologies become more affordable, PdM solutions will become accessible to industries in developing regions, fostering global adoption. Additionally, the collaboration between human operators and AI-driven tools will deepen, leveraging the unique strengths of both to optimize maintenance practices.

In resume, as IoT, AI, and digital twins continue to converge, PdM systems are becoming smarter, faster, and more adaptable. Trends like Predictive Maintenance as a Service (PdMaaS) and sustainability initiatives further expand their scope and relevance. These advancements position PdM as a cornerstone of future industrial operations, driving efficiency while aligning with environmental and economic objectives.

2.2 Electric Pumping Systems

At the heart of any electric pumping system is the electrical pump (and electric machine). Electric pumps are electromechanical devices that consume electrical energy to move fluids from one place to another. They can be found in a variety of industries, including agriculture [9], manufacturing, chemical [21], petroleum [22] and mining. Pumps are also widely used in residential and commercial buildings to move water and other fluids, in addition to municipal water and wastewater treatment plants [20].

The upstream part of the system, from where the liquid enters the pump is called the inlet, also called suction or intake. While the outlet, or discharge, is the downstream part of the system, relative to the pump.

There are several types of pumps, including centrifugal, positive displacement, and axial flow pumps. The choice of pump depends on the specific needs of the application, including flow rate, pressure requirements, and fluid properties. Centrifugal pumps operate by using a rotating impeller to transfer energy to the fluid, converting velocity into pressure. They are best suited for low-viscosity fluids and deliver a continuous flow. These pumps excel in high-flow, low-pressure applications but perform less effectively with high-viscosity fluids or at low flow rates. They are commonly used in applications such as water supply, irrigation, cooling systems, and chemical processing. Centrifugal pumps are simple in design, require low maintenance, and provide a smooth, steady output, though their performance is highly dependent on pressure and fluid viscosity.

Positive displacement pumps move fluids by trapping a fixed volume and forcing it through the system. They are categorized into reciprocating pumps (e.g., piston, diaphragm) and rotary pumps (e.g., gear, screw). These pumps are ideal for high-viscosity fluids and applications that require precise flow rates. Unlike centrifugal pumps, their flow rate is unaffected by changes in pressure, making them suitable for oil and gas transfer, food processing, and chemical injection. While they handle high-pressure applications and provide precise control, they have a more complex design and require higher maintenance. Additionally, they produce a pulsating flow, which

may need dampening for smoother output.

Axial flow pumps use a propeller or similar mechanism to move fluid in the same direction as the pump shaft (axially). These pumps are designed for high-flow, low-pressure applications and are often used in flood control, drainage systems, irrigation, and circulating water in power plants. They are highly efficient for moving large fluid volumes over short distances, providing smooth and steady flow. However, they are not suitable for high-viscosity fluids or scenarios requiring high pressure. Their efficiency also decreases when operating at low flow rates or high-pressure conditions.

Nevertheless, centrifugal pumps are the most common type of pump in industry [23]. As the pump used in this thesis is a centrifugal pump, let us focus only on this type of pumps. One example of a centrifugal pump can be seen in Figure 2.3.

Figure 2.3: A multistage vertical centrifugal pump (from `www.grundfos.com`).

2.2.1 Pump Parameters

There are a few parameters that can be used to describe the performance of a pump, including flow rate, head, power and efficiency.

- **Flow rate** is the volume of fluid that the pump can move in a given amount of time, and is typically measured litres per second (l/s), cubic metres per hour

(m^3/h), or, in imperial units, gallons per minute (gpm).

- **Head** is the height of column of fluid that the pump can lift, measured in metres m or, in imperial units, feet (ft). Head is defined as:

$$H = \frac{P}{\rho g} \tag{2.2}$$

where: H = head;
P = pressure;
ρ = density of the fluid;
g = acceleration of gravity.

According to (2.2), head is proportional to pressure. In any given pumping system g is constant and the fluid will often have a consistent density. In this common context, head is directly equated to pressure through a constant value ($\frac{1}{\rho g}$). Because of this relationship, head can also be used to describe the pressure in a pumping system.

- **Power** is the amount of energy that the pump consumes in a given amount of time, and is typically measured in kilowatts (kW) or imperial horsepower (hp). Input power is the electrical power consumed by the pump, and output power is the mechanical power delivered to the fluid.

- **Efficiency**, η, is the ratio of the mechanical power output of the pump to the electrical power input, and is typically expressed as a percentage.

- **Net Positive Suction Head (NPSH)** is the amount of pressure that the fluid exerts at the pump suction above the vapour pressure of the fluid. As it is a type of head, it is measured in the same units.

The relationship between some of these parameters is described by curves, which are graphical representations of the performance of a pump.

- The **Head-Flow Curve (H-Q)** describes pump-produced head as a function of the flow rate. The H-Q curve is used to determine the operating point of the pump.

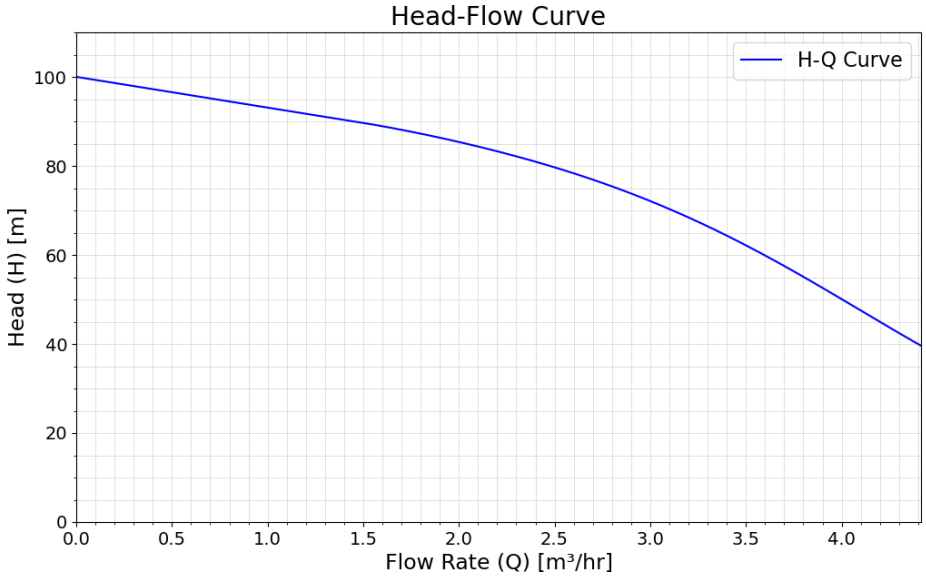


Figure 2.4: An example Head-Flow Curve (H-Q) curve of a centrifugal pump (adapted from `www.grundfos.com`).

- The **Efficiency-Flow Curve (E-Q)** describes pump efficiency as a function of flow-rate. The E-Q is used to determine the efficiency of the pump at different flow rates.

- The **Power-Flow Curve (P-Q)** is a graph of the power consumed by the pump as a function of the flow rate. The power-flow curve is used to determine the power consumption of the pump at different flow rates.

2.2.2 Components of a Centrifugal Pump

Because they're the most used pumps and of their general ubiquity, this work is centred around **centrifugal pumps**. These can be divided into three main components: the impeller, the casing, and the motor.

- The **impeller** is the rotating component of the pump responsible for moving the fluid. There are single-stage and multi-stage impellers, with the latter being used in applications where a high head is required [24].

- The **casing** is the stationary component of the pump, which also includes the stator of the motor. The pump casing is internally designed to direct the flow of fluid through the impeller and is the part of the pump that can be seen and externally accessed.

- The **motor** is the component of the pump that is responsible for converting the electrical power, providing the mechanical power to the impeller. There are pumps with single and three-phase motors. Below the electrical induction motor is better described.

The Induction Motor

The induction motor is a common type of Alternating Current (AC) three-phase motor used in pumps [25], as it is simple, relatively reliable, and cost-effective. It's mainly composed of a stationary and a rotating part: the outer stator and the inner rotor, respectively. There are two types of induction motors: squirrel-cage and wound-rotor.

Variable Frequency Drive (VFD)

A Variable Frequency Drive (VFD) is an extremely common type of motor controller that drives an electric motor by varying the frequency and voltage supplied to the motor. This allows the motor to operate at different speeds, which can be useful in applications where the flow rate of the pump needs to be adjusted, as well as potentially increasing motor efficiency [26]. VFD can also be used to reduce the starting current of the motor, which can be useful in applications where the motor is started frequently, potentially extending motor life [26, 27].

2.2.3 Problems in Centrifugal Pumps

Centrifugal pumps are subject to faults, which can be either electrical, mechanical and hydraulic.

Electrical Faults

Electrical faults are directly tied to the motor of the pump. Faults in the motor can be caused by undervoltage and overvoltage, phase unbalance and poor electric power quality.

- Undervoltage can cause the motor to draw more current than it should, which can lead to overheating and insulation breakdown. It can also lead to mechanical damage [28]. It is also known that it is incredibly detrimental to pump efficiency [29].
- Overvoltage can also increase motor winding temperature, which cause insulation breakdown, which can lead to a short circuit [30].
- Phase unbalance can affect the pump creating vibrations and creating heat, which can lead to motor winding overheat and torque reduction [29].
- Poor electric power quality can cause the motor insulation to degrade and decrease its life [31].

Mechanical Faults

Mechanical faults include:

- misalignment, unbalance, or looseness in the motor's and/or pumps's rotor;
- bearing faults,

Hydraulic Faults

Hydraulic faults in pumping systems include:

- **Cavitation**, an important and concerning fault, which is the formation of bubbles in the fluid that violently collapse, which can cause damage to the impeller and the casing.
- **Dry running**, or the running of the pump without any liquid, which can cause the destruction of bearings and sealing rings [32].
- **Clogging**

Another hydraulic fault is the presence of air in the fluid, which can cause the pump to lose prime and stop pumping.

Yet another hydraulic problem that a pumping system can encounter is dry running.

A final hydraulic fault is pump clogging.

2.3 Automation for Pumping Systems

Automation has profoundly transformed the operation and maintenance of pumping systems, bringing enhanced efficiency, reliability, and scalability to a wide range of industrial applications. By integrating advanced technologies such as sensors, control systems, and data analytics, automated pumping systems significantly reduce the need for human intervention while optimizing performance. These systems are essential in industries like water management, oil and gas, industrial processing, and agriculture, where the operation of pumps is critical to ensuring productivity and operational stability.

Automated pumping systems are designed to monitor, control, and optimize operations in real-time. Equipped with sensors, they measure key parameters such as flow rate, pressure, temperature, and energy consumption, providing a comprehensive picture of the system's status. The data collected by these sensors is processed by control systems like programmable logic controllers PLC and distributed control systems (DCS), which dynamically adjust pump performance to meet operational requirements. Additionally, advanced analytics tools are integrated to identify inefficiencies or potential failures, allowing operators to address issues proactively and minimize downtime. These systems are often enhanced with IoT-enabled platforms that facilitate remote monitoring and management, enabling centralized control and oversight of pumping operations.

2.3.1 Programmable Logic Controller

A PLC is an industrial-grade computer designed to control and automate processes in various sectors, including manufacturing, energy, transportation, and more. Known for its reliability, flexibility, and durability, the PLC has become a vital component in industrial automation systems, where precise and consistent control is essential.

At the core of a PLC's functionality is its ability to monitor inputs, process data, and control outputs to manage machinery and processes effectively. The Central Processing Unit (CPU) is the brain of the system, executing the programmed instructions and facilitating communication between input and output devices. Inputs are gathered through modules that interface with sensors, switches, or other monitoring devices, while outputs control actuators, motors, or other machinery components. The system also includes memory for storing the control program and real-time operational data, and it operates using a stable power supply to ensure uninterrupted functionality. Modern PLCs often feature communication ports that enable integration with other PLCs, industrial networks, or remote devices using standardized communication protocols such as Ethernet/IP, Modbus, or Profibus.

PLCs are programmed using specialized software that supports a variety of programming languages standardized under IEC 61131-3. These include graphical languages like ladder logic, which resembles electrical relay circuits and is widely used for its simplicity, and function block diagrams, which represent functions and their connections visually. Text-based languages such as structured text allow for complex algorithm development, while sequential function charts enable step-by-step control processes. These programming options make PLCs adaptable to a broad range of applications and operational requirements.

The versatility of PLCs has led to their adoption across numerous industries. In manufacturing, they control assembly lines, robotic systems, and conveyor belts. In process control, they manage operations in chemical plants, water treatment facilities, and oil refineries. In energy systems, PLCs monitor and control power generation and distribution, while in transportation, they automate traffic signals, railway systems,

and airport baggage handling processes.

PLCs offer several significant advantages, making them indispensable in industrial environments. Their design ensures durability under harsh conditions, including exposure to extreme temperatures, dust, and vibration. They are highly reliable, ensuring consistent operation with minimal downtime, and their modular structure allows for easy scalability to meet growing operational demands. PLCs are also reprogrammable, making them flexible for adapting to new or changing tasks. Their compatibility with various communication protocols ensures seamless integration with other systems, enhancing overall efficiency.

Despite these advantages, the implementation of PLCs is not without challenges. High initial costs can be a barrier for smaller organizations, and the programming and maintenance of these systems require skilled personnel. As industrial systems become increasingly connected, cybersecurity concerns also pose a growing challenge for PLC systems. However, advancements in technology are addressing these limitations. PLCs are evolving to incorporate IoT capabilities, edge computing, and artificial intelligence, enabling them to process data locally, make intelligent decisions, and connect seamlessly with cloud-based systems for advanced monitoring and analytics.

The PLC remains a cornerstone of industrial automation, providing robust, scalable, and efficient control solutions across diverse applications. As technology continues to advance, PLCs are poised to play an even more integral role in the development of smart factories and the future of industrial operations.

2.3.2 Applications, Advantages and Challenges

The applications of automation in pumping systems span multiple industries. In water management, automated systems are crucial for regulating water distribution and wastewater treatment, ensuring efficient resource utilization while meeting environmental compliance standards. In the oil and gas sector, these systems play a vital role in transporting fluids and supporting drilling operations, where precision and reliability are paramount. Industrial processing facilities depend on automated pumping

systems to manage the flow of chemicals, maintain consistent production quality, and optimize energy use. Agriculture has also benefited significantly from automation, particularly in irrigation systems, where pumps deliver water based on real-time soil and weather conditions to enhance efficiency and conserve resources.

The benefits of automation in pumping systems are substantial. Efficiency is greatly improved as systems respond dynamically to changing demands, reducing energy waste and ensuring consistent performance. Reliability is enhanced through continuous monitoring, which enables the early detection of issues and reduces the risk of unexpected failures. This proactive approach minimizes maintenance costs and helps maintain uninterrupted operations. Safety is also significantly improved, as automated systems reduce the need for manual intervention in hazardous environments, protecting workers and reducing accidents. Furthermore, automation systems are scalable, making them suitable for industries experiencing growth or fluctuating operational demands.

Despite these advantages, the implementation of automation in pumping systems is not without challenges. High upfront costs for purchasing and integrating automation technologies can be a barrier, particularly for smaller organizations. The process of integrating these systems with existing infrastructure requires careful planning and technical expertise to ensure compatibility. The reliance on advanced technologies also necessitates a skilled workforce capable of managing and maintaining automated systems, which can pose a challenge in industries with limited access to specialized talent. As automated systems become increasingly interconnected, cybersecurity risks also emerge, requiring robust measures to protect against potential threats.

Emerging trends in pumping system automation are addressing many of these challenges while pushing the boundaries of what automation can achieve. Artificial intelligence is playing a key role by enhancing predictive analytics, enabling smarter and more accurate system management. The use of digital twins is gaining popularity, allowing operators to simulate and optimize pumping operations in real-time. Edge computing is reducing latency by processing data locally near the equipment, leading

to faster and more responsive decision-making. IoT-enabled technologies are expanding the capabilities of remote monitoring and control, while innovations in energy-efficient systems are helping to minimize the environmental impact of pumping operations.

Automation has fundamentally reshaped the way pumping systems are operated, offering unmatched improvements in efficiency, reliability, and scalability. While challenges such as high costs and the need for skilled personnel remain, advancements in technologies like AI, IoT, and digital twins are providing solutions that continue to expand the potential of automation in this field. As industries increasingly prioritize operational efficiency and sustainability, automation in pumping systems will undoubtedly play a pivotal role in driving innovation, productivity, and environmental stewardship.

2.3.3 The Phoenix Contact PLCnext

In this thesis, the use of a PLC is a key feature. Therefore, in this section we present a brief description and characteristics for the PLC technology used, the Phoenix Contact PLCnext, as reported by the Phoenix Contact company [33].

The Phoenix Contact PLCnext (Figure 2.5) is a groundbreaking ecosystem that combines its PLC platform with other mdules like the community and the PLCnext Store. Represents a significant evolution in industrial automation. Designed to address the challenges of modern, interconnected industrial environments, the PLCnext platform combines the robust, real-time control capabilities of traditional PLCs with the flexibility and openness of modern IT technologies. This convergence allows for seamless integration of automation tasks, IT systems, and cloud-based applications, making it a key enabler for Industry 4.0 and smart factory initiatives.

One of the defining features of the PLCnext platform is its open and modular architecture. Unlike traditional PLCs, which often require specialized programming environments and proprietary tools, PLCnext supports a wide range of programming languages and environments. It of course includes the traditional International Elec-

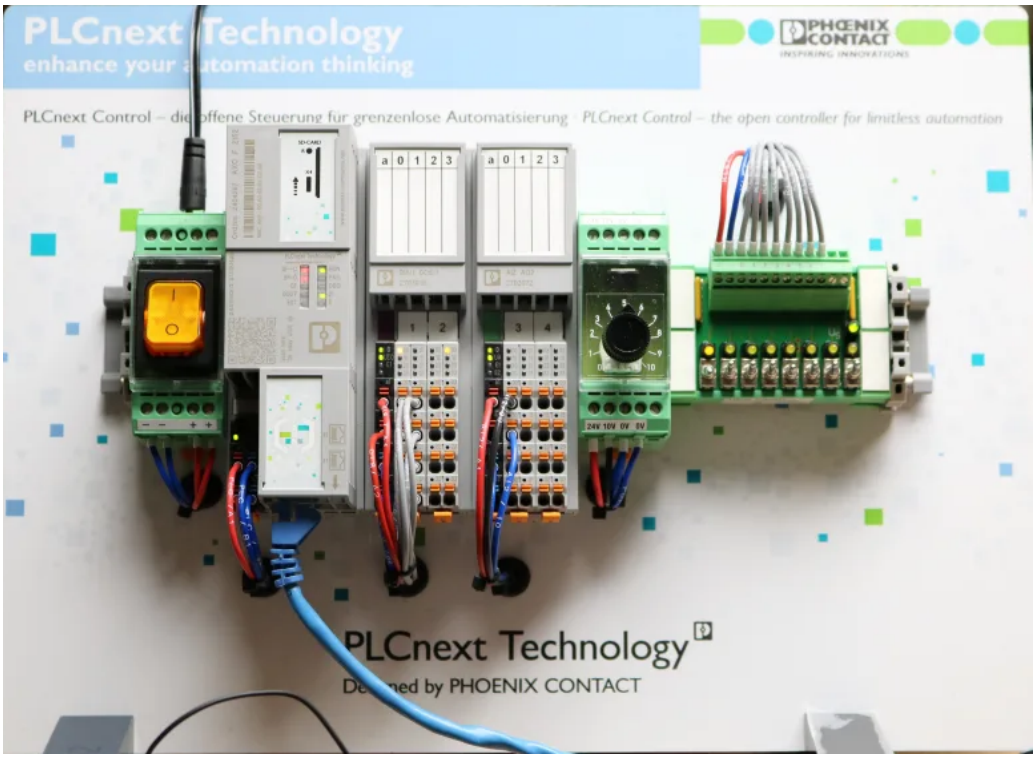


Figure 2.5: Picture of a PLCnext kit from Phoenix Contact, whose PLC is like the one used in this thesis.

trotechnical Commission (IEC) 61131-3 standards, but also high-level languages like C++ and C#, and even Python, to some degree. This flexibility enables engineers from diverse disciplines to collaborate on automation projects, reducing development time and enhancing efficiency. Furthermore, PLCnext provides integration with open-source software and IT standards, allowing users to leverage existing libraries and tools, significantly expanding the platform's functionality.

The PLCnext Control — the hardware — is built on a Linux-based operating system, providing an open and secure foundation for industrial applications. This allows developers to combine deterministic real-time control tasks with non-deterministic IT processes within a single system. For instance, users can run real-time machine control alongside data analytics or cloud-based monitoring applications without compromising performance. This capability bridges the gap between traditional automation systems and emerging IoT technologies, enabling smarter and more adaptive industrial processes.

Connectivity is another key strength of the Phoenix Contact PLCnext. The platform

is designed to facilitate seamless communication across devices, systems, and cloud platforms. It supports standard industrial communication protocols such as OPC Unified Architecture (OPC UA), Modbus, and PROFINET, ensuring compatibility with a wide range of existing systems. Additionally, PLCnext provides direct integration with cloud services, including the Phoenix Contact Proficloud and third-party platforms such as Microsoft Azure and Amazon Web Services (AWS). This cloud connectivity enables advanced features like remote monitoring, PdM, and data-driven decision-making, empowering businesses to optimize their operations.

One of the standout features of the PLCnext platform is the PLCnext Store, which offers a marketplace for downloadable applications and software components, many of them free of charge. This unique ecosystem allows users to access pre-built solutions for specific tasks, further accelerating development and deployment. Whether it's integrating machine learning models or implementing advanced diagnostics, the PLCnext Store provides users with the tools they need to expand the functionality of their systems.

Security is a top priority for the PLCnext platform, reflecting the increasing importance of protecting industrial systems from cyber threats. The system is equipped with robust security measures, including encrypted communication, secure boot processes, and user authentication, to ensure the integrity and confidentiality of operations. The open nature of the platform also supports ongoing updates and patches, keeping systems secure in the face of evolving threats.

The Phoenix Contact PLCnext stands out as a future-proof automation solution, bridging the gap between traditional industrial control systems and modern IT technologies. Its openness, modularity, and integration capabilities make it ideal for applications ranging from machine automation to smart grid management and beyond. By combining real-time control with advanced connectivity and flexibility, PLCnext empowers businesses to embrace digital transformation, paving the way for smarter, more efficient, and adaptive industrial processes.

2.4 Predictive Maintenance in other Works

Recent advances in PdM for pumping systems have demonstrated significant progress in integrating sensor technologies, sophisticated data analysis methods, and machine learning approaches to create more effective monitoring solutions. This section examines the current state of research across several interconnected domains that inform our approach.

2.4.1 Advanced Signal Processing and Fault Detection

Modern approaches to fault detection in pumping systems increasingly rely on sophisticated signal processing techniques applied to vibration data. Research has established precise correlations between specific mechanical and electrical faults and their vibrational signatures across different frequency ranges. Studies show that voltage quality issues manifest in distinct frequency patterns, with voltage unbalance producing oscillations at twice the supply frequency and harmonic distortion generating vibrations at six times the supply frequency [10].

Of particular importance for pump monitoring is the high-frequency range of 4-20 kHz, where cavitation-related vibrations typically manifest [11]. This understanding has led to the development of more targeted monitoring approaches that focus on specific frequency bands for different types of faults. Recent work has also shown that combining multiple analysis methods - including time-domain analysis, frequency-domain analysis, and wavelet transforms - can provide more robust fault detection capabilities.

The processing of vibration signals has evolved to address practical challenges such as noise reduction, feature extraction, and real-time processing requirements. Advanced preprocessing techniques, including adaptive filtering and envelope analysis, have proven effective in extracting meaningful features from raw vibration data. These developments in signal processing form the foundation for more sophisticated fault detection algorithms.

2.4.2 Machine Learning Architectures and Data Analysis

The integration of machine learning with traditional vibration analysis has revolutionized fault detection capabilities. Current research demonstrates a range of successful approaches, from relatively straightforward classification algorithms to complex probabilistic methods. Simple classifiers like K-Nearest Neighbors have achieved remarkable accuracy (up to 97.8%) in specific applications like cavitation detection [34], while more sophisticated approaches combining Bayesian networks with survival analysis have shown promise in predicting remaining useful life [11].

Recent studies have particularly focused on addressing the challenges of imbalanced datasets and the need for robust feature selection. Hybrid approaches that combine multiple machine learning algorithms have shown superior performance compared to single-algorithm solutions. For instance, ensemble methods that combine multiple classifiers have demonstrated improved reliability in fault detection across different operating conditions.

The selection of appropriate features for machine learning models has emerged as a critical research area. Studies have shown that carefully chosen features from both time and frequency domains, combined with domain knowledge about pump operation, can significantly improve model performance. This has led to the development of automated feature selection methods that can identify the most relevant characteristics for different types of faults.

2.4.3 System Architecture and Implementation

Modern PdM systems increasingly adopt distributed architectures that combine edge computing with cloud-based analytics. Research has demonstrated the effectiveness of non-intrusive monitoring systems using cost-effective sensors and IoT architectures [16]. These systems achieve continuous monitoring without requiring extensive modifications to existing equipment, making them particularly suitable for industrial retrofitting. Current architectures typically implement a multi-layer approach:

- Edge devices handle data acquisition and preliminary processing

- Local processing units perform initial analysis and feature extraction
- Cloud systems manage long-term data storage and complex analytics

This distributed approach addresses several practical challenges, including:

- Real-time processing requirements
- Network bandwidth limitations
- Data storage constraints
- System scalability needs

2.4.4 Industrial Integration and Practical Considerations

The implementation of PdM systems in industrial settings presents unique challenges that recent research has begun to address. Studies highlight the importance of system interpretability and reliability [35], demonstrating that maintenance engineers require systems whose decisions they can understand and trust. This has led to developments in explainable AI approaches that provide clear justifications for their predictions.

Industrial implementation considerations extend beyond technical performance to include:

- Integration with existing automation systems
- Compliance with industrial safety standards
- Cost-effectiveness and return on investment
- Training requirements for maintenance personnel
- System reliability and redundancy

Recent work has also focused on developing robust system architectures that can operate reliably in harsh industrial environments. This includes addressing challenges such as electromagnetic interference, variable operating conditions, and the need for minimal system downtime.

2.4.5 Emerging Trends and Future Directions

Current research points to several emerging trends in PdM:

- Integration of multiple sensor types for more comprehensive monitoring

- Development of self-learning systems that adapt to changing conditions
- Implementation of digital twins for improved system modeling
- Use of advanced visualization techniques for maintenance planning
- Integration with broader Industry 4.0 initiatives

These research developments collectively inform our approach to developing an automated PdM system. We build upon these foundations while introducing innovations in automation integration and real-time monitoring capabilities. Our work particularly focuses on creating a practical, industry-ready solution that combines the reliability of traditional automation systems with the advantages of modern PdM techniques.

2.5 Machine Learning Metrics and Procedures

Machine learning and artificial intelligence techniques require careful evaluation and preprocessing to ensure reliable results. This section discusses key metrics and procedures relevant to classification tasks in PdM systems.

2.5.1 Data Preprocessing Techniques

Data preprocessing is crucial for ensuring machine learning models perform optimally. Several standard techniques are commonly employed: Principal Component Analysis (PCA) is a dimensionality reduction technique that transforms high-dimensional data into a lower-dimensional space while preserving maximum variance. In vibration analysis, PCA can help identify the most significant frequency components contributing to system behaviour.

Standard scaling normalizes features by removing the mean and scaling to unit variance, ensuring all features contribute proportionally to the model. This is particularly important when dealing with features of different magnitudes, such as vibration amplitudes and electrical measurements.

2.5.2 Classification Performance Metrics

Several metrics are used to evaluate classification model performance [36]: Accuracy (Equation 2.3) measures the proportion of correct predictions among total predictions. While intuitive, accuracy alone can be misleading, especially with imbalanced classes:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Samples}} \tag{2.3}$$

Precision (Equation 2.4) quantifies the proportion of correct positive predictions, crucial for avoiding false alarms in maintenance systems:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2.4}$$

Recall (Equation 2.5), also known as sensitivity, measures the proportion of actual positive cases correctly identified. This is particularly important when false negatives must be minimized:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2.5}$$

Because both precision and recall are generally important, the F1-score (Equation 2.6) provides a balanced measure between precision and recall:

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.6}$$

2.5.3 Macro vs Micro Averaging

When dealing with multi-class classification problems, metrics can be averaged in different ways:

Macro-Averaging: Macro-averaging (Equation 2.7) calculates metrics for each class independently and then takes their unweighted mean. This treats all classes equally, regardless of their size:

$$\text{Macro-F1} = \frac{1}{n}\sum_{i=1}^{n}\text{F1}_i \tag{2.7}$$

where:

- n is the number of classes.
- F1_i is the F1-score for class i.

Micro-Averaging: Micro-averaging (Equation 2.8) aggregates the contributions of all classes to compute global metrics. This approach accounts for class imbalance by weighting classes according to their frequency in the dataset:

$$\text{Micro-F1} = \frac{2 \times \text{TPtotal}}{2 \times \text{TPtotal} + \text{FPtotal} + \text{FNtotal}} \tag{2.8}$$

For PdM applications, macro-averaging is often preferred as it ensures equal importance is given to all fault conditions, regardless of their frequency in the training data. However, in critical cases where certain fault conditions are more impactful, micro-averaging might be more appropriate to reflect operational priorities.

2.5.4 Cross-Validation

Cross-validation helps assess model performance and generalization capability. K-fold cross-validation partitions the data into k subsets, using k-1 subsets for training and one for validation, rotating through all possible combinations. This provides a much more robust evaluation of model performance than a single train-test split.

For imbalanced datasets, stratified k-fold cross-validation maintains the proportion of samples for each class in the training and validation splits, ensuring representative evaluation across all operating conditions.

2.6 Machine Learning Classification Algorithms

2.6.1 Linear Classifiers

Linear Support Vector Machine (LSVM) operates by finding an optimal hyperplane that separates different classes while maximizing the margin between them. The margin is defined as the distance between the hyperplane and the nearest data points from each class, known as support vectors. The optimization problem involves minimizing the norm of the weight vector while ensuring correct classification of training samples, with slack variables allowing for soft margins in non-perfectly separable cases [37]. In vibration-based fault detection, Linear Support Vector Machine (SVM)'s efficacy stems from its ability to handle high-dimensional feature spaces efficiently through the kernel trick, though only the linear kernel is used in this case. The algorithm's maximum margin principle provides robustness against noise in vibration measurements, while its linear decision boundaries align well with the often linearly separable patterns in frequency domain data.

Logistic Regression (LR) approaches classification by modeling the probability of class membership through the logistic function. For multi-class problems, this extends to the softmax function, providing a probability distribution over all possible classes. The model parameters are learned through maximum likelihood estimation, typically using optimization methods like gradient descent with regularization to prevent overfitting. The algorithm produces interpretable coefficients that directly relate to feature importance in the classification decision. For vibration analysis, these coefficients can identify which frequency components most strongly indicate specific fault conditions. The probabilistic nature of its outputs provides valuable uncertainty quantification for maintenance decision-making.

2.6.2 Tree-Based Methods

Decision Tree (DT) construct a hierarchical model where each internal node represents a binary decision based on a single feature threshold. The tree is built recursively, with

each split chosen to maximize information gain or minimize impurity measures. The construction process continues until stopping criteria are met, such as maximum depth or minimum samples per leaf node. For vibration signal classification, Decision Trees can capture non-linear relationships and handle multi-class problems naturally. The algorithm's splits effectively segment the frequency space into regions corresponding to different operational conditions. However, the orthogonal decision boundaries created by axis-parallel splits may not efficiently capture complex patterns in vibration data.

Random Forest (RF) is an ensemble learning method that constructs multiple decision trees using bootstrap aggregation (bagging) and random feature selection. Each tree is trained on a bootstrap sample of the original data, and at each candidate split, only a random subset of features is considered, typically the square root of the total feature count for classification tasks. This feature bagging, combined with sample bagging, ensures tree decorrelation and reduces variance in predictions. The algorithm provides robust feature importance measures through metrics like mean decrease in impurity, offering insights into the relevance of different frequency components for fault detection. The ensemble's majority voting mechanism provides more stable predictions than single decision trees.

2.6.3 Distance-Based and Probabilistic Methods

K-Nearest Neighbors (KNN) classifies samples based on a majority vote of the k nearest training examples in the feature space, typically using Euclidean distance. The algorithm is instance-based, making no assumptions about the underlying data distribution and deferring all computation until classification time. The choice of k represents a trade-off between decision boundary smoothness and local sensitivity. In vibration analysis, KNN's distance-based approach can be problematic when slight shifts in frequency components lead to large Euclidean distances despite representing similar conditions.

Gaussian Naïve Bayes models the likelihood of each feature as a Gaussian distri-

bution conditional on the class, assuming feature independence. The classifier applies Bayes' theorem with these likelihood estimates to predict class probabilities. While computationally efficient, the independence assumption rarely holds for frequency domain features where correlations between components are common in vibration data.

2.6.4 Non-linear Extensions

Polynomial SVM extends LSVM by implicitly mapping input features to a higher-dimensional space using a polynomial kernel function. This enables the algorithm to find non-linear decision boundaries in the original feature space while maintaining the maximum margin principle. The degree of the polynomial kernel determines the complexity of these boundaries. While more flexible than LSVM, the increased dimensionality of the feature space significantly impacts computational requirements and increases the risk of overfitting. For vibration data, this added complexity rarely justifies the marginal improvement in classification performance over linear approaches.

2.6.5 Ensemble Boosting Methods

Adaptive Boosting (AdaBoost) constructs a strong classifier by iteratively training weak learners, typically shallow decision trees, while adjusting sample weights to focus on previously misclassified examples. Each weak learner is assigned a weight based on its error rate, and final predictions combine these weighted predictions. The algorithm's adaptive nature allows it to focus on harder-to-classify samples, though this can lead to sensitivity to noise and outliers in vibration measurements.

Random Under Sampling Boosting (RUSBoost) modifies the AdaBoost algorithm to address class imbalance by randomly under-sampling the majority class before training each weak learner. This makes it particularly suitable for fault detection scenarios where normal operation samples typically outnumber fault conditions. The algorithm maintains AdaBoost's adaptive weighting scheme while ensuring balanced training sets for each weak learner, though the random under-sampling process may discard potentially informative majority class examples.

2.6.6 Algorithm Selection Considerations

For vibration-based fault detection, LSVM and LR emerge as particularly suitable choices, offering an effective balance between computational efficiency and classification performance. Their linear decision boundaries prove sufficient for most frequency domain classification tasks, while their efficient implementations enable real-time monitoring.

— Les arbres qui poussent lentement portent les meilleurs fruits.

Molière

3

Preliminary Data Acquisition and Analysis

3.1 Preliminary Data Acquisition

3.1.1 Hydraulic System

Work started with the assembly of the closed water circuit shown in Figure 3.1 as a controlled experimental setup. This test bed was composed of a three-phase centrifugal pump that was securely bolted to the floor to minimize unwanted vibrations, and a 500 l water reservoir that provided the necessary fluid capacity for our experiments. The two main components were separated by manual ball valves strategically positioned at the inlet and outlet of the pump, which could be manually closed or opened

to stress the pump under different operational conditions. Between the outlet of the pump and the outlet valve, a t-junction was fitted with a small valve, providing additional flexibility for future experimental modifications.

Figure 3.1: The hydraulic system, designed for the simplest possible real-world data acquisition.

The Pump

The pump used was a *Grundfos* CR3-8 A-A-A-E-HQQE — a vertical centrifugal 8 stage pump, rated for a flow rate of 3 m^3/h and a head of 38.3 m (see subsection 2.2.1), and can also be seen in 3.1. This type of pump was chosen for its simplicity and relative ubiquity. Its induction motor is rated for 400 V at 50 Hz Y connection, 750 W power output, and 2840-2870 rotations per minute (rpm). In Figure 3.2 it is possible to see the relationship between the speed at which the pump is working, described by the several curves, representing percentage of the working frequency, in relation to both head and flow. It is also possible to see that the speed is directly proportional to pump efficiency.

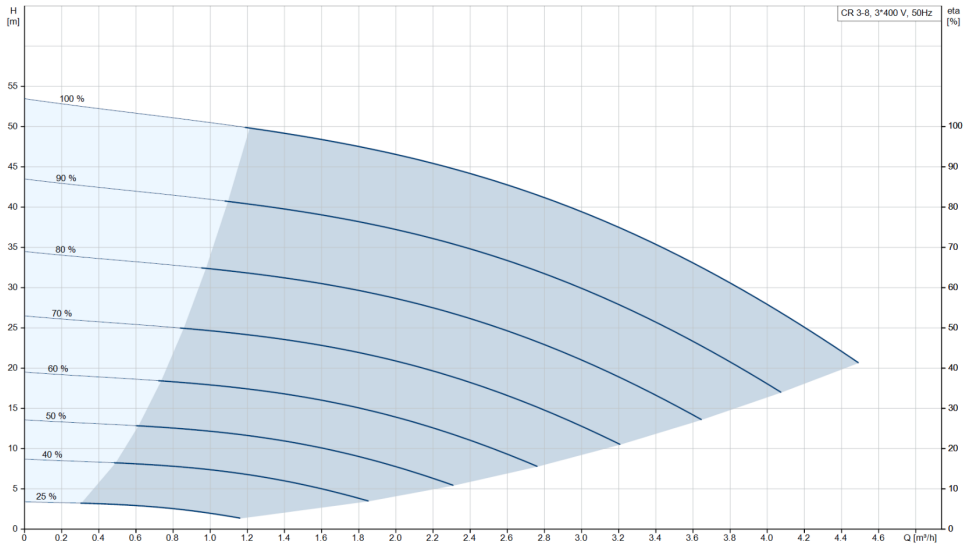


Figure 3.2: H-Q curves of the pump in question, also showing its efficiency for each percentage of the rated frequency (from `www.grundfos.com`).

Electric Panel

Initially, a simple electrical panel (Figure 3.3) was constructed for the simple on/off switching of the pump, using spare parts, but still having all the necessary electrical protection, and some indicator Light Emitting Diode (LED)s as to the state of the pump. Because of lack of space in the main body of the panel that was available another, smaller one, was used for manual control using the white button for turning the pump on through the use of a contactor and the red button to switch it off. According to the pump manufacturer, the starting current under these conditions is 580-620% of the nominal current.

Figure 3.3: The first electric panel, designed for simple pump state commutation.

3.1.2 Preliminary Data Acquisition and Classification

Using the test bed, the pump was run at steady state and the valve was changed to different closures, always turning the pump off again before repeating. In this case the outlet valve closed to different percentages and the water volume within the reservoir were varied as described:

- **Outlet Valve Closure:** 0%, 25%, 50%, 75%, 90%, and 100%.
- **Water Volume:** 95 l, 100 l, 125 l, 150 l, 175 l, 200 l, 225 l, 250 l, 275 l, 300 l, 350 l, 400 l, 450 l, 465 l, 495 l and 505 l.

As the outlet valve closes, pressure on the pump's discharge rises considerably. A too closed inlet or outlet is a very frequent cause of pump overheat [38], which is bound to reduce the pump's lifespan. As the water volume in the circuit is increased, the pressure in the intake of the pump rises, which is generally beneficial to the pump, as it helps deter the occurrence of cavitation by keeping the fluid at a pressure which is higher than the fluid's vapour pressure. The two lowest water volume levels are so low that when water enters the reservoir from pump discharge, it is directly hitting the reservoir pipe seal, from which the water goes to the pump, letting air mix with water, which is detrimental to the system, as it decreases pump efficiency, can cause corrosion, and can cause problems in filtering [39].

At each combination of outlet valve closure and water volume, the voltage created on the terminals of a piezoelectric element, which was attached to the pump, was measured during pump runtime, at steady state. This voltage is proportional to the acceleration of the vibration of the pump. These measurements were acquired using a piezoelectric sensor connected to a 12-bit, 1.245 MS/s multi-function DAQ computer card, the National Instruments PCI-6070E, for a precise and reliable data collection, controlled through a LabVIEW application. The piezoelectric sensor and the card were intermediated by a simple low-pass filter, cutting out frequencies over 8kHz, which was then connected to the card through an oscilloscope probe. This experimental design aimed to elucidate the relationship between the operational conditions of the pump and its vibrational behaviour.

Samples were taken at 24 kHz in 2 second bursts, ensuring an abundance of sample points. This experiment yielded a total 90-sample dataset, each comprised of a time-series of 48000 datapoints. These timeseries were then converted to the frequency domain through the FFT algorithm, which applies the Discrete Fourier Transform (DFT). Each of these 90 timeseries/spectra was classified in one of 3 classes according to their specific conditions, into faulty operation, "F", good working condition, "O", and a third, intermediate level, "I", distributed as shown in Table 3.2. A too closed output valve puts the pump into too much stress and a too empty reservoir allows for air to get inside the tubing because of the way the experimental rig was constructed, which is also detrimental for the pump.

Table 3.1: The initial classification used for each pair of valve closure/water volume inside the reservoir – Faulty (F), Intermediate (I) and Optimal (O).

Water	**Closure (%)**					
Volume (l)	**0**	**25**	**50**	**75**	**90**	**100**
95	F	F	F	F	F	F
100	F	F	F	F	F	F
125	I	I	I	I	I	F
150	O	O	O	I	I	F
175	O	O	O	O	I	F
200	O	O	O	O	I	F
225	O	O	O	O	I	F
250	O	O	O	O	I	F
275	O	O	O	O	I	F
300	O	O	O	O	I	F
350	O	O	O	O	I	F
400	O	O	O	O	I	F
465	O	O	O	O	I	F
495	O	O	O	O	I	F
505	O	O	O	O	I	F

Table 3.2: Class distribution in the preliminary data acquired.

Class	O	I	F	Total
Samples (spectra)	47	18	25	90

3.2 Preliminary Data Analysis

As a preliminary study, different data preprocessing methods were used, in order to empirically better understand the nature of this kind of pump vibrational frequency spectra.

As per the Nyquist sampling theorem, the voltage signal that was taken from the piezoelement, which is band-limited to 8kHz, can be digitized at a 24 kHz sampling rate without aliasing. The highest possible frequency component of such spectrum can be no higher than 12 kHz, half the sampling frequency. The spectrum's full range was, then, from 0 to 12 kHz in steps of 0.5 Hz. This makes it so each of the spectra is comprised of 24000 points, each a frequency and its magnitude. In Figure 3.4 is an example of what one looks like.

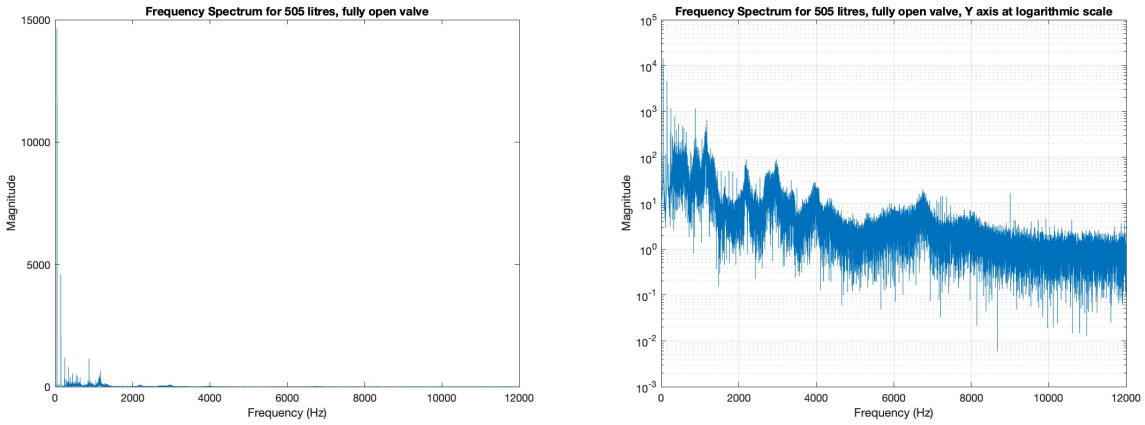


Figure 3.4: An example frequency spectrum – one of the original 90.

As figure 3.4 shows, the data is very dense, which can have quite an impact on a system that processes such data, both in storage capacity and computational power required. Additionally, sometimes peaks that are often present in most spectra can sometimes drift to a neighbouring frequency – such as the 250 Hz peak, the 5th harmonic, appearing at 250.5 Hz or in the 251 Hz. Because of these problems, the idea of grouping frequency in bands was explored — a process henceforth called banding. [16] used 1 kHz bands to this effect.

In banding, the average of the frequency values in a band is used and coupled with the energy level of a group of frequency points, the Root Mean Square (RMS) of their

magnitudes. Three types of banding methods were used, as described by Table 3.6:

- No banding at all;
- Equally sized bands — 8, 16, 32, 64 and 256 Hz —each beginning when the last one endeed;
- *Growing* method, an algorithmic method.

For the equally sized bands, the seemingly strange bandwidths were chosen in order to try and mitigate the problem of bands ending in harmonics of 50 Hz, the fundamental frequency. A band of 10 or 5 Hz, for example, would have several bands ending exactly in important harmonics of the fundamental; if in a sample a peak is captured at 502 instead of 500 it would fall in a different frequency band as compared to other samples. Different sizes were chosen in order to maximize the amount of insight that could be obtained about how the granularity affected the ability to infer faulty operation from the vibration data.

The *Growing* method is illustrated by figure 3.5. It ends the first band at 75 Hz and thee centres bands in the first 5 harmonics, as the 1st, 3rd and 5th are some the most frequent and highest magnitude peaks in the analysed spectra. As the fundamental is 50 Hz the first band was let to come from 1 Hz after, that being centred in harmonics of 50 Hz, means that each band is 50 Hz precisely. From the end of the 5th band on, to mitigate the fact that the drift of the peak from the harmonic is larger for larger frequencies, it creates bands obeying these 2 rules: the width of the band is approximately equal to 0.2 times the central frequency and the band starts at the frequency the last band ended.

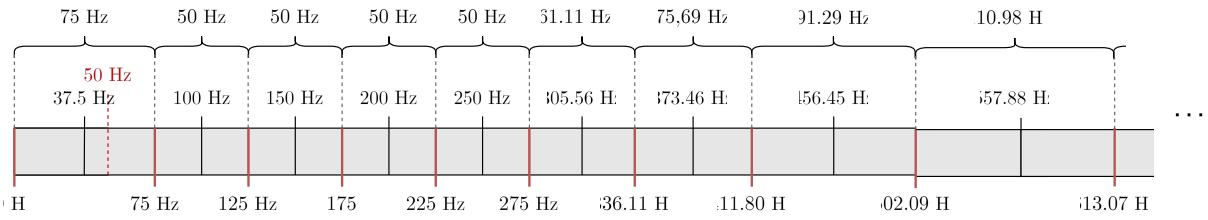


Figure 3.5: Illustrative example of the working of the *Growing* method of banding. All values are in Hz: the top row corresponds to the band width; the red number to the fundamental frequency, 50 Hz; the row below it to the bands' centre frequencies, and the bottom row to the start and cutoff frequencies.

It is also possible to see, in figure 3.4 just how massive the magnitude range is, that is, the difference between the highest and lowest magnitudes. Because of that, normalisation techniques were tested, namely logarithmic and linear normalisation, along with not normalising at all.

The resulting spectra, an illustrative example of which can be found in Figure 3.6, could then be arranged in a dataframe more than one way.

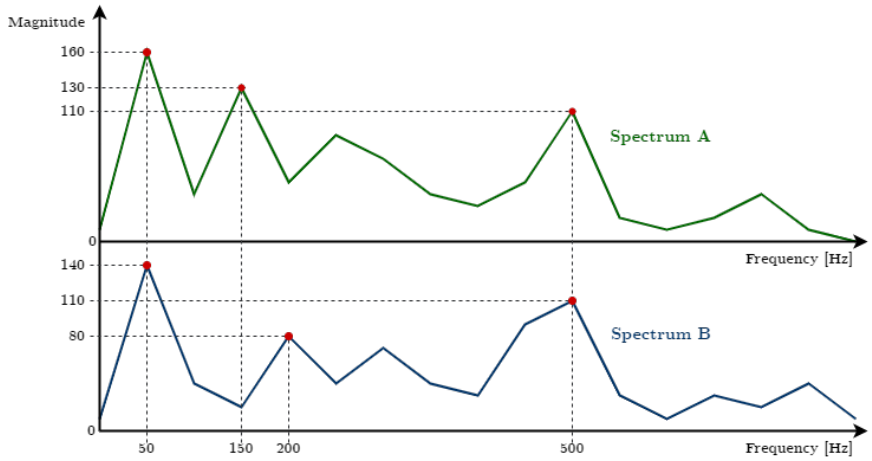


Figure 3.6: An illustrative example of the kind of spectra that could be had after banding. In red are highlighted the 3 highest peaks in each spectra.

A way of dataframing these peaks could be to have a column for each of the frequency components in the spectra — No Peaks; in 3.3 is the result of this dataframing method for the example in Figure 3.6.

Table 3.3: Result of the No Peaks dataframing method for Figure 3.6.

Frequency	**0**	**50**	**100**	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
Spectrum A	10	160	40	130	50	90	70	40	30	50	110	20	10	20	40	10	0
Spectrum B	10	140	40	20	80	40	70	40	30	90	110	30	10	30	20	40	10

Another way could be to use the n peaks with the highest magnitudes in each of the spectra. The n chosen was half the number of peaks in sample whose spectrum had the least number of peaks.

Using the highest peaks, the dataframe could be arranged using the frequency peaks found in all samples of the training dataset as a column — henceforth called

Sparse Table. In Table 3.4 is the result of using dataframing method for the example in Figure 3.6. In this method of dataframing there are quite a few peaks which are an important peak in a sample but aren't in another sample — in this second one the magnitude in the dataframe would be recorded as 0, creating a relatively sparse matrix.

Table 3.4: Result of the Sparse Table dataframing method for Figure 3.6, if using the 3 highest peaks.

Frequency	**50**	**150**	**200**	**500**
Spectrum A	160	130	0	110
Spectrum B	140	0	80	110

Another alternative for using the frequency peaks is by having double the columns to the frequency peaks, every 2 columns being the frequency and magnitude of a peak, ordered by the peaks' magnitudes, as shown in Table 3.5.

Table 3.5: Result of the Double Column dataframing method for Figure 3.6, if using the 3 highest peaks.

	1st Peak		**2nd Peak**		**3rd Peak**	
	Frequency	**Magnitude**	**Frequency**	**Magnitude**	**Frequency**	**Magnitude**
Spectrum A	50	160	150	130	500	110
Spectrum B	50	140	500	110	200	80

Table 3.6: Different possibilities for every data processing type

Type	**Details**
Banding	No banding 8 Hz 16 Hz 32 Hz 64 Hz 256 Hz Growing
Normalisation	Not normalised Linear Logarithmic
Dataframing	Double Columns Sparse Table No Peaks

Each of the 63 possible combinations of data processing possibilities shown in Table 3.6 was trained for classification as described by 3.1. Initially 9 classification algorithms were trained, using grid search to find the hyperparameters that produce the best model using that algorithm and getting a classification report with the results and metrics of the best model for each of the algorithms. The algorithms in which models were trained were:

- Random Forest;
- Linear SVM;
- Polynomial SVM;
- KNN;
- Logistic Regression;
- Decision Tree;
- Gaussian Naïve-Bayes;
- AdaBoost;
- RUSBoost.

The purpose of trying all combinations of these preprocessing methods and algorithms is to investigate the effectiveness of each method and algorithm in this kind of vibration data.

Because of the relative lack of samples and somewhat imbalance in data, stratified 5-fold cross validation was used and the Macro-F1 value was chosen as the main testing metric. In order to better investigate the effectiveness of each possibility in a relatively small, imbalanced dataset, each possibility was trained 5 times, each with a different seed for splitting the data in train and test sets, in order to get a better statistical grasp of the results.

The results of training the 5 splits of the 9 different models for each of the 63 preprocessing combinations can be seen in Figure 3.7, as boxplots. Each subplot is the distribution of the macro F1-scores calculated from the confusion matrices received from the inference of the 10-sample test set on each of the combination of possibilities for every other data processing type and algorithm. This means that, for example, the boxplot named linear in the normalization subplot represents the distribution of all the F1-score results for the testing of models that contained any algorithm, as well as dataframing and banding methods in which the normalisation was linear. Consequently, one third of all models' testing results are represented in this boxplot, given the fact that there were only 3 normalisation types under study.

In every boxplot in this document the whiskers represent the minimum or maxi-

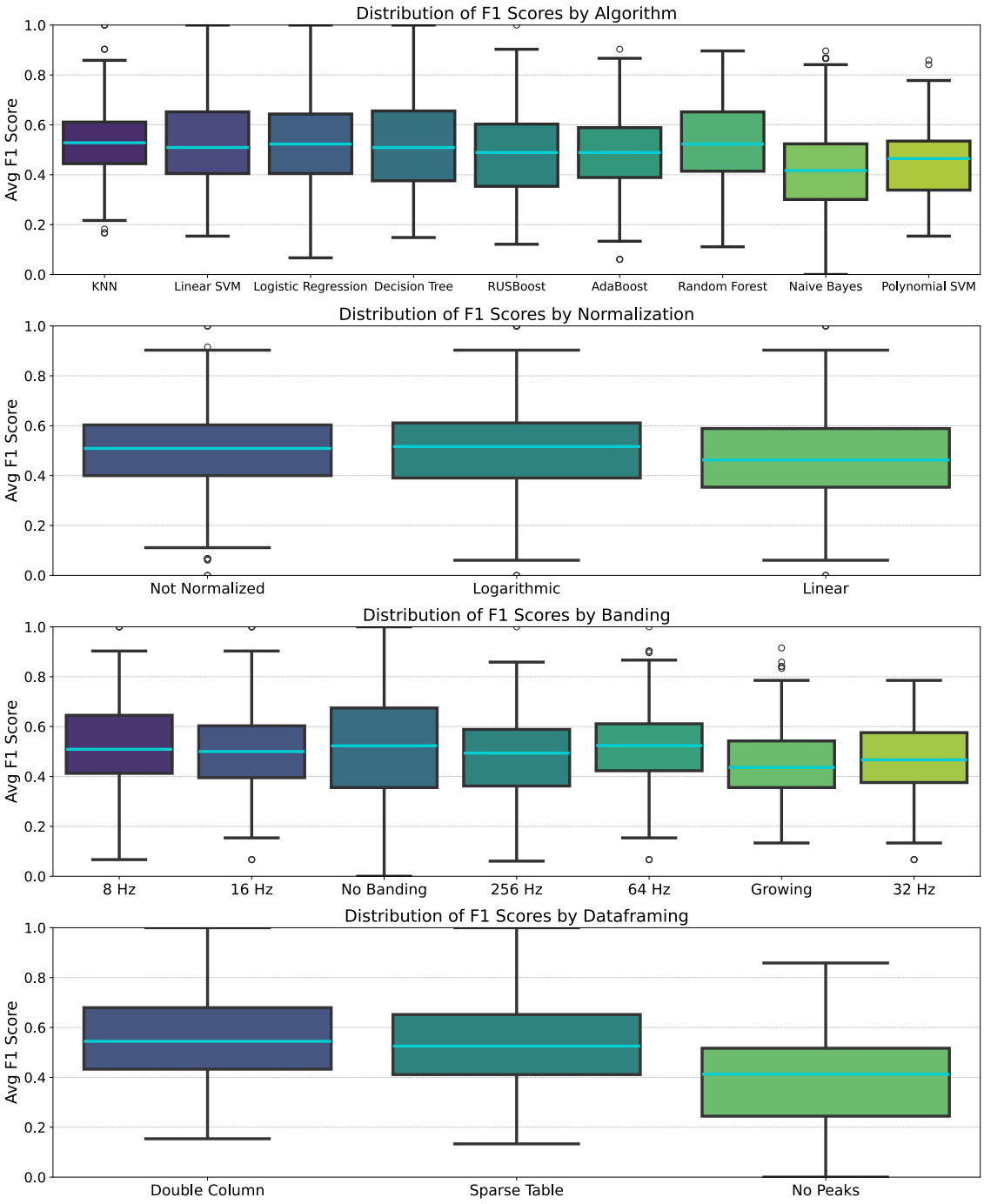


Figure 3.7: F1-score distributions for the testing of the 2835 models trained.

mum values found 1.5 times the interquartile range away from the 1st or 3rd quartile, depending on it being the top or bottom whisker. Datapoints lying outside this range are treated as outliers and are denoted as a rhombuses. Median lines are represented in cyan.

A perfect F1-score would be 1, the closer each boxplot's median line is to 1 the better. Boxplots with top whiskers reaching 1 mean that that method or arlgorithm has

reached a perfect F1-score at least once a.

One thing that is very apparent in Figure 3.7 was just how not using peaks is a worse option than other dataframing methods. 75% of its F1-scores were lower than median of F1-scores of the other dataframing options.

Table 3.7: Top 5 combinations, sorted by the difference between their respective F1-score means and standard deviations, along with their F1-score means.

Banding	Normalisation	Dataframing	Algorithm	F1-score Mean-StdDev	F1-score Mean
8	Logarithmic	Sparse Table	Logistic Regression	0.71036246	0.779230399
64	None	Double Column	Decision Tree	0.70798262	0.768354978
8	None	Double Column	Linear SVM	0.68124782	0.753535354
8	Logarithmic	Double Column	Linear SVM	0.67783814	0.827243867
None	Logarithmic	Sparse Table	Logistic Regression	0.67333942	0.802193362

From the 5 macro F1 results of the different splits for each algorithm and preprocessing possibility, to find the best performing combinations, combinations were ordered by the difference between each of their means and standard deviations. This was done because, as the means of the F1-scores aren't close to a perfect score, high F1-score means might have a big variance for different splits of the data. This will give the combinations with the highest scores while punishing combinations that change too much. The 5 combinations where this difference was highest can be seen in Table3.7.

These 5 combinations presented in Table 3.7 share, in comparison to the pool of all possibilities used, few possibilities for each of the preprocessing methods and classification algorithms. These possibilities, as shown on 3.8, were chosen as the specific methods to use with the data acquired in the automation system, though still using all possible combinations.

Table 3.8: The data processing possibilities that were kept, chosen from Table 3.7.

Type	Details
Banding Width	1 Hz 8 Hz 64 Hz
Normalisation	None Logarithmic
Dataframing	Sparse Table Double Column
Algorithm	Logistic Regression Decision Tree Linear Support Vector Machine

In Figure 3.8 can be seen the distributions of the macro F1-scores of the testing of the combinations chosen after the results from Table 3.7. It features the results of the same training as before, but taking out all models that were preprocessed using methods or trained using algorithms that aren't in Table 3.8. In this figure all quartiles increased for all possibilities that remained, as other options gave worse scores in general.

In Table 3.9 the mean F1-scores for all the combinations featuring the kept possibilities is displayed. In it, the data shows that spectra normalised logarithmically, both not banded and banded at 8 Hz, gave some of the best F1-score results.

It is also possible to see that some of the models failed to converge as they had F1-scores lower than 0.1. This is likely due to grid search matrices that were too limited, lacking more/better suited hyperperameter possibilities.

Table 3.9: Average testing F1-scores for all models trained for all combinations of preprocessing techniques across the 5 train/test splits.

Preprocessing Technique			**Classification Algorithm**		
Normalisation	**Banding**	**Dataframing**	**Decision Tree**	**Logistic Regression**	**Linear SVM**
Logarithmic	No Banding	Double Column	0.552	0.708	0.073
		Sparse Table	0.711	0.802	0.742
	8 Hz	Double Column	0.703	0.773	0.827
		Sparse Table	0.617	0.779	0.774
	16 Hz	Double Column	0.506	0.646	0.076
		Sparse Table	0.594	0.574	0.498
	64 Hz	Double Column	0.521	0.678	0.064
		Sparse Table	0.579	0.532	0.506
Not Normalised	No Banding	Double Column	0.586	0.679	0.633
		Sparse Table	0.629	0.679	0.748
	8 Hz	Double Column	0.057	0.007	0.754
		Sparse Table	0.643	0.593	0.067
	16 Hz	Double Column	0.537	0.064	0.702
		Sparse Table	0.572	0.519	0.526
	64 Hz	Double Column	0.768	0.636	0.665
		Sparse Table	0.589	0.524	0.561

This preliminary data analysis, while limited by its lack of samples and the fact that it used a different data acquisition device, was able to show a definite path in the way these kinds of data should be treated. With it, quite a few preprocessing possibilities were able to be discarded, in what went from the need to train 9 algorithms for 7

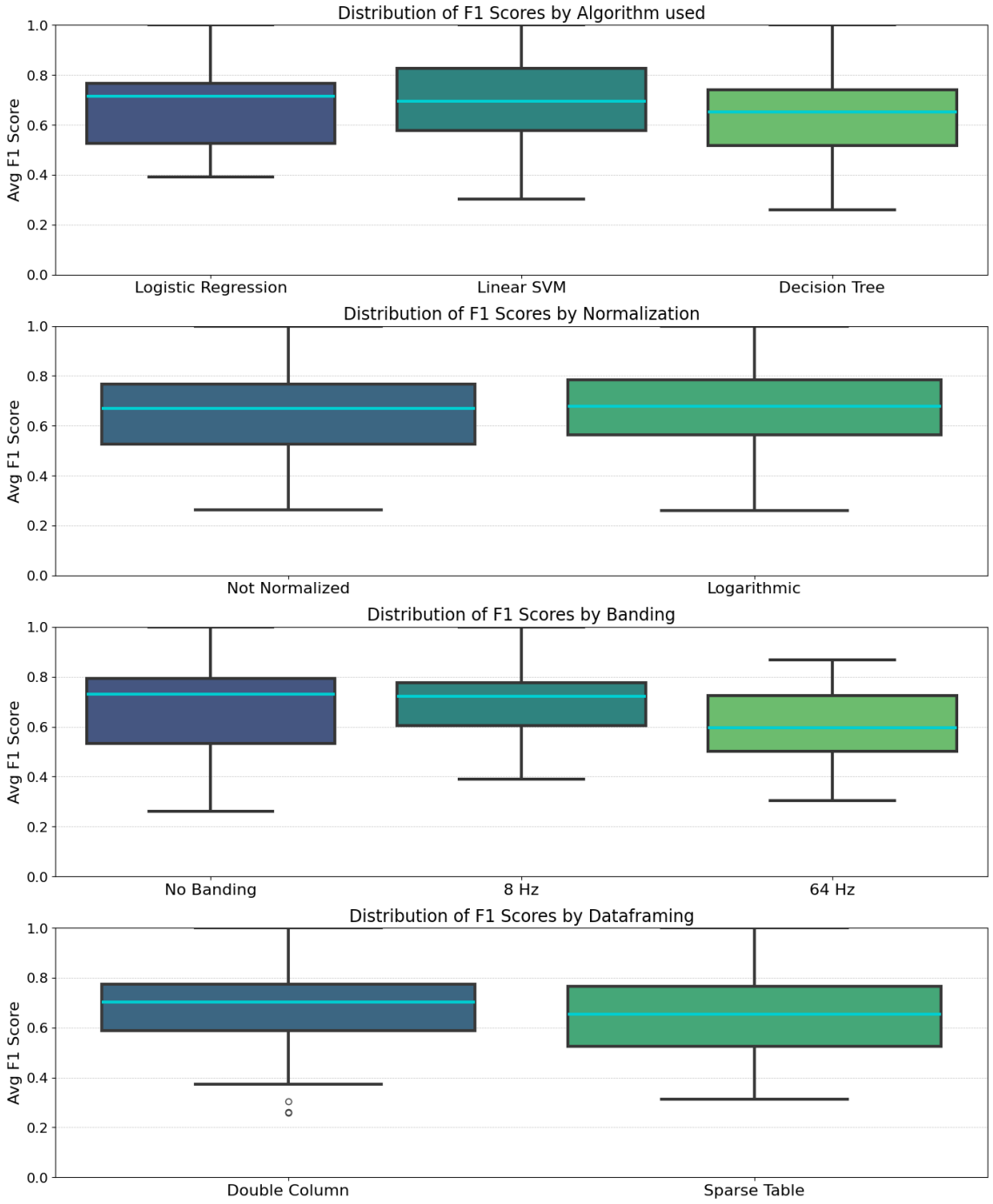


Figure 3.8: F1-scores distributions for the testing of the first set of combinations trained, using only preprocessing methods and models used by the best performing models trained in the preliminary data analysis.

band widths (including no banding, 1 Hz), for 3 types of normalisation (including no normalisation), for 3 dataframing methods, in a total of 567 models; to only 36 different combination possibilities.

Another way that the preliminary data analysis is lacking is the fact that data was

gathered from a pump whose state is commuted using a contactor. While a great number of pumps are still triggered by a contactor, because of its drawbacks, like excessive starting currents, it's lack of pumping process control, increased power consumption and the potential for water hammer, most pumps in industry are controlled by a VFD. VFD-controlled pumps have a different vibration characteristic to that of pumps run directly, using a contactor [40].

— Merenje i instrumenti su osnova naučnog znanja.

Nikola Tesla

4

Data Acquisition in the Automation System

In industry, many electrical machines are commonly used in an automation setting, controlled through a PLC and driven through a VFD. The latter allows for speed control and a low motor start-up current, but are associated with excessive motor vibration [40]. These types of systems often employ networks and can, using the right technology, be controlled and have data acquired remotely — and automatically.

4.1 Automation and Data Acquisition Hardware

With that in mind, a proper automation system was devised, as described in Figure 4.1. In addition to an automation panel, a pressure sensor at the discharge of the pump

would also be added, as per industrial application conventions. Besides the control panel, the manual ball valves would be traded for motorised ball valves so as to be able to control their closure, as motorised valves allow for a much more granular, precise and automated dataset.

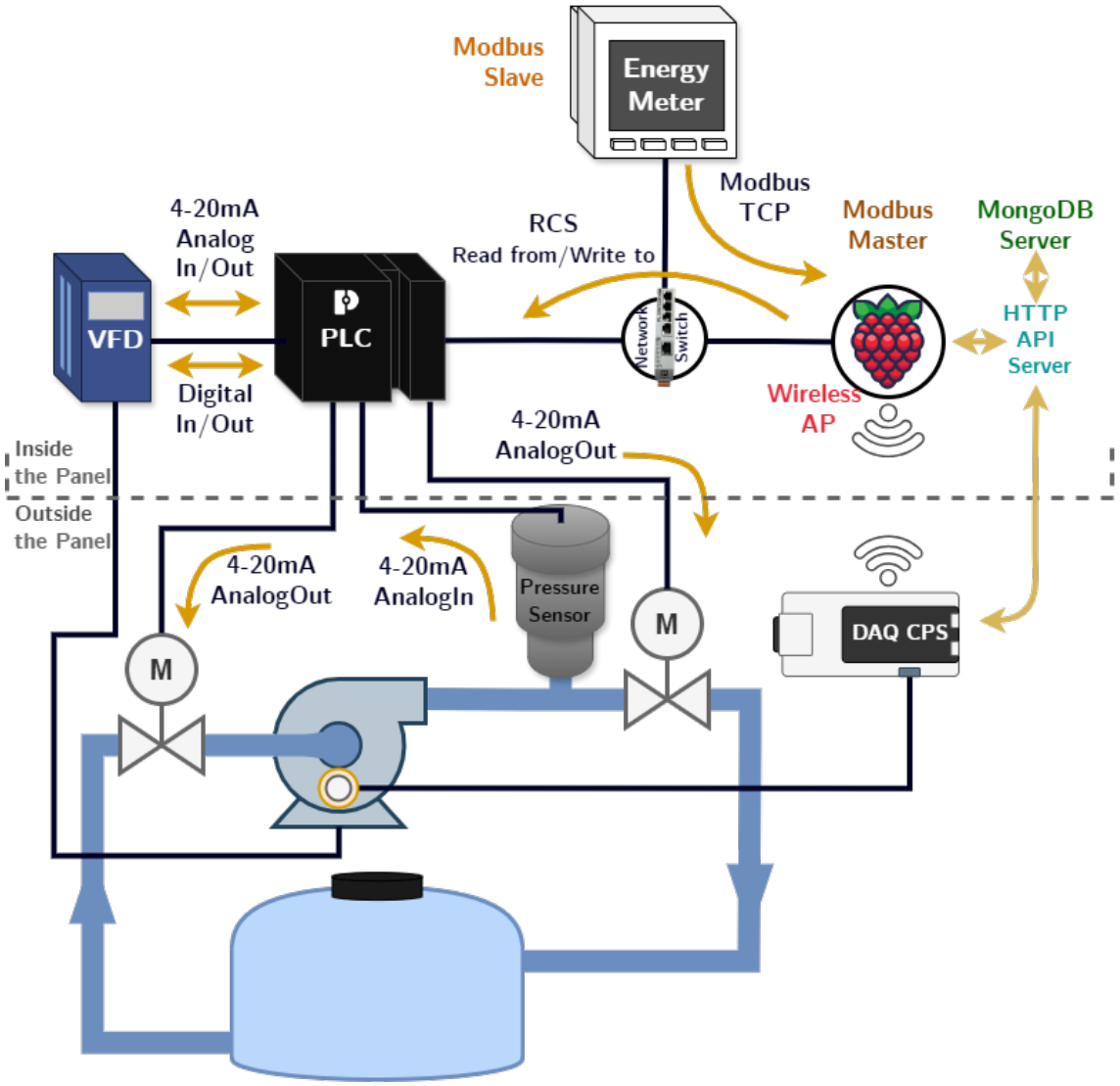


Figure 4.1: A diagram showcasing the automation system, displaying the roles of each device and their communication relationships. In it we can see the closed hydraulic system with motorised valves, a pressure sensor and the pump; the PLC,VFD, energy meter, Raspberry Pi, network switch and DAQ Cyber-Physical System (CPS), connected to the piezo element.

The hydraulic system shown in Figure 3.1 underwent a series of strategic modifications in accordance with the specifications detailed in Figure 4.1. To facilitate these improvements, the original electric control panel was decommissioned and removed from service, as it was incompatible with the new design requirements. Following

the successful implementation of the changes outlined in Figure 4.1, the resulting automated system configuration can be observed in Figure 4.2, which demonstrates the full integration of the updated components and control architecture. This modernization effort has resulted in a more efficient and capable hydraulic system that better serves the intended operational purposes.

Figure 4.2: The automation system built in its entirety.

4.1.1 Automation Panel Hardware

This automation system has its core in the automation panel, whose project can be found in appendix A, which was built from scratch and fitted with many of the staples found in most automation control panels, like a PLC, VFD and an Energy Meter, none of which were part of the previous panel.

(a) (b)

Figure 4.3: The new electric panel, closed (a) and open (b).

As seen in Figure 4.3, the panel has, in its face (Figure 4.3a), from the bottom up, and left to right:

- For security reasons, a physical main power switch that switches off the entire system.;
- A green LED pushbutton, which, when pressed and in manual mode, turns on the pump if the red LED pushbutton isn't lit up and turns on its green light if the pump is running;
- A 3-position switch, made for switching between manual operation, off and automated operation, in which commands are sent by the PLC;
- A red LED pushbutton, which, when pressed and in manual mode, turns off the pump if the green LED is lit up and turns its light on when either the motor circuit breaker is tripped, the monitoring relay is tripped or the VFD gives an error state;
- The energy meter, which is going to be explained in this section;
- A green, yellow and red LED array, which, respectively, indicate the presence of voltage in each of the 3 phases of the system.

This automation panel allows not only the regular use of pump control, but also, because of its ability to be controlled remotely, to be put in a data acquisition mode; and — after data is, offline, duly analysed, preprocessed and a model is trained — have the

resulting model be made to infer the system state runtime, according to what is going to be discussed in Chapter 5.

Programmable Logic Controller (PLC)

As seen in Figure 4.4, the PLC that was used is an *AXC F 2152*, manufactured by *Phoenix Contact*.

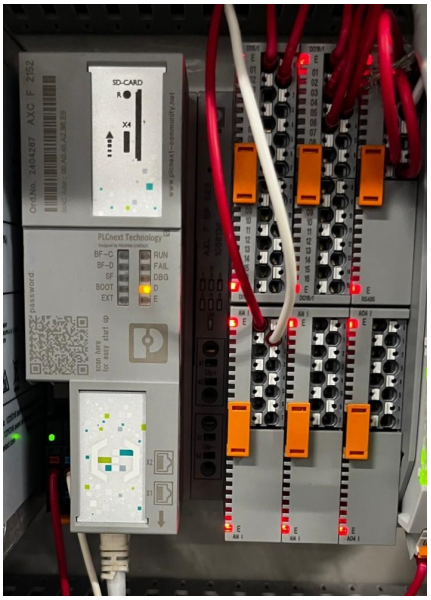


Figure 4.4: The AXC F 2152 PLC

This PLC is a Linux-based computer with the PREEMPT-RT patch, allowing it the ability to execute real-time programs. Software components by Phoenix Contact make it possible to be programmed using any of the IEC 61131 standard programming languages through the *PLCnext Engineer* software; as well as *MATLAB Simulink* and C++.

The company developed an *execution and synchronisation manager* which handles all of the real-time programs running in the machine. But being a Linux-based device the PLC is, of course, able to locally run other, non real-time programs. It is, however, a highly barebones Operating System (OS) compared to what would be usual for a Linux distribution, especially given some of restrictions implemented by the company in the machine. It is hard to make a package manager work well with it and it accepts only storage memory in the form of proprietary — and expensive — standard-sized Secure Digital (SD) memory cards.

Still, non-real-time programs in the operating system or in the local network can access the variables in these IEC 61131 programs through Remote Service Call (RSC), a Transmission Control Protocol (TCP)-based closed protocol that *Phoenix Contact* developed and uses in these kinds of Linux-based PLCs. These RSC services run over a Transport Layer Security (TLS) socket.

The PLC works in tandem with several digital and analogue I/O modules, featuring:

- 16 digital inputs;
- 16 digital outputs;
- 8 4-20mA analogue inputs;
- 4 4-20mA analogue outputs;
- 4 Recommended Standard 485 (RS485) communication ports — which weren't used.

The 4-20 mA analogue communication protocol was chosen for the analogue inputs as current-driven signals aren't susceptible to transmission loss [41], and because it automatically detects a circuit failure by virtue of the signal's baseline (0) being 4 mA.

The PLC's digital inputs and outputs were used as an interface to control the VFD.

The analogue inputs were used to read the pressure sensor and the VFD's frequency, and the analogue outputs were used to control the motorised valves and set the VFD.

It is connected to the panel network switch through an Ethernet port, and can thus be accessed through the local network. The real-time routines of the PLC were programmed using the Function Block Diagram (FBD) programming language, making these routines, by design, reliable.

A Python library developed by the company called PyPlcnextRsc allows a Python routine to create RSC services, effectively controlling the PLC, reading from — and writing to its variables. As previously stated these can be made from within the device itself or from another device, through the local network. This library isn't guaranteed to work for Python versions different from Python 3.9, be them older or more recent.

Analogue Signal Conditioners

For analogue inputs to the PLC, 3-way signal conditioners were used, like the one seen in Figure 4.5, as is usual in industry.

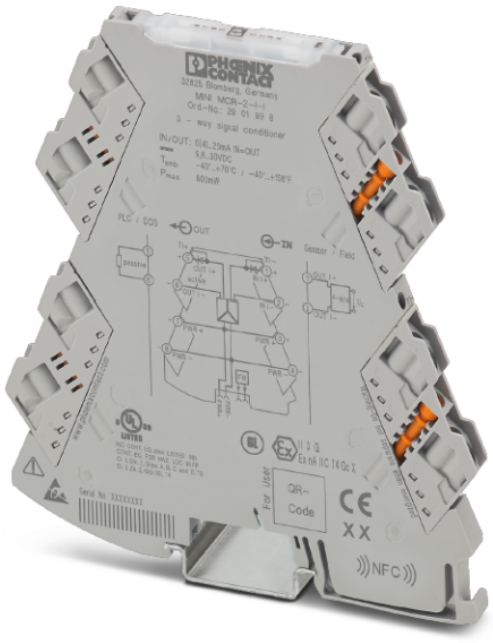


Figure 4.5: 3-Way signal conditioner (extracted the MINI MCR-2-I-I(-PT) manual).

One of these signal conditioners, which requires an external supply of 9.6-30 V DC, was used between the pressure sensor signal wires and the PLC analogue inputs to reduce electrical interference signal, but mainly to electrically isolate the signals, serving chiefly as a safety device for the PLC's analogue input module. Another was used to receive data about the VFD's current wroking frequency. The signal conditioner used in this case was a MINI MCR-2-I-I(-PT) manufactured by *Phoenix Contact*.

Variable Frequency Drive (VFD)

The VFD used, as seen in Figure 4.6, is a *Universal Motors* Goodrive20, set up in V/F Mode, commanded by the PLC through an analogue output for frequency set and some digital outputs for actuating. The VFD is set up to control the pump's frequency as the PLC commands it through one of its analogue outputs. The only problem being that the analogue input of the VFD are, contrary to the output modules chosen for the

PLC, 0-10 V.

Aditional digital outputs were used to give information to the PLC about the state of the VFD, such as if it was running or not, or if it was in an error state. What each of these digital VFD outputs does had to be parameterised through the VFD's display.

Figure 4.6: The used VFD, *Universal Motors* Goodrive20.

This solution was the best possible as, although advertised, this particular VFD available information wasn't enough to properly set up Modbus communication over RS485.

Interfacing Electronics

To interface the PLC with the VFD, because of their aignal incompatibility a purposely designed circuit had to be built. As stated before The PLC's output are 4-20 mA while the VFD's inputs are 0-10V.

So, to control the frequency setpoint of the VFD with the PLC the circuit, as described by Figure 4.7, was assembled. The circuit implements the function described

by equation (4.1), which allows to have 0V and 10V at the output, for input currents of, respectively, 4mA and 20mA.

$$V_o = 627.7 \times In - 2.6 \text{ (V)} \tag{4.1}$$

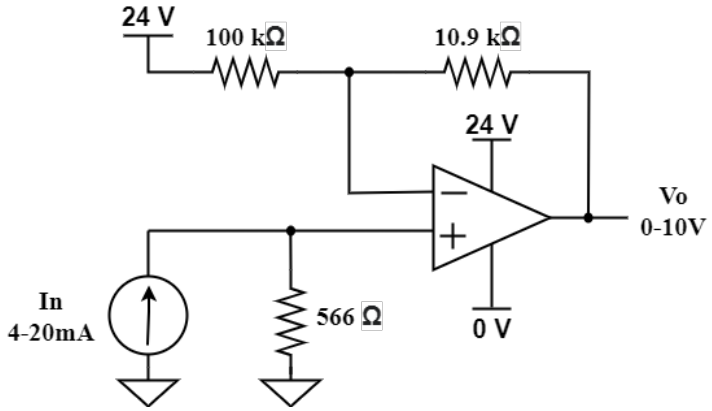


Figure 4.7: Circuit developed for the conversion of a 4-20 mA signal to a 0-10 V signal.

Energy Meter

An energy meter is not an uncommon sight in industrial applications, as it is used to monitor the energy consumption of the system. The energy meter used, seen in Figure 4.3a, whose display can be seen in Figure 4.8, is an *EEM-MA770* by *Phoenix Contact*, connected to the 3 phase power system using 4 wires, with current being measured through 3 50/5 current transformers. It is connected through Ethernet to the panel network switch and communicates through Modbus TCP/IP, as a slave. It is able to measure, every second:

- phase and line voltages;
- phase and neutral currents;
- active, reactive and apparent power;
- power factors;
- frequency;
- phase angles;
- Total Harmonic Distortion (THD) of voltages and currents;
- the first 63 harmonics of each voltage and current.

Figure 4.8: The EEM-MA770 energy meter.

All of these values were requested and received for data acquisition. Some other values that the *EEM-MA770* is able to measure, like those related to metering, were not requested as they aren't relevant for the scope of this thesis. This particular energy meter has a built-in web server, which can be seen on Figure 4.9, which allows for the easy display and monitoring of the data it is acquiring.

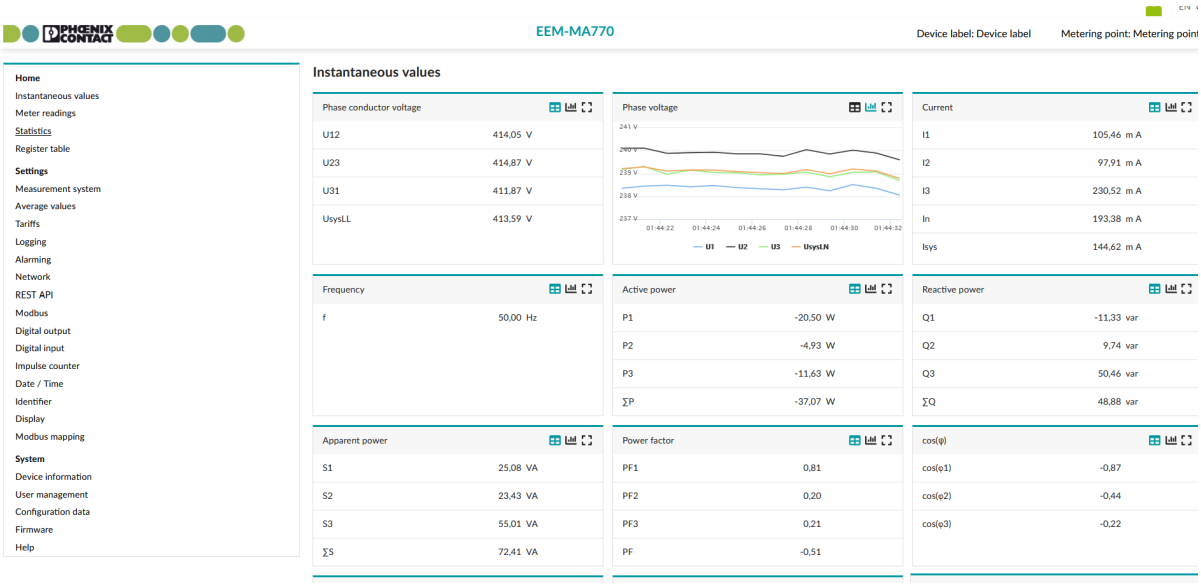


Figure 4.9: The energy meter's web page, showing some of it's capabilities.

Raspberry Pi

The Raspberry Pi used was a Raspberry Pi 5 with an active cooler, and a PCie M.2 2280 Non-Volatile Memory Express (NVMe) Shield with a 500 GB NVMe SSD, as seen in Figure 4.10. It is powered by one of the internal outlets on the automation panel.

Figure 4.10: The Raspberry Pi used, in the automation panel.

The Raspberry Pi serves as the command centre for the predictive maintenance system within the automation system itself, as Figure 4.1 shows:

- the Raspberry uses the RSC to control the PLC;
- It serves a Mongo Database, through a container;
- In addition, it hosts an Application Programming Interface (API) built with Flask that allows communication with the MongoDB — the API is also able to trigger the Vibration DAQ CPS (more on this in subsection 4.1.3);
- It also serves as a wireless Access Point (AP) for the local cabled network, so that the DAQ CPS can interact with the system wirelessly;
- Moreover, the Pi is also a Modbus master.

While managing all that, the Raspberry Pi ran the whole data acquisition routine, controlling the valves, and VFD frequency through the PLC, using the RSC at all the combinations possible with the parameters from Table 4.1 and then running the pump to steady state, using the same method, waiting 20 s to take measurements: through

Modbus acquire most data from the energy meter, according to a curated list; getting the pressure value from the PLC and a list of voltages corresponding to vibration coming from the Vibration DAQ CPS.

The Raspberry Pi runs Raspbian OS and can be accessed through either regular Secure Shell (SSH) and Virtual Network Computing (VNC), as well as using File Tranfer Protocol (FTP) to transfer data between a terminal to load scripts to and from it.

Thermal Management

A thermal management system was designed for the automation panel, to maintain optimal operating conditions for the electronic components. A C-curve 10 A circuit breaker-protected thermostat-controlled cooling fan system was implemented, consisting of a 230 V AC fan and a bimetallic thermostat set to trigger at 20 °C. This setup was important especially in the hotter months of the year, given the heat generation from several components, particularly the VFD and power supply. A fan was positioned at the top of the panel to take advantage of natural convection, while a filtered vent at the bottom allowed for proper air circulation.

(a) (b)

Figure 4.11: The thermal management system, with the thermostat (a) and respective fan (b).

Protection Devices

As can be seen in Appendix A, several protection devices were integrated into the automation panel to ensure safe operation and protect both equipment and personnel:

A three-phase monitoring relay, a *Phoenix Contact* EMD-BL-PH-480-PT, was installed to monitor the quality of the power supply. Set to trip at an incorrect phase sequence or a 10% asymmetry after 2 seconds, this device protects the system from damaging operating conditions that could arise from supply issues.

A motor circuit breaker set to trip at 4 A was used, providing both short-circuit and thermal overload protection. This device was specifically selected to match the motor's characteristics and the starting current requirements when operated through the VFD. It is attached to an accessory dry contact for use in the command of the electric panel.

A 4-pole fused disconnect isolator was installed using 2 A fuses to protect both the phase indicating lights and three-phase monitoring relay.

A 1-pole fused disconnect isolator was installed using 10 A fuses to protect the energy meter.

A C-curve 6 A circuit breaker protects the 24V DC power supply that supplies all the relay command and motorised valves.

A C-curve 10 A circuit breaker protects the thermal management part.

A C-curve 16 A circuit breaker protects 3 parallel Schuko sockets.

A 300 mA, 25 A residual current circuit breaker was included to provide protection against ground faults and indirect contacts, essential for personnel safety given the presence of both high and low voltage circuits within the panel.

These protection devices were coordinated to ensure selective operation, with the device closest to any potential fault acting first, thus minimizing the impact of any electrical issues on the overall system operation.

Figure 4.12: One of the 2 motorised valves, installed in the system.

4.1.2 External Automation Hardware

Motorised Valves

The motorised valves, as seen in Figure 4.12, are controlled by the PLC through each of its 4-20 mA analogue outputs. They are used to stress the pump, by closing the inlet and outlet valves to different percentages, and are used to simulate different operational conditions, like a clogged pipe or a closed valve, in an automated way.

It is important to note that **as detritus and impurities accumulate in an hydraulic system, this will start to gradually detriment it**, and, if that happens in the suction or in the discharge, it will impact the pump and system in different ways. **The changing of the valves to various points just serves as a proxy for different types of problems that may happen in a pumping system**. The valves serve as a **fault injection** method in the laboratory, to test system behaviour in different operating conditions. However, during on-field operation, the detritus and impurities will induce system behaviours similar to the testing conditions, **allowing error detection and diagnosis** (i.e., know where the problem may lie).

These motorised valves have a built-in flow switch which does not allow the valve to change position if any fluid is flowing through it — it will not only refuse to start if some water is still flowing even after the pump has been turned off, but also will stop if the pump starts. Even though it would seem as if having to stop to change position adds time to the data acquisition process, these delays should always be present, as a

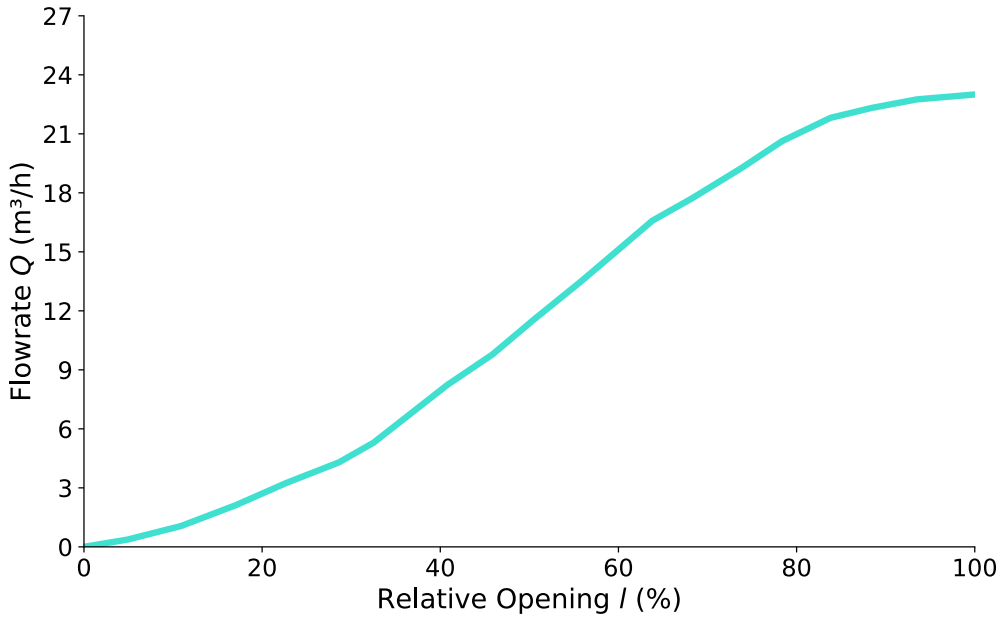


Figure 4.13: Example of a ball valve opening/flowrate characteristic, adapted from [42].

system will take seconds to stabilise its pressure and flow if a valve is opened or closed a certain amount while fluid is being pumped [42].

Ball valves like this have a sigmoid-looking characteristic as shown in Figure 4.13, meaning that near fully open and fully closed, differences in the valve's closure mean little difference in the flowrate of fluid [42]. This was taken into consideration in the decision of the valve setpoints in Table 4.1, concentrating the granularity where changes in the relative opening of the valve will imply a bigger difference in flowrate.

Pressure Sensor

It is very common, in most industrial applications, to have a pressure sensor somewhere in the discharge of the pump. In light of this, a pressure sensor was added to the system in its discharge, as seen in Figures 4.1 and 4.14. Because pumping systems usually have a pressure sensor only at the discharge, no other pressure sensor was added to the system, so as to model what would be a common industrial application. The pressure sensor used is a 1.6 MPa, 4-20 mA sensor, read by the PLC through one of the analogue inputs, through the analogue signal conditioner.

Figure 4.14: The pressure Sensor used.

4.1.3 External Electronic Hardware

Vibration Data Acquisition (DAQ) Cyber-Physical System (CPS)

In addition to the hydraulic system's usual hardware, a non-invasive, portable DAQ CPS (Figure 4.17) developed by Brito [16] that's used to acquire vibration data from the pump, was added to the system.

The hardware is powered by any power outlet close to the pump. It takes vibration data from the pump through the use of a piezoelectric element that's coupled to the pump as seen in Figure 4.16.

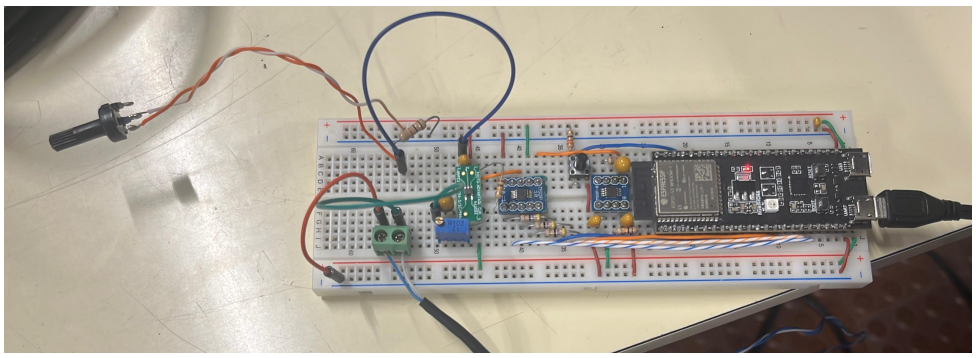


Figure 4.15: Brito's first version of the Vibration DAQ CPS, as it was acquiring data from the pumping system.

Figure 4.16: Coupling of the piezoelectric element to the pump.

The first version of Brito's DAQ CPS used a 16-bit, high speed Analog to Digital Converter (ADC), taking 24000 samples at 24 kSa/s, and publishing them through the Message Queuing Telemetry Transport (MQTT) protocol, using a Raspberry Pi 4 that was added to the automation panel. The API developed by Brito ran on a Raspberry Pi 4, which was also an MQTT broker as the API was, then, made to work with MQTT, saving to a MongoDB collection both the raw vibration samples and the spectrum, with the accompanying frequencies, which it processed in Brito's API.

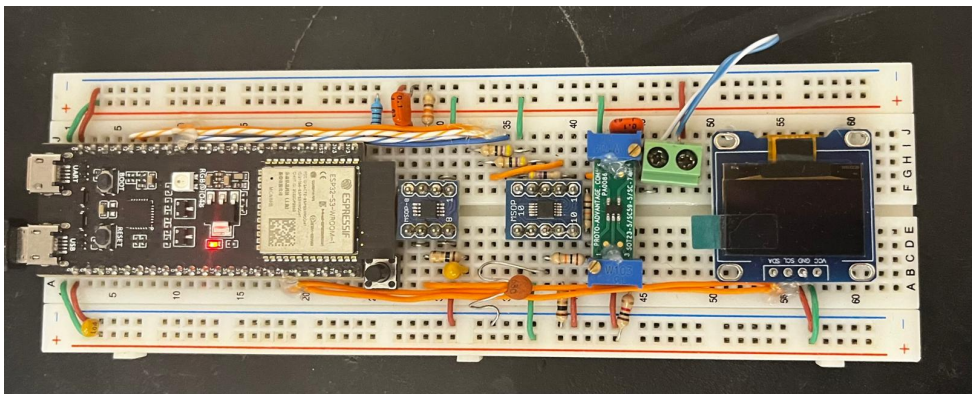


Figure 4.17: Brito's new Vibration DAQ CPS, as it was acquiring data from the pumping system.

The data from this version, as per the Nyquist theorem, could return a frequency spectrum with a maximum of 12 kHz. This, combined with its low-pass filter to avoid aliasing, made it so that frequencies in the higher part of the resulting spectrum weren't

as trust-worthy, while still being well under desired range of interest for vibrational analysis in electrical machines, which can be from 4 kHz to 20 kHz in what respects to cavitation [11, 43].

Some time having passed, Brito was able to develop a second version of the vibration DAQ CPS, updating the hardware and firmware. Brito also overhauled the API in the Raspberry Pi 4 and built it around interfacing a relational database.

The second version of Brito's DAQ CPS used the same 16-bit, high speed ADC, taking 100,000 samples at 100 kSa/s — giving one magnitude per natural frequency value, through the FFT.

This new version did not use MQTT and so the new Raspberry Pi (5) did not run any broker and an Hypertext Transfer Protocol (HTTP) API was able to be developed using Flask to interface with the device, making it take a sample and respond with those samples' values.

4.2 Data Acquisition

As stated in subsection 4.1.1, the Raspberry Pi commanded the data acquisition routine as well as running several daemon-like processes necessary for the data acquisition procedure, including a Mongo server, an API and, for the first version of the Vibration Data Acquisition (DAQ) Cyber-Physical System (CPS), an Eclipse Mosquitto MQTT broker. In Python 3.9 — because of the RSC library limitations — a data acquisition routine was developed for the Raspberry that, given the closures for both the inlet and outlet valves and the set frequency of the VFD automatically creates a dataset.

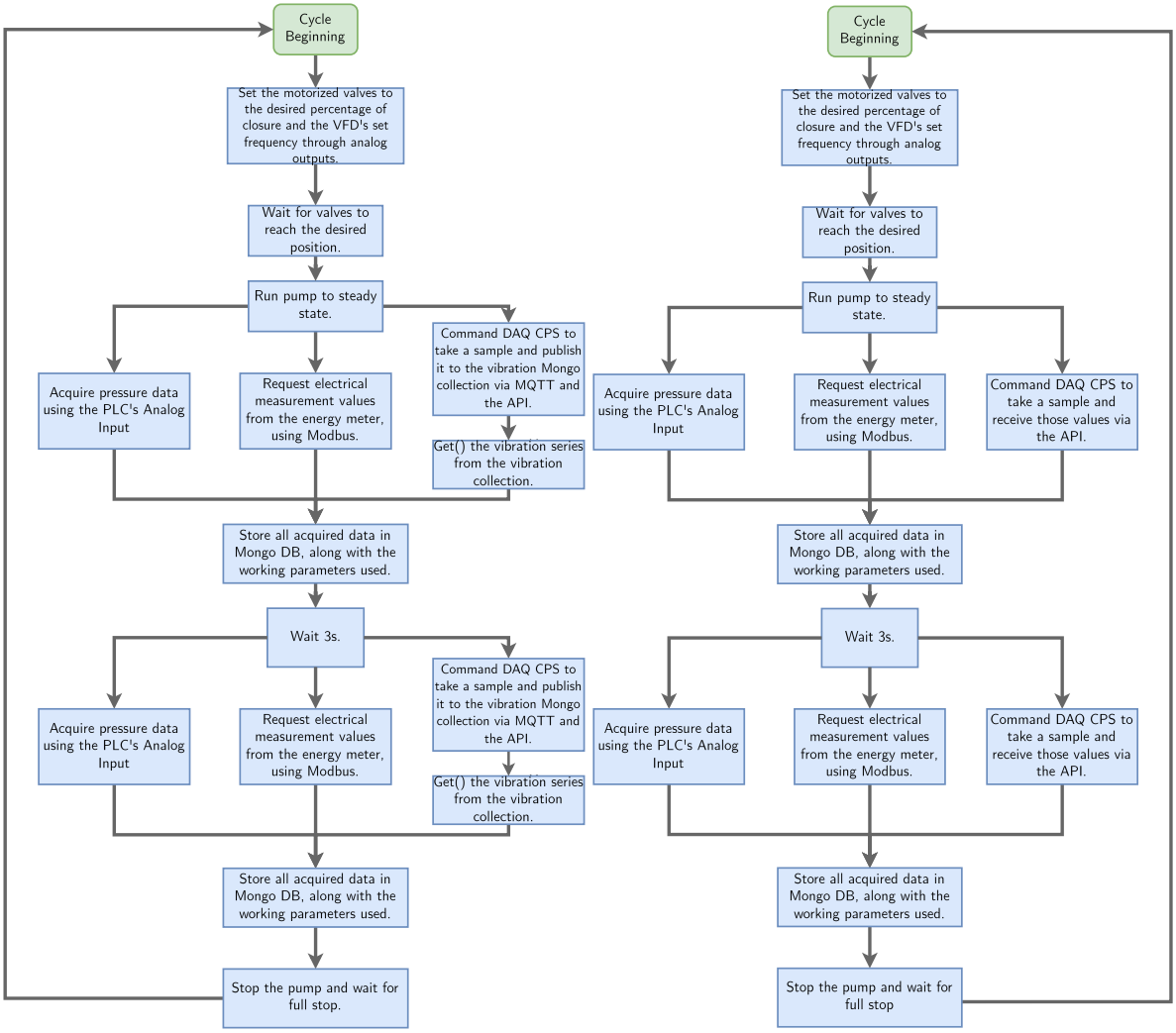


Figure 4.18: Flowcharts of the data acquisition cycle, old one (left) vs. new one (right).

The data acquisition routine is mostly a cycle as described by Figure 4.18 that repeats itself for how many combinations of system of parameters there are, storing all values in a Mongo Database. Before the cycle begins it just makes sure that the pump is turned off and to turn every valve to 0% of closure, to start fresh.

The biggest difference between the two workflows is that when data was taken using the first version of Brito's work, as it was already going to a collection within the Mongo Database. Those values were simply extracted from the collection and put into a dictionary along with the of all the set system working parameters for that sample, the pressure measured, acquired through RSC and all of the values coming from the energy meter acquired through Modbus. In the second version, the API being different, all data was just taken, via the API, RSC and Modbus and just stored like before, but

now using only one MongoDB collection. On Figure 4.19 can be seen an example of an entry from this collection.

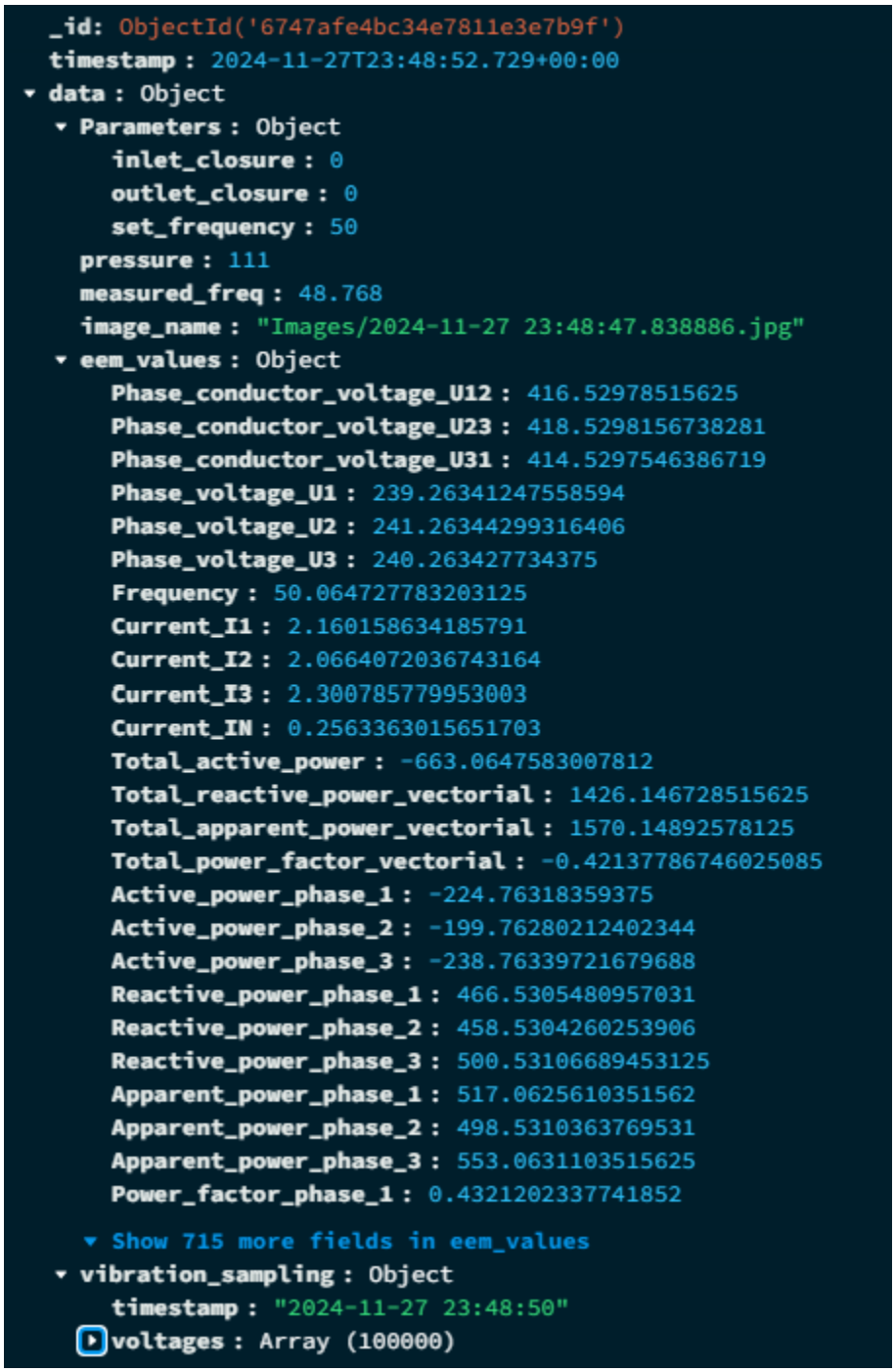


Figure 4.19: Example of a Mongo Database entry from the Data Acquisition Collection.

To collect the data from the DAQ CPS, pressure sensor, and energy meter, 2 new samplings were performed, with seconds of difference, for each combination of the system working parameters seen on Table 4.1, making a total of 512 different samples.

Table 4.1: The system parameters for the new data acquisition.

Parameter	Details
Frequency	30 Hz 40 Hz 50 Hz 60 Hz
Inlet Closure	0% 20% 35% 45% 55% 65% 80% 100%
Outlet Closure	0% 20% 35% 45% 55% 65% 80% 100%

As the valves don't work if fluid is flowing the amount of time that's waited until they reach the desired position is dependent on how big the difference is between their old value and new one.

— Las cosas podían haber sucedido de cuaquier otra manera y, sin embargo, sucedieron así.

Miguel Delibes

5

Data Analysis

Data coming from data acquisition, as it was implemented, came in 3 groups:

- discharge pressure;
- energy data and
- vibration sampling.

In form of a JavaScript Object Notation (JSON) file, coming from a MongoDB collection. The nested JSON structure was transformed into a Comma-Separated Value (CSV) file, a flat, tabular format by extracting leaf nodes, i.e. keys directly associated with values, and using them as column headers. Intermediate hierarchical relationships and parent nodes were omitted, retaining only the terminal data points.

5.1 Vibration

The vibration sampling, as described in the last chapter, was, initially, of 24000 samples at 24 kSa/s. After the FFT algorithm, and using only the first half of the absolute value of the result, this gives a frequency spectrum that goes from 0 to half the sampling frequency minus 1 — 12 kHz, in this case; 50 kHz for the second data acquisition.

Because the signal was centred at a positive voltage value so that vibration wouldn't saturate the amplifier (as seen in Figure 5.1), the frequency spectrum has a very high Direct Current (DC) component — the 0 Hz component of the frequency spectrum. This DC component is, then, not only unimportant but it also dwarfs the other frequency components. For these reasons, the frequency component at 0 Hz was eliminated.

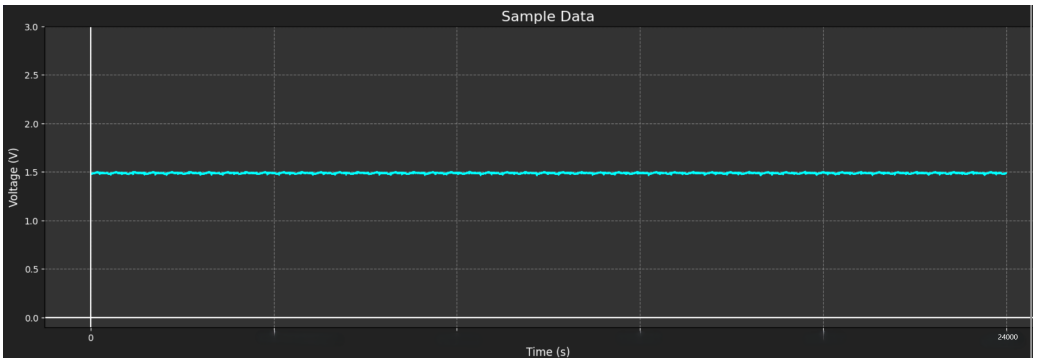


Figure 5.1: Example of a vibration sample as it comes from the vibration DAQ CPS.

To this resulting spectrum were applied the chosen methods of preprocessing in Chapter 3, displayed in Table 3.8, and each of the combinations of preprocessing methods were trained using the 3 different algorithms in that table, a total of 64 models.

The preprocessing pipeline is then, after calculating the spectrum:

1. band 1-1, 8-8 or 64-64 Hz;
2. apply logarithmic normalisation — or not;
3. get the n highest peaks;
4. dataframe the resulting series, using either Double Column or Sparse Table (see section 3.2 for how each works).

Figure 5.2 shows, for one spectrum, the processes through which the 512 spectra

would have to go through to then still be dataframed in both Double Column and Sparse Table for the 64 Hz example. It is of note that all of the 3 first plots have had their peaks found and were, through them, dataframed and used to train models, not only the 3rd. That is, they are all part of the combinations possible in Table 4.1: the first is a 1 Hz-banded non normalised, the second is a 64Hz-banded non-nomralised and the third one is the 64Hz-banded logarithmically nomralised.

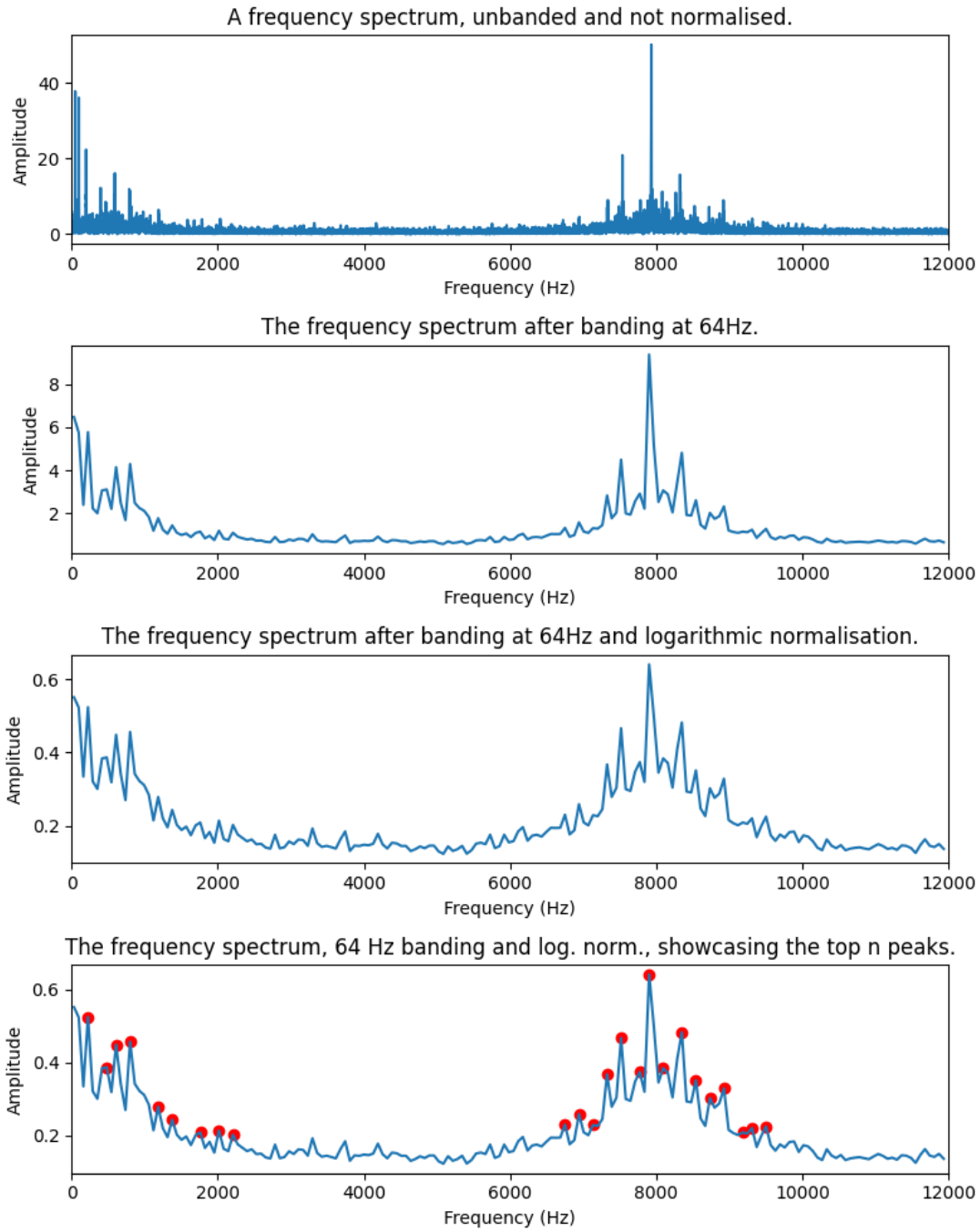


Figure 5.2: One Frequency Spectrum through the steps necessary to arrive at the highest n peaks for a 64 Hz-banded logarithmically normalised spectrum.

After both banding and applying — or not — normalisation, there are a certain amount of peaks in each of the 512 spectra. After running all the spectra in a given combination of preprocessing methods, as explained in Section 3.2, the number of peaks of the sample with the spectrum with the lowest amount of peaks found is halved and that number is then set as the n number of highest peaks that will be used in all of the 512 spectra of the combination. This n per combination is then stored, as it needs to be used for inference. The numbers used and stored for the samplings made with the first and second vibration DAQ CPS versions can be found in Table 5.1.These numbers of course will change sampling to sampling, but they need to be stored as they are important to be be consistent and use them for inference.

The reason for using half the lowest number of peaks found in a sample is because, as there are spectra with more peaks than others, it is not only plausible but also probable that a sampling would be made to infer whose frequency spectrum had lesser peaks than the spectrum with the least peaks in the training spectra. This way, a big margin of error is given.

Table 5.1: Number of peaks used by banding width with the samplings made. The number is the same for non-normalised and normalised spectra.

	Banding Width		
	1 Hz	**8 Hz**	**64 Hz**
Number of Peaks: 12 kHz	1937	217	24
Number of Peaks: 50 kHz	8211	885	108

5.2 Additional Values

To the dataframed peaks obtained from the vibration analysis, additional sensor data was integrated to provide a more comprehensive view of the system's state. Specifically, two types of measurements were concatenated:

1. The discharge pressure value obtained from the pressure sensor installed at the pump's outlet. This single but crucial measurement provides direct information

about the hydraulic condition at the discharge of the system.

2. The remaining energy meter parameters that showed variation across samples. From the initial 741 features collected by the energy meter, only 419 were retained after removing those that maintained constant values throughout all measurements. These parameters include:

    - Phase and line voltages
    - Phase and neutral currents
    - Active, reactive, and apparent power measurements
    - Power factors
    - System frequency
    - Phase angles
    - Voltage and current THD values
    - Most of the first 63 harmonics of each voltage and current measurement

The 419 measurements that had been left had to be saved for the future, because those are the ones that are needed to be used for inference.

The concatenation of these values with the processed vibration data resulted in a comprehensive dataset that captures both mechanical vibration characteristics and electrical operating conditions of the system. This multi-sensor approach is supposed a richer set of features for the subsequent machine learning analysis, potentially enabling more robust fault detection and classification capabilities.

5.3 Training and Classification

The resulting dataframe was then preprocessed using a combination of Standard Scaler and PCA. The PCA transformation was configured to retain enough principal components to explain at least 95% of the variance in the data. Following established machine learning practices, 5-fold cross validation was employed along with stratified shuffle split. 15% of the total data was allocated to be the test set, corresponding to 77 samples.

For classification purposes, a novel 9-class labelling system was developed, as illustrated in Table 5.2. Each class is represented by a 4-letter code, with the first and third letters indicating the closure state (L, M, or H) of the inlet and outlet valves respectively — which, again, serves only as fault injection. The classification thresholds were defined as follows:

- For inlet closure (first letter):
  - L: less than 35%
  - M: between 35% and 65%
  - H: greater than 65%
- For outlet closure (third letter): same thresholds apply

Table 5.2: Class distribution of the new target as described by inlet closure vs. outlet closure.

Outlet Closure	**Inlet Closure**							
	0 %	20 %	35 %	45 %	55 %	65 %	80 %	100 %
0 %	LILO	LILO	MILO	MILO	MILO	MILO	HILO	HILO
20 %	LILO	LILO	MILO	MILO	MILO	MILO	HILO	HILO
35 %	LIMO	LIMO	MIMO	MIMO	MIMO	MIMO	HIMO	HIMO
45 %	LIMO	LIMO	MIMO	MIMO	MIMO	MIMO	HIMO	HIMO
55 %	LIMO	LIMO	MIMO	MIMO	MIMO	MIMO	HIMO	HIMO
65 %	LIMO	LIMO	MIMO	MIMO	MIMO	MIMO	HIMO	HIMO
80 %	LIHO	LIHO	MIHO	MIHO	MIHO	MIHO	HIHO	HIHO
100 %	LIHO	LIHO	MIHO	MIHO	MIHO	MIHO	HIHO	HIHO

The last approach (As used in subsection 3.1.2), while being a simple way to suggest the state of the pump, it is somewhat simplistic and gives little information about the state of the pumping system besides that there's a problem somewhere in the hydraulic system. The proposed 9-class approach gives information on the state of the hydraulic system's suction and discharge, which, of course, gives much more information on the general well being of the system.

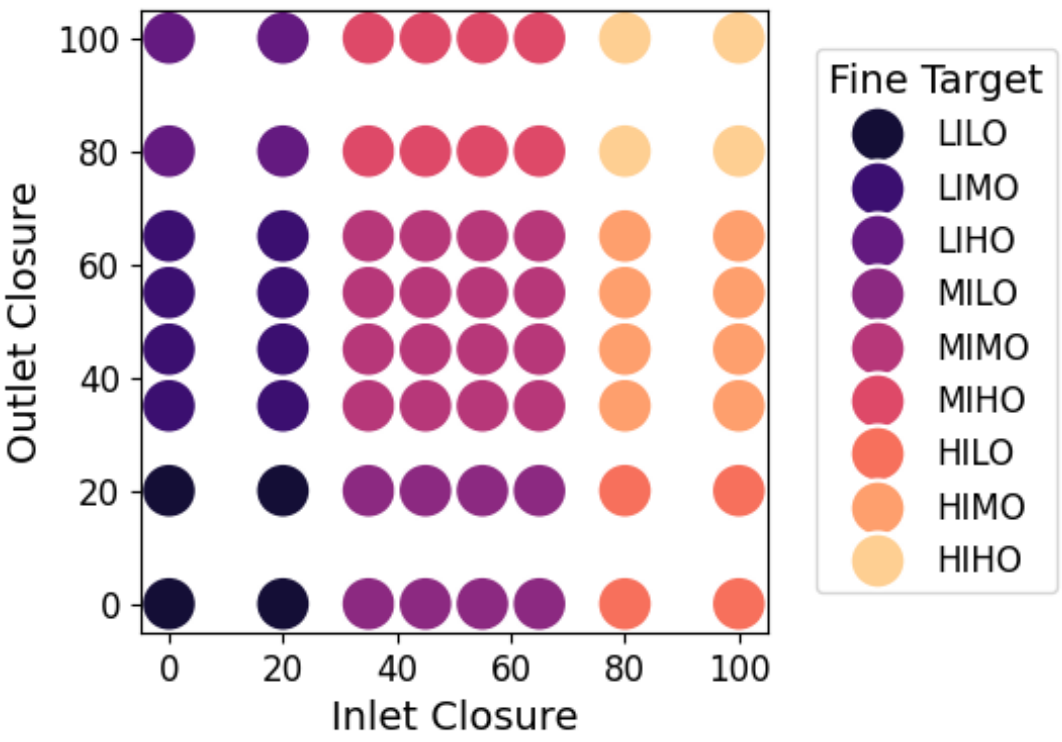


Figure 5.3: Distribution of classes for the 9 class approach.

It is important to note that each of the setpoints of inlet and outlet valve closure. Two samples were taken at each of 4 different VFD frequencies. The desired frequencies were 30, 40, 50 and 60 Hz.

The training was carried out on each of the 8 different possibilities of banding, normalisation and dataframing times the 3 chosen algorithms. Converging without much difficulty on some of the combinations.

Table 5.3: Class distribution of the new target by number of samples

Class	LILO	LIMO	LIHO	MILO	MIMO	MIHO	HILO	HIMO	HIHO	**Total**
Samples (Spectra)	32	64	32	64	128	64	32	64	32	**512**

Relative to the previous approach, this one is also **much** more balanced approach in its calss distribution as can be seen in Table 5.3, relative to Table 3.1.2. There is still some improvement to be made in class distribution balance, though.

— Человек он умный, но чтоб умно поступать

— одного ума мало.

Достоевский, Фёдор Михайлович

6

Experimental results

This chapter presents the experimental results obtained from the process of implementing and testing an automation-based automated predictive maintenance system. The analysis begins with a very high-level a simple overview of what was made in Chapter 3, after which an evaluation of the physical implementation of the proposed system through the automation panel construction and setup, followed by a detailed examination of the data acquisition and classification results using different sampling rates, preprocessing techniques, and machine learning algorithms.

6.1 Actuating Panel

The work began with a simple setup that allowed the pump to be switched on or off and from which vibration data was gathered, with a piezzo-element, and a commer-

cial analogue signal capture card, in order to evaluate the characteristics of data that would be handled, and how to best treat them. Throughout Chapter 3, as the results from training all possible combinations it was possible to reduce the training load of all available possibilities that were initially thought of for this kind of vibration analysis from 567 to 36 possibilities, making the work in Chapter 5 so much more manageable. The objective of the chapter was to better understand vibrational data and to understand what really doesn't work. Given how much less possibilities there were for the future it could be called a success.

6.2 Industry 4.0 Automation Panel

As a laboratory experimental setup, the system operates in a closed loop without specific operational objectives, although any particular routine can easily be programmed, in IEC611301-3 languages, through *PLCnext Engineer* to run any process routine for an industrial application.

The automation panel was constructed entirely from scratch, involving detailed designing and assembly of all components including and establishing electrical connections following general best-practice guidelines. This ground-up construction approach ensured that every aspect of the panel met industrial standards while incorporating the specific requirements for data acquisition and subsequent system monitoring.

This panel is then part of a wider IIoT system, made completely with the Industry 4.0 paradigm in mind, that's fully controllable and monitorable from anywhere with access to the network the panel is in.

Using Python, several scripts were created for the **full** control of the pumping system using the Raspberry Pi, interfacing the PLC using RSC, the energy meter with Modbus requests, and the vibration DAQ CPS through the API. The data acquisition routine developed as described in Section 4.2, is just an example of the amount of control that the developed system allows.

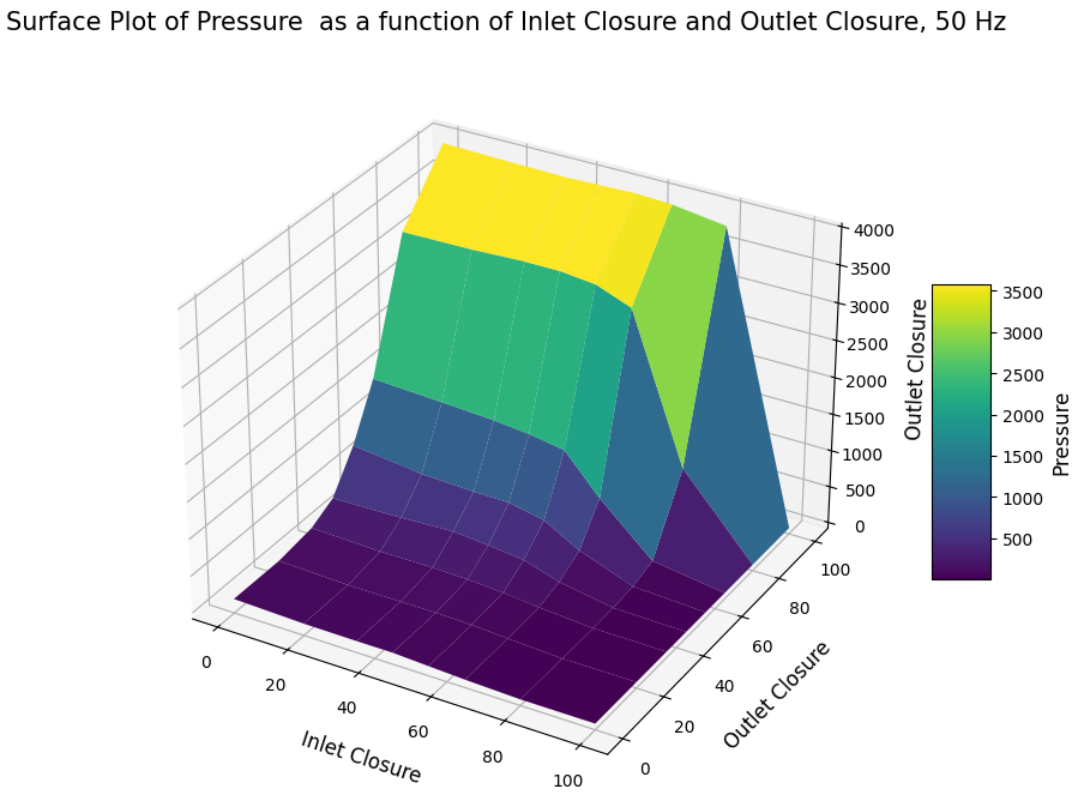


Figure 6.1: Pressure as it relates to the closure of the inlet and outlet valves, at 50 Hz.

As described by 4.1, all access to the values coming from the pressure sensor and control of the valves is done by the PLC. During the data acquisition process, because the Raspberry Pi controls the PLC, relationships like the ones shown in Figure 6.1 can be better understood.

The fact that the Raspberry Pi is a Modbus master and can make Modbus requests to the energy meter makes it possible to analyse relationships like the one on Figure 6.2, where we can see how Active Power relates to the closure of the inlet and outlet valves.

Integration with Brito's hardware and software was successfully completed and reliable vibration data was taken. Figures 6.3, 6.4 and 6.5 are example results from system operation that show, in a visual form, the effectiveness of the test bench system, and that vibration analysis can be used to uncover, in the spectra, changes occurred in the inlet or outlet closures. To aid readability of the results and allow a visual representation to the reader, the 3 figures represent small parts of spectra, in this case banded to 64 Hz.

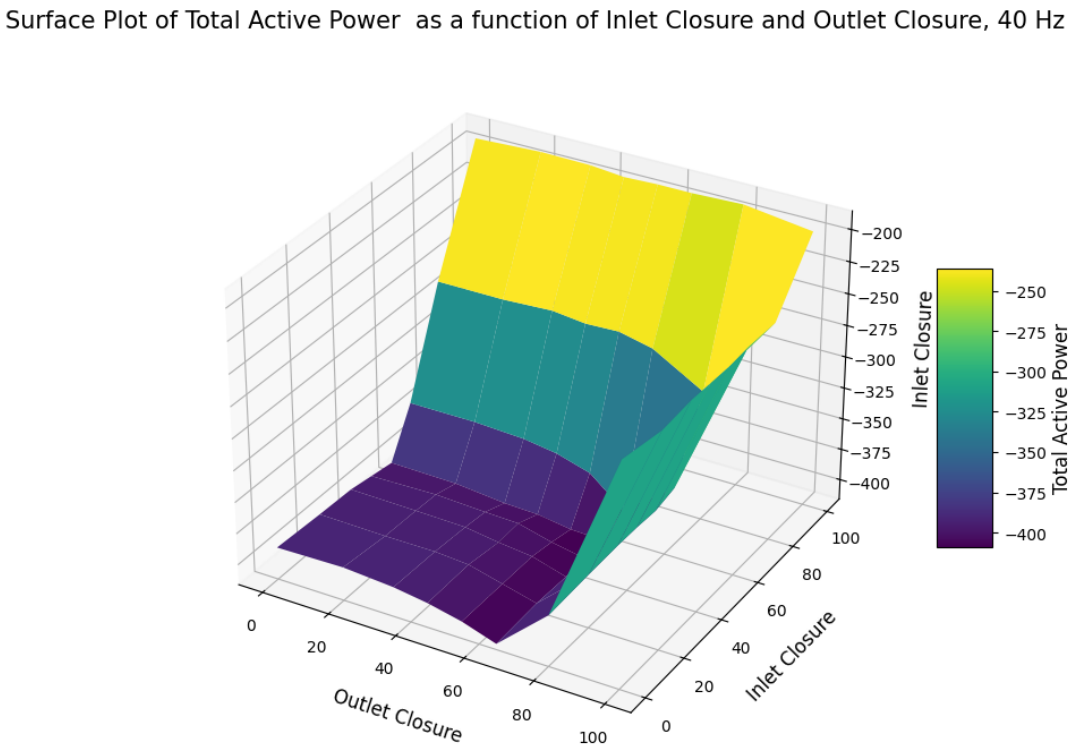


Figure 6.2: Active power as it relates to the closure of the inlet and outlet valves, at 40 Hz.

Figure 6.3 shows part of the spectra near 800 Hz, as the VFD's set frequency changes, while keeping the inlet and outlet valves open. This test has no fault injected (as the valves are both opened) and this spectra shows the characteristic of a normal behaviour of the system.

Figures 6.5 and 6.4 show also parts of the spectra, but considering fault injection by closing the valves. Now, we can see how the spectra differ while changing closures of, respectively, the inlet and outlet valves.

In Figure 6.5, part of the spectra near 8 kHz show that closing the inlet valve increases the vibration in this frequency range, which clearly indicates that different pressures in the suction can be detected in the vibration data.

In the same way, Figure 6.4 presents part of the spectra near 44 kHz, as the outlet closure changes, and we can observe that the magnitude of the vibrations in this frequency values increase with the closure of the valve, showing that pressure changes in the discharge can be detected in the vibration data. Another important aspect here is to acknowledge that 45 kHz is still an important frequency to observe.

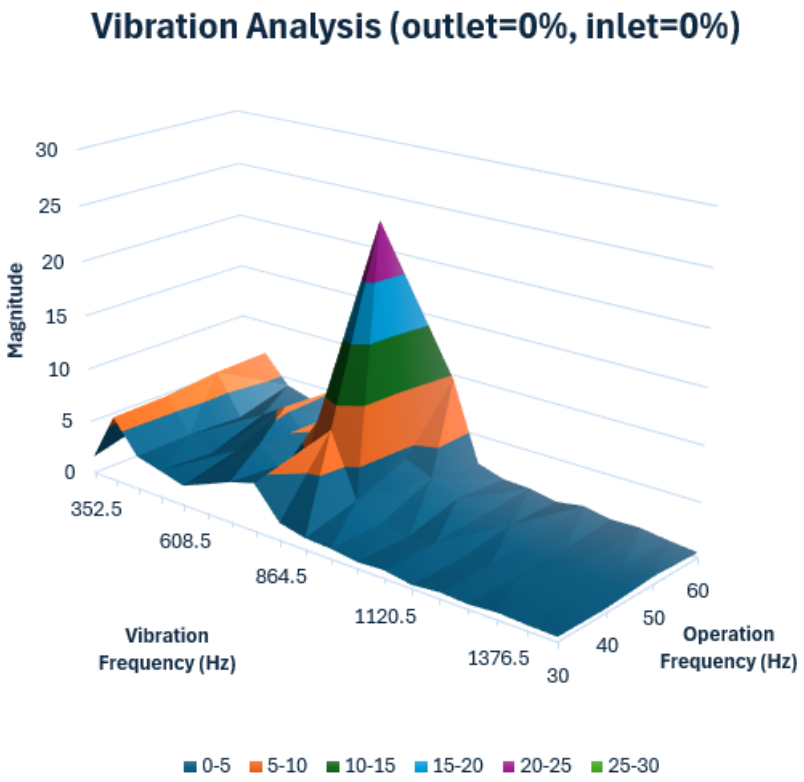


Figure 6.3: Part of vibration spectra taken using Brito's DAQ CPS, with VFD set frequency.

All data taken with the Raspberry Pi from the system was, previously stated stored in a Mongo Database, an example of an entry in that database's collection was 4.19. From the database the data was invariably turned into a CSV file, using techniques described in Chapter 5, and is saved in a file after the collection was extracted from the Mongo Database. Then it is immediately turned into Pandas dataframes. In Figure 6.6 can be seen a small part of a dataframe, part of the dataset taken from the system, using the data acquisition routine.

Several datasets were taken throughout the time of this thesis, at least two of them were then used to train supervised classification models according to the 9 classes established in Table 5.2 and are presented in this chapter. Their only difference, datawise, is their vibration sampling rate, which, in turn changes the maximum frequency for their spectra. Each one was evaluated the same multiple combinations of preprocessing methods and classification approaches to determine optimal strategies for condition monitoring. The results of the testing demonstrate how well the system is capable to identify different operating conditions by combining vibration analysis, energy

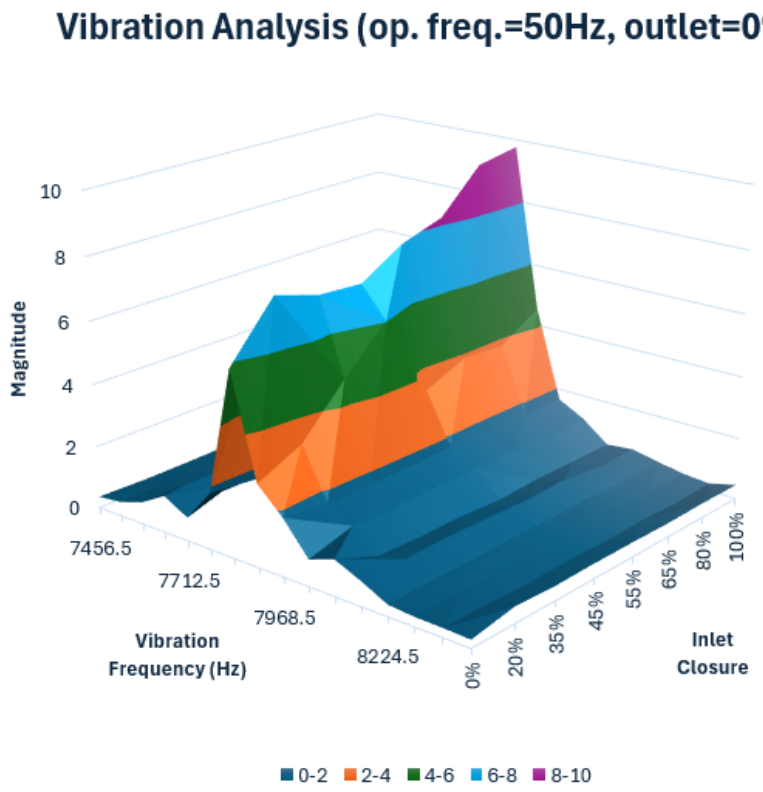


Figure 6.4: Part of vibration spectra taken using Brito's DAQ CPS, with changing inlet closure.

consumption metrics, and pressure data. Performance is assessed using the Macro F1-score for the test set, providing an idea of the system's classification capabilities.

For the case of the 24 kSa/s the testing results, after the training of the 36 models according to the combinations left from from Chapter 3, can be seen in Table 6.1 showing comprehensive performance metrics including the F1-scores, precision, and recall for the models of all tested combinations.The Macro F1-results distributions of testing are shown by Figure 6.7a.

Here we can see a definite trend in the results, with the best models being the ones that used the LSVM algorithm, with 1 Hz or 8 Hz banding. As can be seen in 6.7b, the Decision Tree algorithm and the 64 Hz Banding preprocessing parameter were lowering the distributions of the results and taking them out shot the results way up.

The test results for the best models created from this dataset were quite good, still, wanting to have access to the higher frequency levels described as important by [11, 43],the DAQ CPS was modified to increase its effective sampling speed.

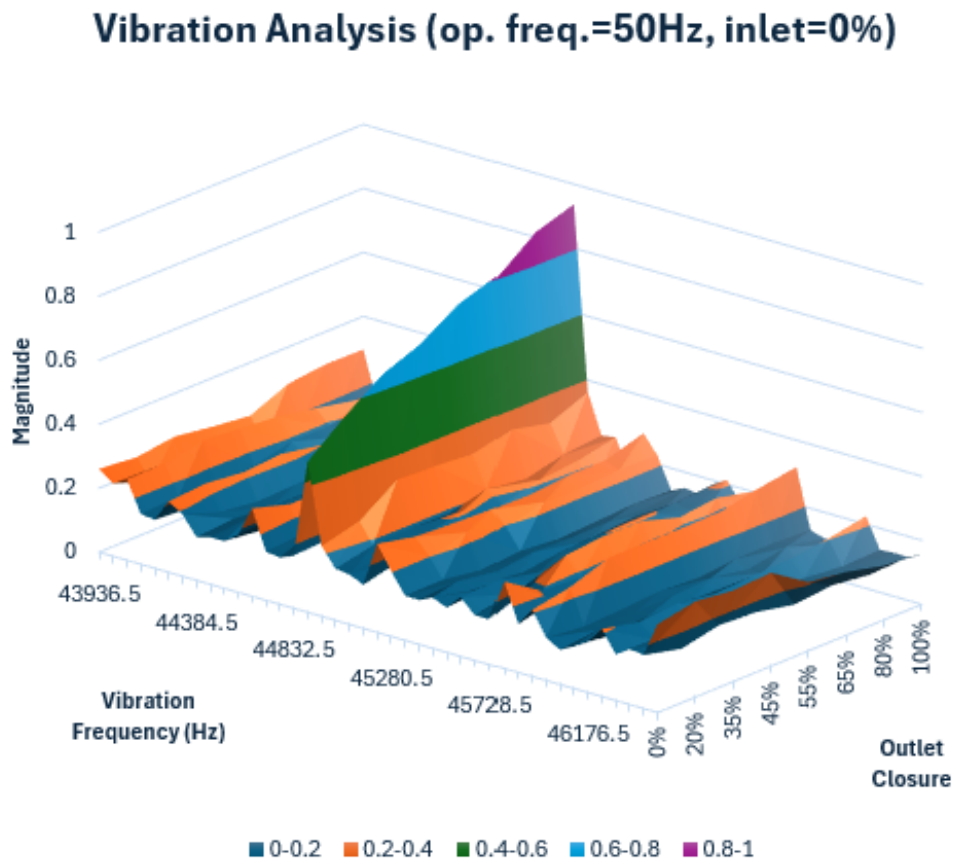
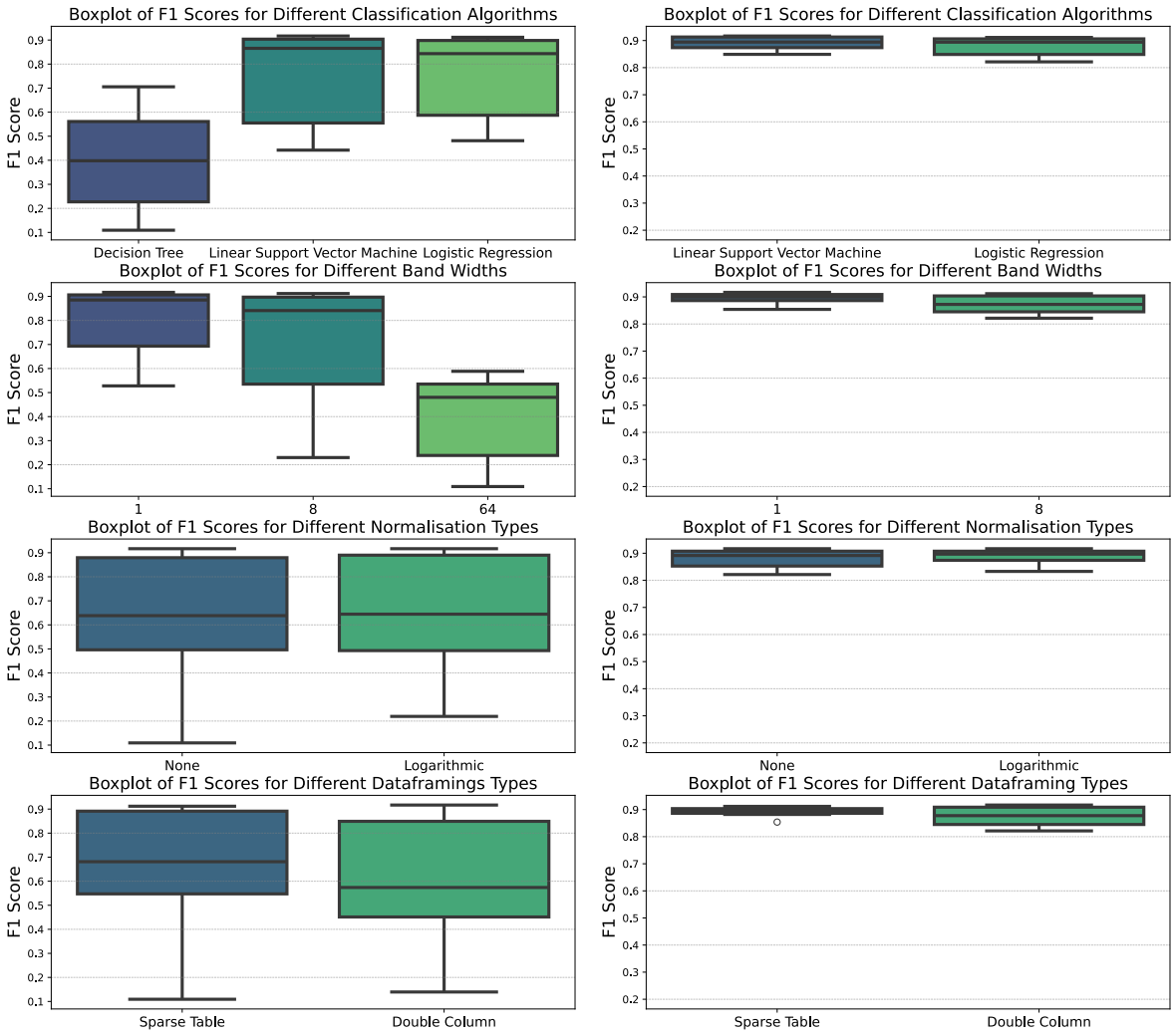


Figure 6.5: Part of vibration spectra taken using Brito's DAQ CPS, with changing outlet closure.

The second phase of experimentation leveraged an increased sampling rate of 100 kSa/s, more than quadrupling the temporal resolution of the vibration data capture compared to the initial phase. This substantial increase in sampling rate allowed for the capture of higher frequency components and finer temporal details in the vibration signals, potentially enabling better detection of subtle system behaviours. Even though some considerations were taken regarding the poor results given by both the 64 Hz-banding and the DT algorithm it was ultimately decided to keep them in the training for this new experiment, to see if the added frequency range would change their poor

	inlet_closure	outlet_closure	set_frequency	pressure	measured_freq	Current_I1	Total_active_power	vibration_samples
0	0	0	50	111	48.768000	2.160159	-663.064758	0.031128404662013054|0.031631954014301 3|0.0378...
1	0	0	50	111	48.768000	2.144533	-659.064697	0.031128404662013054|0.031631954014301 3|0.0378...
2	0	20	50	111	47.413333	2.019531	-623.064148	0.021057451143860817|0.0239871833473444|0.0277...
3	0	20	50	111	47.413333	2.035157	-623.064148	0.021057451143860817|0.0239871833473444|0.0277...
4	0	35	50	184	46.736000	1.970703	-607.063904	0.019134813919663 43|0.018081940710544586|0.019...

Figure 6.6: Screenshot of an example of some rows and some features of a dataframe from data acquired from the system using a Data Acquisition Routine on the Raspberry Pi.



(a) (b)

Figure 6.7: Macro F1-score distributions per preprocessing method/algorithm (a) and after taking all models that had either 64 Hz banding or had been trained using a Decision Tree (b).

behaviour. The complete results of this phase are presented in Table 6.2, with the same metrics as before for best comparison.

The enhanced sampling rate led to improvements across most model configurations, with particularly notable advancements in classification accuracy and consistency. The 3 best-performing models from this dataset training gauntlet, all utilizing LSVM with 8 Hz banding all achieved an F1-score higher than the best from the first test. Across the board there were meaningful improvement over all the results from

Table 6.1: Results of the inference of the 77 sample test set of each of the 32 models trained, ordered by Macro F1-score, for a 24 kSa/s vibration acquisition.

Normalisation	Band Width	Dataframe	Algorithm Used	F1-Score	Precision	Recall
Logarithmic	1 Hz	Double Column	LSVM	0.916728	0.943378	0.908642
None	1 Hz	Double Column	LSVM	0.916728	0.943378	0.908642
None	8 Hz	Sparse Table	LSVM	0.911813	0.939153	0.908642
Logarithmic	8 Hz	Sparse Table	LR	0.911424	0.935225	0.908642
None	1 Hz	Double Column	LR	0.906177	0.925168	0.902794
Logarithmic	1 Hz	Double Column	LR	0.906177	0.925168	0.902794
Logarithmic	8 Hz	Sparse Table	LSVM	0.900905	0.934584	0.896296
None	8 Hz	Sparse Table	LR	0.895503	0.915697	0.897531
Logarithmic	1 Hz	Sparse Table	LR	0.892472	0.934428	0.875309
None	1 Hz	Sparse Table	LSVM	0.887775	0.946876	0.864198
Logarithmic	1 Hz	Sparse Table	LSVM	0.882071	0.930909	0.864198
None	1 Hz	Sparse Table	LR	0.853899	0.880844	0.85835
None	8 Hz	Double Column	LSVM	0.849038	0.890729	0.850617
Logarithmic	8 Hz	Double Column	LSVM	0.849038	0.890729	0.850617
Logarithmic	8 Hz	Double Column	LR	0.832908	0.861728	0.834893
None	8 Hz	Double Column	LR	0.821357	0.852998	0.822547
Logarithmic	1 Hz	Sparse Table	DT	0.705524	0.74758	0.719493
None	1 Hz	Sparse Table	DT	0.656643	0.731614	0.673489
None	1 Hz	Double Column	DT	0.620021	0.637871	0.638077
None	64 Hz	Sparse Table	LR	0.588454	0.611781	0.582521
Logarithmic	64 Hz	Sparse Table	LR	0.583556	0.644589	0.566147
Logarithmic	64 Hz	Sparse Table	LSVM	0.564974	0.586488	0.578558
Logarithmic	8 Hz	Sparse Table	DT	0.541492	0.552497	0.560884
Logarithmic	1 Hz	Double Column	DT	0.527851	0.538777	0.55848
None	64 Hz	Sparse Table	LSVM	0.525394	0.566843	0.534113
None	8 Hz	Sparse Table	DT	0.516192	0.528171	0.539961
None	64 Hz	Double Column	LR	0.489667	0.514506	0.491163
Logarithmic	64 Hz	Double Column	LR	0.481405	0.506173	0.485315
None	64 Hz	Double Column	LSVM	0.478919	0.512103	0.491878
Logarithmic	64 Hz	Double Column	LSVM	0.442144	0.492969	0.45731
Logarithmic	8 Hz	Double Column	DT	0.279759	0.296216	0.277583
Logarithmic	64 Hz	Double Column	DT	0.244913	0.236752	0.263158
None	8 Hz	Double Column	DT	0.229748	0.248848	0.235932
Logarithmic	64 Hz	Sparse Table	DT	0.219354	0.232379	0.223067
None	64 Hz	Double Column	DT	0.139388	0.161199	0.125211
None	64 Hz	Sparse Table	DT	0.109177	0.144372	0.104808

the 24 kSa/s sampling rate. This improvement suggests that the higher sampling rate captures additional relevant information for fault classification.

All model combinations showed improved performance, with even previously lower-performing configurations achieving better results. The higher sampling rate appeared to provide more stable and reliable feature extraction, leading to more consistent classification outcomes.

Regarding preprocessing approaches, 8 Hz banding emerged as the optimal choice, outperforming both finer (1 Hz) and coarser (64 Hz) banding options. The superiority of 8 Hz banding suggests it captures the most relevant frequency information

Table 6.2: Results of the inference of the 77 sample test set of each of 34 the models trained, ordered by Macro F1-score, using 100 kSa/s vibration acquisition.

Normalisation	Band Width	Dataframing	Algorithm Used	F1-Score	Precision	Recall
None	8 Hz	Sparse Table	LSVM	0.931278	0.95202	0.922222
Logarithmic	8 Hz	Sparse Table	LSVM	0.929798	0.95	0.922222
None	8 Hz	Double Column	LSVM	0.924074	0.948052	0.921637
Logarithmic	8 Hz	Double Column	LSVM	0.913973	0.929533	0.921637
None	1 Hz	Double Column	LSVM	0.911232	0.967078	0.88642
Logarithmic	1 Hz	Double Column	LSVM	0.911232	0.967078	0.88642
Logarithmic	1 Hz	Double Column	LR	0.906904	0.940946	0.897531
None	1 Hz	Double Column	LR	0.904265	0.952121	0.88642
None	1 Hz	Sparse Table	LR	0.900121	0.955967	0.875309
Logarithmic	8 Hz	Sparse Table	LR	0.898601	0.921296	0.911111
None	8 Hz	Sparse Table	LR	0.898601	0.921296	0.911111
None	1 Hz	Sparse Table	LSVM	0.895062	0.961686	0.864198
Logarithmic	1 Hz	Sparse Table	LSVM	0.895062	0.961686	0.864198
Logarithmic	1 Hz	Sparse Table	LR	0.891735	0.95194	0.864198
None	8 Hz	Double Column	LR	0.874386	0.895345	0.888304
Logarithmic	8 Hz	Double Column	LR	0.874386	0.895345	0.888304
None	1 Hz	Sparse Table	DT	0.813504	0.876515	0.796296
Logarithmic	1 Hz	Sparse Table	DT	0.791945	0.809436	0.796946
None	8 Hz	Sparse Table	DT	0.785567	0.815304	0.782781
None	1 Hz	Double Column	DT	0.769802	0.814033	0.770435
Logarithmic	64 Hz	Sparse Table	LSVM	0.769489	0.773713	0.778168
None	64 Hz	Double Column	LSVM	0.767142	0.773753	0.766992
None	64 Hz	Sparse Table	LSVM	0.76606	0.774375	0.77167
Logarithmic	64 Hz	Double Column	LSVM	0.758198	0.760614	0.761144
Logarithmic	64 Hz	Sparse Table	LR	0.7266	0.769957	0.727875
Logarithmic	8 Hz	Sparse Table	DT	0.691182	0.710442	0.68681
Logarithmic	1 Hz	Double Column	DT	0.676589	0.735282	0.670435
None	64 Hz	Sparse Table	LR	0.667893	0.688321	0.671085
Logarithmic	64 Hz	Double Column	LR	0.65605	0.656602	0.669786
None	64 Hz	Double Column	LR	0.634915	0.631662	0.652827
Logarithmic	8 Hz	Double Column	DT	0.520931	0.577425	0.492398
None	8 Hz	Double Column	DT	0.49537	0.520916	0.495452
Logarithmic	64 Hz	Double Column	DT	0.460447	0.468318	0.479467
Logarithmic	64 Hz	Sparse Table	DT	0.356716	0.357143	0.373619
None	64 Hz	Sparse Table	DT	0.304575	0.316122	0.318713
None	64 Hz	Double Column	DT	0.226857	0.225669	0.241196

while effectively reducing noise. Additionally, LSVM continued to demonstrate superior performance, particularly when combined with appropriate preprocessing.

One of the most meaningful takeaways of these two experiments is that the normalisation step is, by and large, unnecessary. Though it can be seen that it can sometimes have an effect, it is ultimately a superfluous processing step that only costs computational resources without adding much. This was proven time and time again, ever since Chapter 3.

The comprehensive results from this higher sampling rate phase strongly validate the effectiveness of the automated predictive maintenance system. The improvements

in both absolute performance and consistency demonstrate the importance of appropriate sampling rates in vibration-based condition monitoring. Furthermore, the results reinforce the significance of choosing appropriate preprocessing methods and classification algorithms, while suggesting that higher sampling rates may make the system more robust to these choices.

6.3 Inference on the Raspberry Pi

From an implementation perspective, improved performance must be weighed against the increased computational requirements of higher sampling rates. The results suggest that 100 kSa/s might represent a practical optimum for industrial applications, offering significant performance improvements while maintaining reasonable computational demands. This balance between performance and resource requirements is crucial for practical industrial deployment.

After these models were created, code was developed to infer using these models on the Raspberry Pi. While the code to develop all of those preprocessing techniques is one thing when one has a dataset it is an entirely different thing when it is done to a single sample. Of course, with this in mind several assets had to be stored, in pickle format, about exactly what was used during training to be applied the same way in runtime. One example is the number of peaks to take into account: as previously told the half of the number of peaks of the sample with the least peaks is going to be the peak number for all of that banding, be it for training, be it for inference, or else there would just be an error due to having too large or too little features entering the inference model. For this reason several pickles are saved as assets to use in inference, especially because the inference function is designed to infer using every single one of the 34 models, and they differ widely in the finished data that enters the model.

As described in Table 6.3, using legacy raw data, the dataset from the 100 kSa/s data acquisition, a test was made in which the raw data was inserted in the Pi. Although they were loaded from their file, they are prepared to become **exactly** as they were the

time they were firstly acquired from the Raspberry Pi. And the data was all prepared using the functions that had been designed to work on the Pi for only one sample according to the specifications of the model.

Table 6.3: Results of the inference of raw legacy data in the Raspberry Pi using its specific data preparation.

Dataframing	Double Column
Normalisation	None
Band Width	8 Hz
Algorithm Used	LSVM
Population	512
Accuracy (%)	98.828125
Macro Recall (%)	98.784722
Macro F1-Score (%)	98.870081
Average Inference Time (ms)	213.955
Std. Deviation (ms)	10.436
Std. Deviation (% of the average)	4.8777

Using this model for this test, out of the 512 samples, 6 were misclassified, giving a Macro F1-score of 98.9%, with an average inference time which was little less than 214 milliseconds with a standard deviation of less than 5%. Which, of course, proves the whole system as a concept.

It is out of the scope of this thesis to reach a product which at any point in time takes all the measurements from the automation system and infers in real time. The thesis' initial objective as can be seen in 1.2, was to have a proof of concept. Nevertheless, code was developed in that way of measuring data and inferring into one of the classes described in 5.3 using any model that's been trained now or can be trained in the future. In appendix B is code for the Raspberry Pi that loads the required assets, takes all the measurements, and infers using the desired model.

"Resumindo e baralhando.."

Jorge Semião

7

Conclusions and future work

In this thesis, the development of an automation system for predictive maintenance was presented, tailored for electric machines used in pumping systems. The work integrates Industry 4.0 technologies such as IoT, ML algorithms, and advanced sensor systems to enhance the reliability and efficiency of industrial operations.

An important part of the work carried out in this thesis was the development of a test bench for an automation pumping system with predictive maintenance features. The hardware involved includes an electric pump, a water reservoir, an automation and control board, an energy meter, vibration sensors, pressure sensor, motorized valves, and temperature sensors. Also, it is important to state that with this test set-up, the tests to assess the predictive maintenance strategy can be implemented automatically, and including fault-injection behaviours by controlling the motorized valves.

However, the work carried out in this thesis is not just limited to the test bench. The thesis bridges the gap between theoretical models of predictive maintenance and their practical implementation in industrial environments, emphasizing the importance of automation and data-driven decision-making.

The core achievements of the work include the successful development and deployment of a system capable of real-time data acquisition, advanced vibration analysis, and fault prediction using ML. In this regard, a detailed analysis was made for define the best procedures to data processing, prior to data analysis. Moreover, automation-based sensors were integrated with electronic-based sensors in the same predictive maintenance system, all of this contributing to a complete data collection.

Finally, ML algorithms was used to demonstrate that it is possible to implement a predictive maintenance system with a standard automation pumping system, by using the models to predict faults induced in the test pump. And the results showed the effectiveness of the method and the predictive maintenance strategy. The developed system not only reduces unexpected failures but also aligns with modern demands for sustainability and operational efficiency. By combining data acquisition, real-time analysis, and predictive modelling, the research offers a comprehensive approach that can be adapted across various industries reliant on electric machines.

7.1 Analysis of the Work Done

It is true that, while reasonably reliable, a Raspberry Pi isn't up to the standards of reliability of an industrial setting. The proposed system was designed to not be affected in its PLC process routines — which are highly reliable — regardless of the state of the Raspberry Pi. Still, the beauty of the proposed system is that the processing unit — - the Raspberry Pi, in this case — doesn't have to be anywhere near PLC or harsh conditions, just needs to be connected to the Local Area Network (LAN). Although, it can be replaced by an online processing unit, for example, the Raspberry Pi allows for the system to work locally, when internet is not available, and allows to maintain a

low budget in the system.

These findings have important implications for practical implementation, suggesting that while higher sampling rates provide better performance, they must be balanced against computational and storage requirements. The results also indicate that with proper configuration, the system can achieve highly reliable classification of machine states, making it suitable for industrial predictive maintenance applications where reliable fault detection is crucial for maintaining operational efficiency.

The research findings yielded several additional technical insights. Notably, frequency components above 20 kHz were found to contain valuable diagnostic information, challenging traditional assumptions about relevant frequency ranges. The data suggested that even low-magnitude frequency bands may hold important diagnostic value, though this requires further investigation. Interesting relationships were observed between VFD frequency setpoint and vibration spectral components. Importantly, normalization of vibration data appeared to have minimal impact on classification performance, suggesting this preprocessing step may be unnecessary.

7.2 Future Work

Several promising directions for future research and development emerged from this work:

7.2.1 System Enhancement

- The integration of another pressure sensor at the suction side of the system could help establishing a ground-truth with a deeper understanding of what's happening in the system;
- Increasing of the dataset, not necessarily much more, but some finer granularity in the valve closures, would be important to improve the test bench's reliability;
- Development of more sophisticated fault injection methods beyond valve closure, to simulate a wider range of real-world failure scenarios;

- Extension of the current 9-class classification system to identify more specific fault-types and their severity levels;
- Exploration of system scalability for n-pump configurations, investigating the interactions and fault patterns in multi-pump systems;
- Investigation of spectral behavior variations under different VFD frequency set-points to better understand system dynamics;
- Detailed analysis of alternative dataframing approaches to capture smaller frequency components that may contain crucial diagnostic information.

7.2.2 Technical Improvements

- It was not this thesis' objective to make the best possible model and so very little though was given to fine tuning the model: a more thorough grid search would very much help to improve system's reliability;
- Exploration of deep learning approaches, particularly for processing raw vibration signals without extensive feature engineering;
- Investigation of alternative preprocessing techniques that could improve classification performance while reducing computational overhead;
- Implementation of transfer learning techniques to adapt the system to different pump types and configurations.

7.2.3 Validation and Testing

- Extended testing with different types and sizes of pumps, to validate the system's generalizability, especially in non-laboratorial and industrial pumps;
- Long-term reliability studies under various operational conditions:
- Comparative studies with other predictive maintenance approaches.
- Integrate data collection and analysis mechanisms into the automation system, to interact with the pumping systems to gather and assess vibrational data.

7.2.4 Cost-Benefit Analysis

- Detailed studies on the economic impact of implementing the system in various industrial settings;
- Analysis of potential energy savings through optimized maintenance scheduling;
- Investigation of the system's impact on equipment lifetime and reliability;
- Development of Return of Investmen (ROI) models for different implementation scenarios.

The implementation of these future work items could significantly enhance the system's capabilities and broaden its applicability across different industrial contexts. Particularly important would be the development of more robust hardware solutions and the validation of the system's effectiveness across a broader range of operational conditions and equipment types.

References

[1] Yang Lu. The current status and developing trends of industry 4.0: A review. *Information Systems Frontiers*, 2021. doi: 10.1007/s10796-021-10221-w.

[2] Wei Zhang, Ming Li, Jian Wang, Yuxin Liu, and Xiaodong Zhang. Predictive maintenance in industry 4.0: A case study on the application of machine learning for fault detection. *Applied Sciences*, 12(16), 2022. ISSN 2076-3417. doi: 10.3390/app12168081. URL https://www.mdpi.com/2076-3417/12/16/8081.

[3] Mounia Achouch, Mariya Dimitrova, Khaled Ziane, Sasan Sattarpanah Karganroudi, Rizck Dhouib, Hussein Ibrahim, and Mehdi Adda. On predictive maintenance in industry 4.0: Overview, models, and challenges. *Applied Sciences*, 12(16), 2022. ISSN 2076-3417. doi: 10.3390/app12168081. URL https://www.mdpi.com/2076-3417/12/16/8081.

[4] Lei Chen, Lijun Wei, Yu Wang, Junshuo Wang, and Wenlong Li. Monitoring and predictive maintenance of centrifugal pumps based on smart sensors. *Sensors*, 22 (6), 2022. ISSN 14248220. doi: 10.3390/s22062106.

[5] Jorge Filipe, Ricardo J Bessa, Marisa Reis, Rita Alves, and Pedro Póvoa. Data-driven predictive energy optimization in a wastewater pumping station. *Applied Energy*, 252(June):113423, 2019. ISSN 0306-2619. doi: 10.1016/j.apenergy.2019.113423. URL https://doi.org/10.1016/j.apenergy.2019.113423.

[6] M.M. Hamasha, A.H. Bani-Irshid, S. Al Mashaqbeh, et al. Strategical selection of maintenance type under different conditions. *Scientific Reports*, 13:15560, 2023. doi: 10.1038/s41598-023-42751-5. Received: 24 April 2023; Accepted: 14 September 2023; Published: 20 September 2023.

[7] James J. Houle, Robert M. Roseen, Thomas P. Ballestero, Timothy A. Puls, and James Sherrard. Comparison of maintenance cost, labor demands, and system performance for LID and conventional stormwater management. *Journal of Environmental Engineering*, 139(7):932–938, 2013. ISSN 0733-9372. doi: 10.1061/(asce)ee.1943-7870.0000698.

[8] Yaguo Lei, Naipeng Li, Liang Guo, Ningbo Li, Tao Yan, and Jing Lin. Machinery health prognostics: A systematic review from data acquisition to RUL prediction. *Mechanical Systems and Signal Processing*, 104:799–834, 2018. ISSN 10961216. doi: 10.1016/j.ymssp.2017.11.016. URL https://doi.org/10.1016/j.ymssp.2017.11.016.

[9] N. R. Sakthivel, V. Sugumaran, and S. Babudevasenapati. Vibration based fault diagnosis of monoblock centrifugal pump using decision tree. *Expert Systems with*

Applications, 37(6):4040–4049, 2010. ISSN 09574174. doi: 10.1016/j.eswa.2009.10.002. URL http://dx.doi.org/10.1016/j.eswa.2009.10.002.

[10] Pablo Donolo, Guillermo Bossio, Cristian De Angelo, Guillermo García, and Marcos Donolo. Voltage unbalance and harmonic distortion effects on induction motor power, torque and vibrations. *Electric Power Systems Research*, 140: 866–873, 2016. ISSN 03787796. doi: 10.1016/j.epsr.2016.04.018. URL http://dx.doi.org/10.1016/j.epsr.2016.04.018.

[11] Abhimanyu Kapuria and Daniel G. Cole. Integrating survival analysis with bayesian statistics to forecast the remaining useful life of a centrifugal pump conditional to multiple fault types. *Energies*, 16(9), 2023. ISSN 19961073. doi: 10.3390/en16093707.

[12] Sánchez Ocaña Wilson, Christian Carvajal, Jose Poalacin, and Elizabeth Salazar. Detection of cavitation in centrifugal pump for vibration analysis. *Proceedings - 2018 4th International Conference on Control, Automation and Robotics, ICCAR 2018*, pages 460–464, 2018. doi: 10.1109/ICCAR.2018.8384720.

[13] Michael Short and John Twiddle. An industrial digitalization platform for condition monitoring and predictive maintenance of pumping equipment. *Sensors (Switzerland)*, 19(17), 2019. ISSN 14248220. doi: 10.3390/s19173781.

[14] Grundfos. GRUNDFOS. https://www.grundfos.com/, 2024. Accessed: 2024-08-30.

[15] SHNEIDER. SHNEIDER. https://www.se.com/ww/en/work/solutions/machine-control/pumping/, 2024. Accessed: 2024-08-30.

[16] Sérgio Duarte Lopes Brito. Sistema de hardware para monitorização não intrusiva de motores elétricos. 2023.

[17] Demet Özgür-Ünlüakın and Ayşe Karacaörenli. Analysis of reactive maintenance strategies on a multi-component system using dynamic bayesian networks. In *Proceedings of the International Symposium for Production Research*, pages 101–110, 2018. doi: doi.org/10.1007/978-3-319-92267-6_8.

[18] Yongyi Ran, Xin Zhou, Pengfeng Lin, Yonggang Wen, and Ruilong Deng. A survey of predictive maintenance: Systems, purposes and approaches. *IEEE COMMUNICATIONS SURVEYS & TUTORIALS*, XX(Xx):1–36, 2019. URL http://arxiv.org/abs/1912.07383.

[19] Nicholas Renzetti, Rossi Setchi, and Nuno Lagos. Manufacturing intelligence for industrial engineering: Methods for system self-organization and adaptation. *Applied Mechanics and Materials*, 811:363–369, 2015. doi: 10.1080/21693277.2015.1074124.

[20] Hilary A. Johnson, Kevin P. Simon, and Alexander H. Slocum. Data analytics and pump control in a wastewater treatment plant. *Applied Energy*, 299(June): 117289, 2021. ISSN 03062619. doi: 10.1016/j.apenergy.2021.117289. URL https://doi.org/10.1016/j.apenergy.2021.117289.

[21] Friedrich Wilhelm Hennecke. Life cycle costs of pumps in chemical industry. *Chemical Engineering and Processing: Process Intensification*, 38(4-6):511–516, 1999. ISSN 02552701. doi: 10.1016/S0255-2701(99)00047-1.

[22] Ramez Abdalla, Hanin Samara, Nelson Perozo, Carlos Paz Carvajal, and Philip Jaeger. Machine learning approach for predictive maintenance of the electrical submersible pumps (ESPs). *ACS Omega*, 7(21):17641–17651, 2022. ISSN 24701343. doi: 10.1021/acsomega.1c05881.

[23] Joon Hyung Kim, Him Chan Lee, Jin Hyuk Kim, Sung Kim, Joon Yong Yoon, and Young Seok Choi. Design techniques to improve the performance of a centrifugal pump using CFD. *Journal of Mechanical Science and Technology*, 29(1):215–225, 2015. ISSN 19763824. doi: 10.1007/s12206-014-1228-6.

[24] Ryutaro Ujiie, Asuma Ichinose, Yohei Nakamura, Kazuyoshi Miyagawa, and Takeshi Sano. Influence of clearance flow on disk friction loss in low specific speed hydraulic machines. *IOP Conference Series: Earth and Environmental Science*, 240(3), 2019. ISSN 17551315. doi: 10.1088/1755-1315/240/3/032043.

[25] Parasuram P. and Alexander G. Fault diagnosis of centrifugal pumps using motor electrical signals. *Centrifugal Pumps*, (June), 2012. doi: 10.5772/26439.

[26] Yavor Lozanov, Svetlana Tzvetkova, and Angel Petleshkov. Study of the effectiveness of a variable frequency drive of an induction motor. *2019 11th Electrical Engineering Faculty Conference, BulEF 2019*, pages 1–6, 2019. doi: 10.1109/BulEF48056.2019.9030775.

[27] Ezz Badry, Salah Kamel, and Loai S. Nasrat. Coordination study of a realistic oil production area including variable frequency drives. *Proceedings of 2018 International Conference on Innovative Trends in Computer Engineering, ITCE 2018*, 2018-March:368–373, 2018. doi: 10.1109/ITCE.2018.8316652.

[28] Yuji Akiyama. Induction motor residual voltage. *Conference Record - IAS Annual Meeting (IEEE Industry Applications Society)*, pages 24–29, 1990. ISSN 01608592. doi: 10.1109/ias.1990.152160.

[29] Shivam Chauhan and Shashi Bhusan Singh. Effects of voltage unbalance and harmonics on 3-phase induction motor during the condition of undervoltage and overvoltage. *2019 6th International Conference on Signal Processing and Integrated Networks, SPIN 2019*, pages 1141–1146, 2019. doi: 10.1109/SPIN.2019.8711753.

[30] Piotr Gnaciński. Windings temperature and loss of life of an induction machine under voltage unbalance combined with over- or undervoltages. *IEEE Transactions on Energy Conversion*, 23(2):363–371, 2008. ISSN 08858969. doi: 10.1109/TEC.2008.918596.

[31] P. Caramia, G. Carpinelli, P. Verde, G. Mazzanti, A. Cavallini, and G. C. Montanari. An approach to life estimation of electrical plant components in the presence of harmonic distortion. *Proceedings of International Conference on Harmonics and Quality of Power, ICHQP*, 3(5):887–892, 2000. ISSN 21640610. doi: 10.1109/ICHQP.2000.896846.

[32] Carsten Kallesøe, Vincent Cocquempot, and Roozbeh Izadi-Zamanabadi. Model based fault detection in a centrifugal pump application. *EEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY*, 14(2):204–215, 2006.

[33] PHOENIX CONTACT. PHOENIX CONTACT. `https://www.phoenixcontact.com/`, 2024. Accessed: 2024-09-30.

[34] Marisa Ehemann, Frank Trankle, and Nicolaj C. Stache. AI-based cavitation detection in process valves. *IEEE International Conference on Industrial Informatics (INDIN)*, 2023-July, 2023. ISSN 19354576. doi: 10.1109/INDIN51400.2023.10218193.

[35] Shreyas J. Upasane, Hani Hagras, Mohammad Hossein Anisi, Stuart Savill, Ian Taylor, and Kostas Manousakis. A type-2 fuzzy-based explainable AI system for predictive maintenance within the water pumping industry. *IEEE Transactions on Artificial Intelligence*, 5(2):490–504, 2024. doi: 10.1109/TAI.2023.3279808. URL `https://doi.org/10.1109/TAI.2023.3279808`.

[36] Kanae Takahashi, Kouji Yamamoto, Aya Kuchiba, and Tatsuki Koyama. Confidence interval for micro-averaged f 1 and macro-averaged f 1 scores. *Applied Intelligence*, 52(5):4961–4972, 2022.

[37] Vladimir N. Vapnik. *Statistical Learning Theory*. 1988. ISBN 978-0-471-03003-4.

[38] Kristin Hodgkinson. Health & safety: Staying safe when working with pumps. *World Pumps*, 2014(12):34–36, 2014. ISSN 02621762. doi: 10.1016/s0262-1762(14)70313-x.

[39] P. Wisner, N. Kouwen, and F. Mohsen. Removal of air from water lines by hydraulic means. *Journal of Hydraulic Engineering*, 101:243–257, 1975.

[40] Mikhail Tsypkin. Vibration of induction motors operating with variable frequency drives - A practical experience. *2014 IEEE 28th Convention of Electrical and Electronics Engineers in Israel, IEEEI 2014*, pages 1–5, 2014. doi: 10.1109/EEEI.2014.7005894.

[41] J. Michael Jacob. *Industrial Control Electronics - Applications and Design*. 1989.

[42] Baoling Cui, Zhe Lin, Zuchao Zhu, Huijie Wang, and Guangfei Ma. Influence of opening and closing process of ball valve on external performance and internal flow characteristics. *Experimental Thermal and Fluid Science*, 80:193–202, 2017. ISSN 08941777. doi: 10.1016/j.expthermflusci.2016.08.022. URL `http://dx.doi.org/10.1016/j.expthermflusci.2016.08.022`.

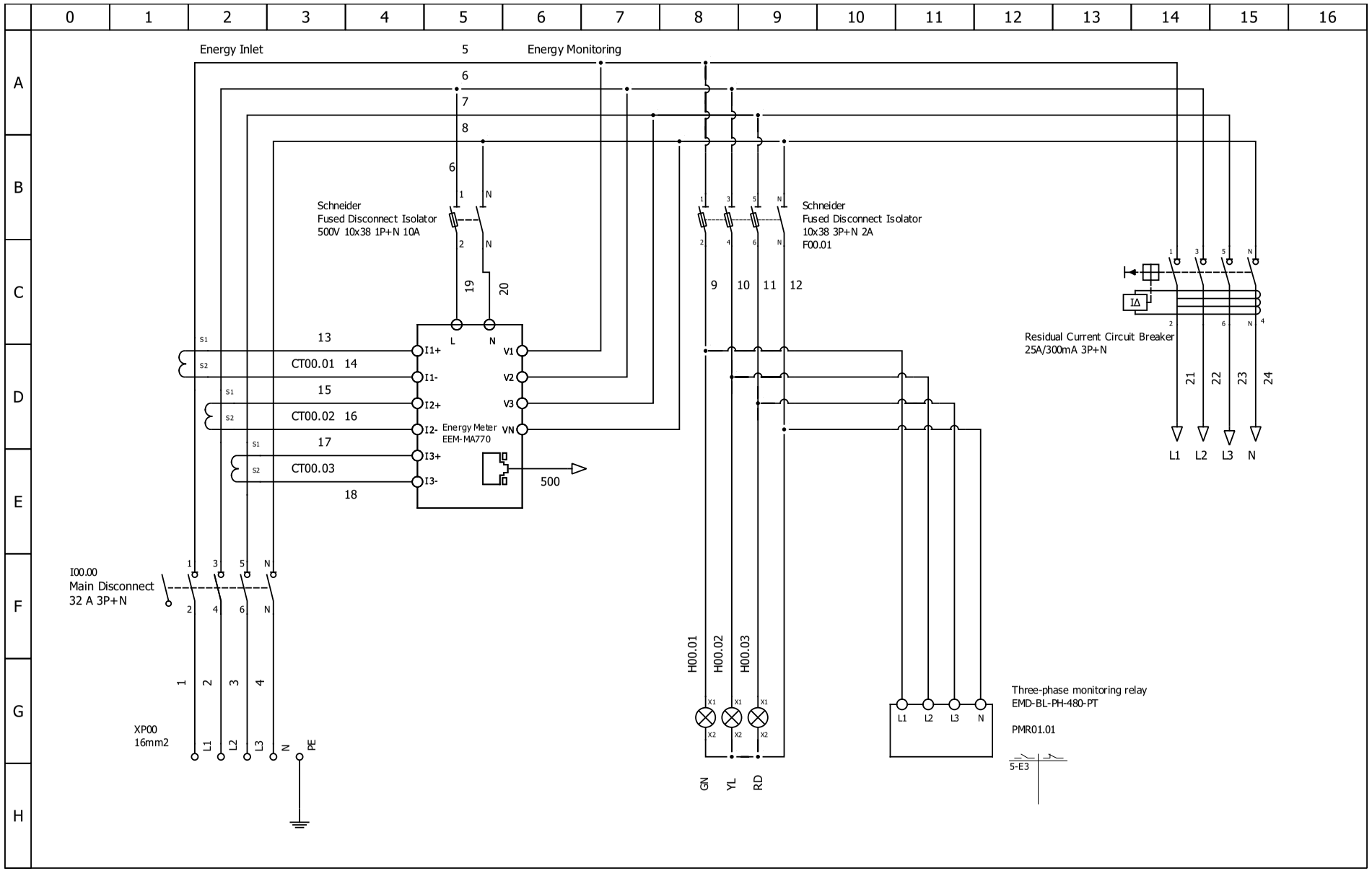
[43] Gustav Osswald. Vibration analysis for predictive maintenance of a rotary pump: Optimal accelerometer configuration based on vibration analysis for cavitation detection of a bi-winged positive displacement pump. page 98, 2019.

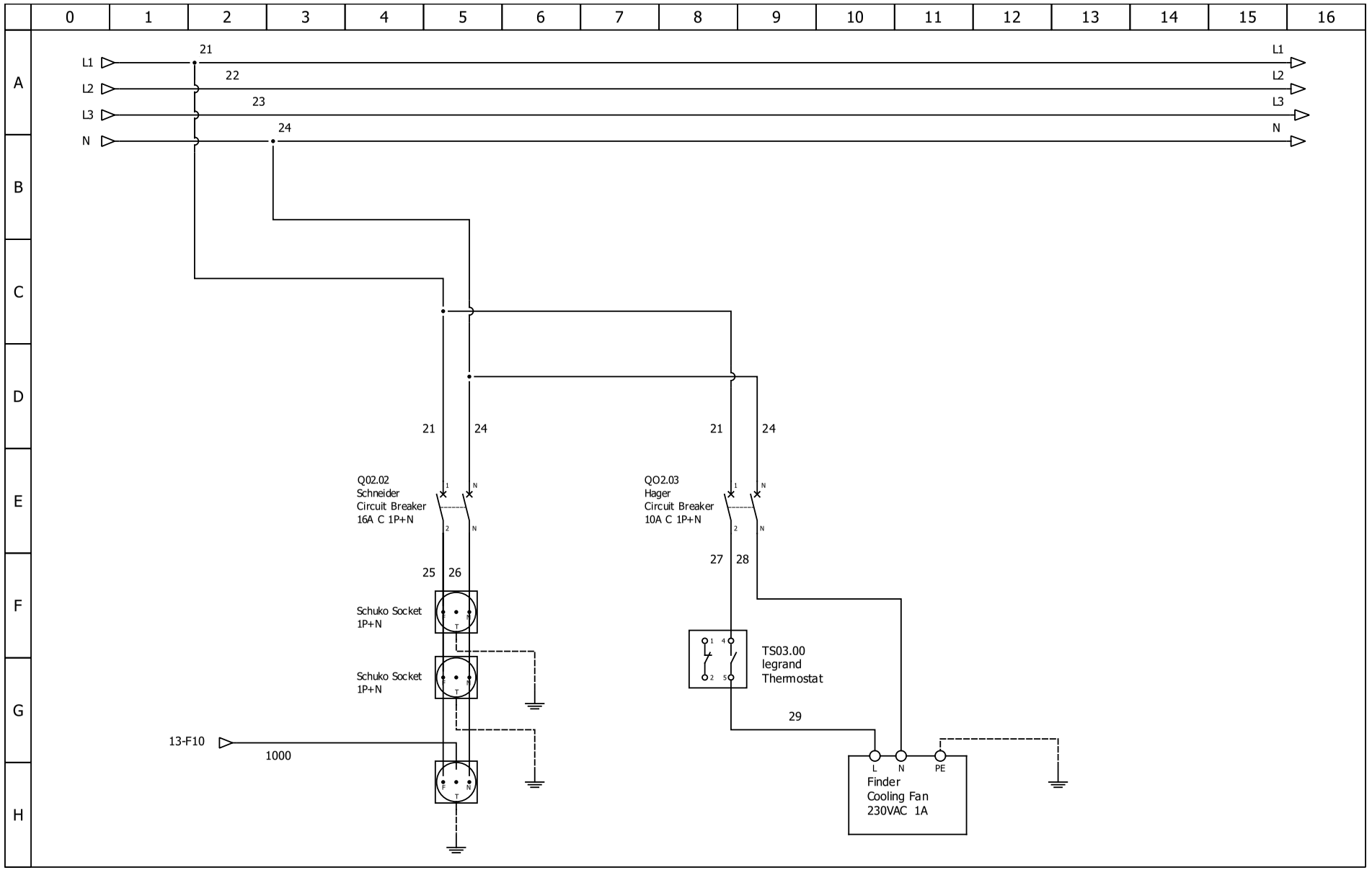
Appendices

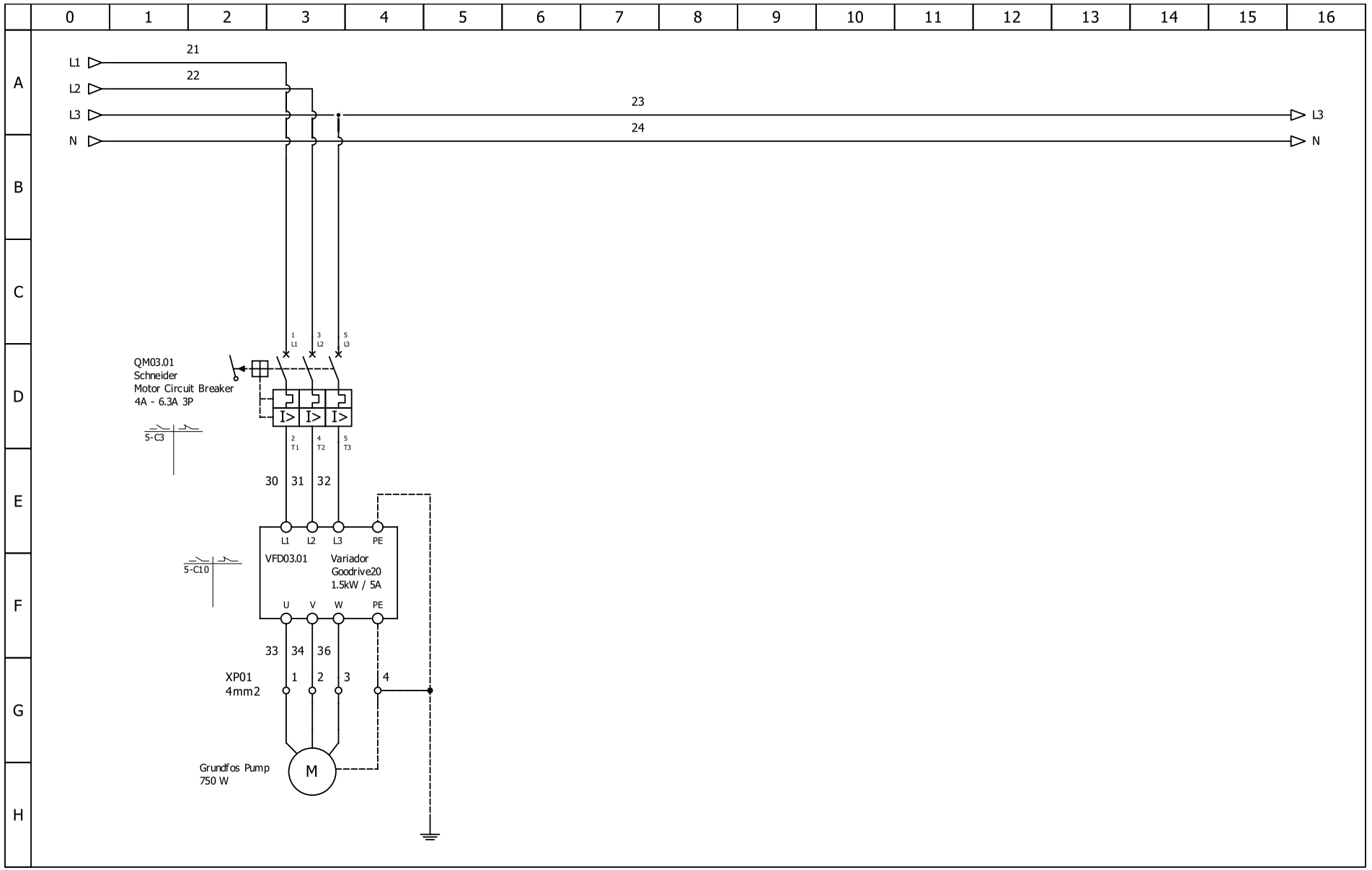
A

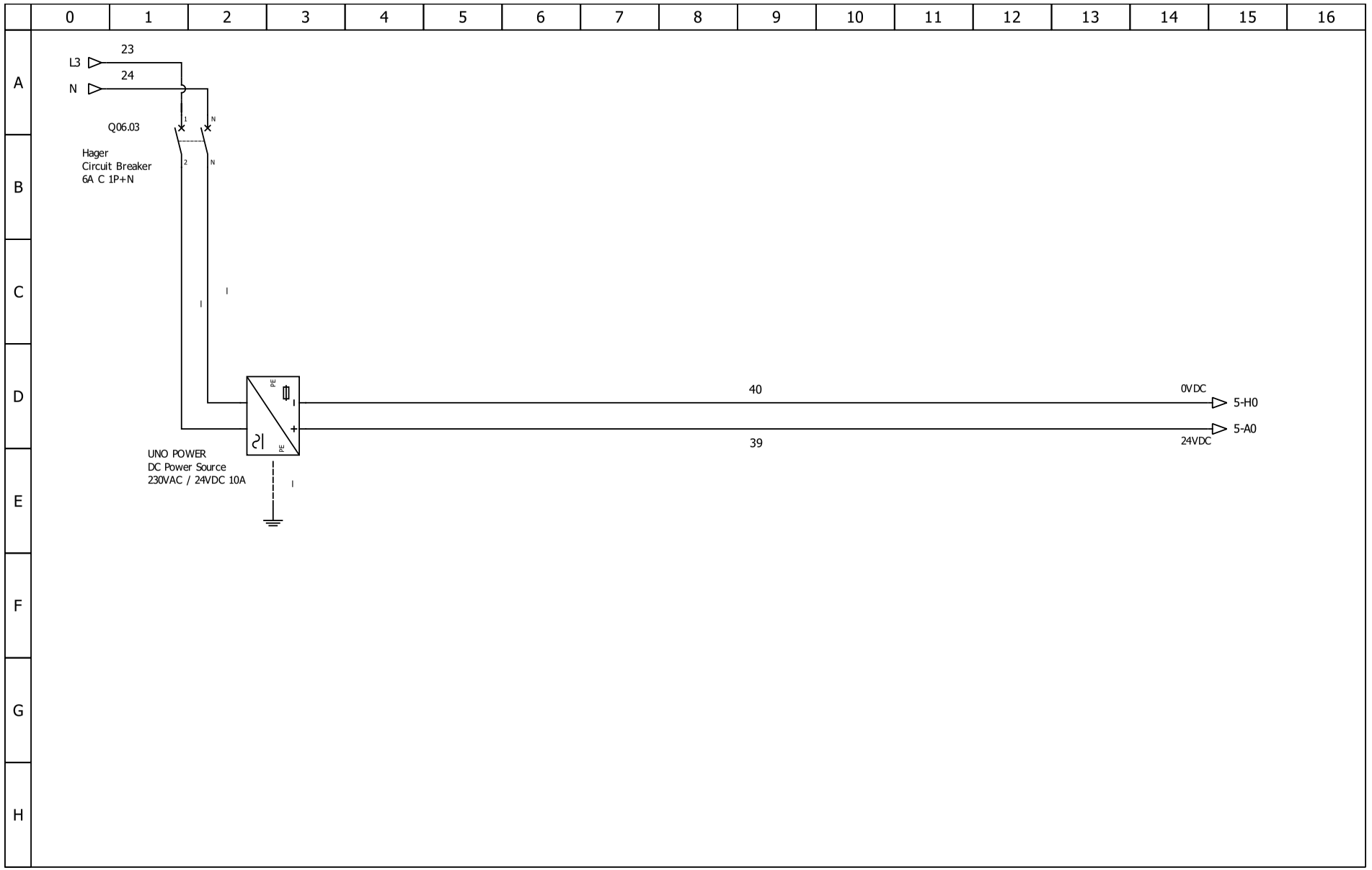
Automation Panel Project

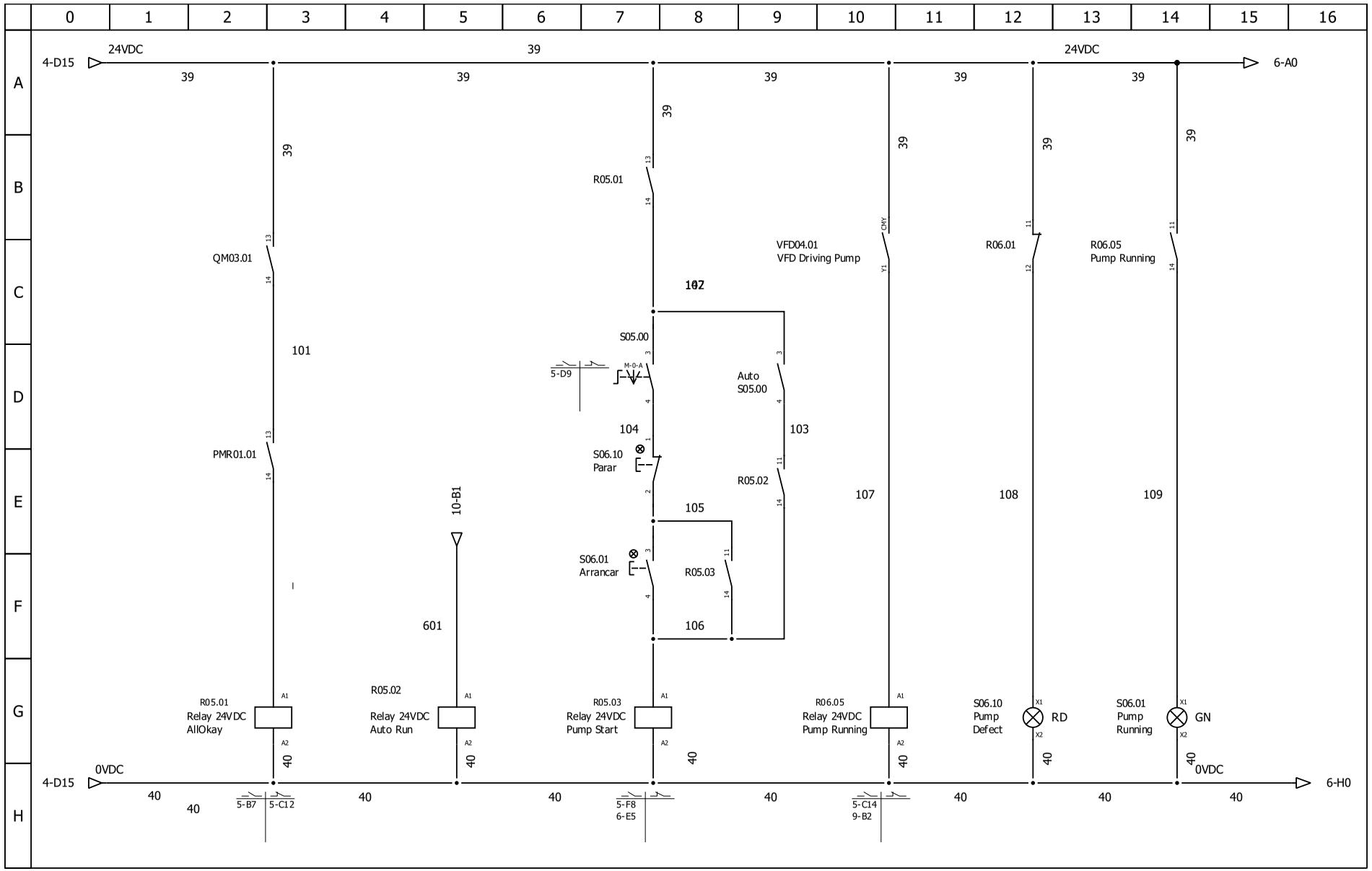
Here is the detailed electrical project documentation for the automation control panel that was designed and built as part of this thesis work. The following schematics show the complete wiring diagrams and component connections that were implemented.

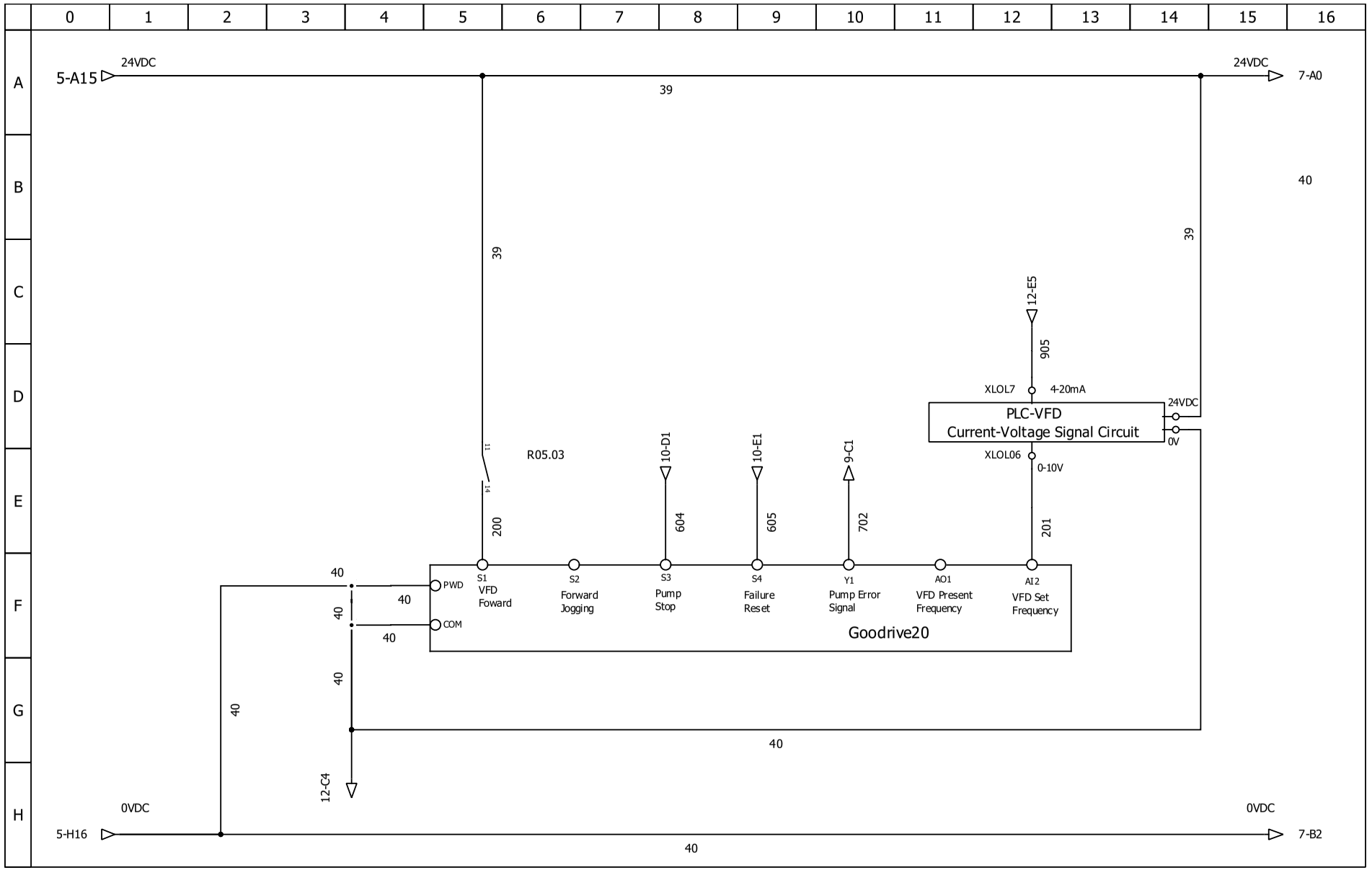

Energy Inlet
Energy Monitoring
Schneider
Fused Disconnect Isolator
500V 10x38 1P+N 10A
Schneider
Fused Disconnect Isolator
10x38 3P+N 2A
F00.01
Residual Current Circuit Breaker
25A/300mA 3P+N
CT00.01
CT00.02
CT00.03
Energy Meter
EEM-MA770
I00.00
Main Disconnect
32 A 3P+N
XP00
16mm2
H00.01
H00.02
H00.03
GN
YL
RD
Three-phase monitoring relay
EMD-BL-PH-480-PT
PMR01.01
5-E3
L1
L2
L3
N
PE

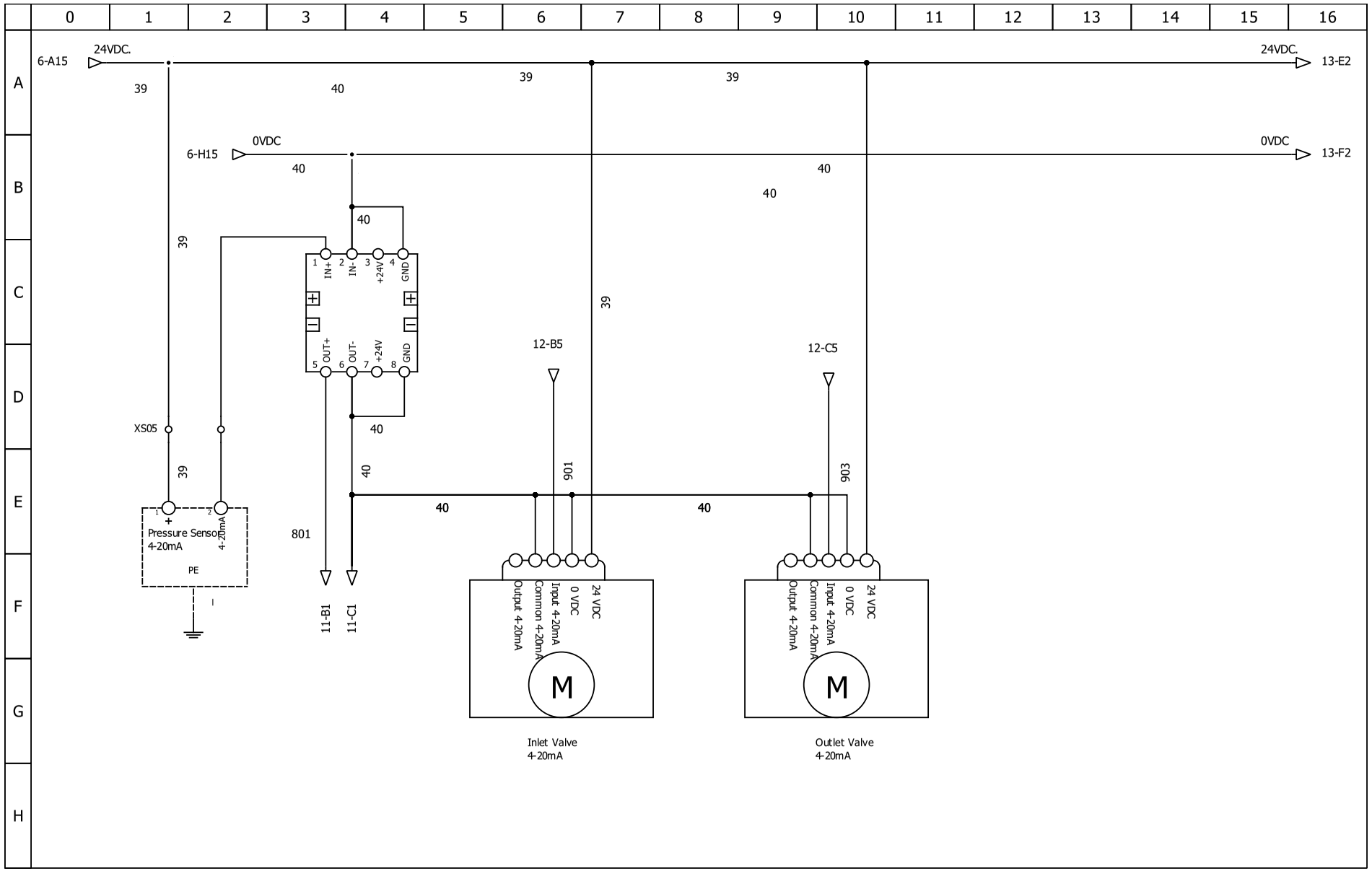

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
A
B
C
D
E
F
G
H
L1
L2
L3
N
21
22
23
24
L1
L2
L3
N
21
24
21
24
Q02.02
Schneider
Circuit Breaker
16A C 1P+N
QO2.03
Hager
Circuit Breaker
10A C 1P+N
25
26
27
28
Schuko Socket
1P+N
Schuko Socket
1P+N
TS03.00
legrand
Thermostat
29
13-F10
1000
L
N
PE
Finder
Cooling Fan
230VAC 1A

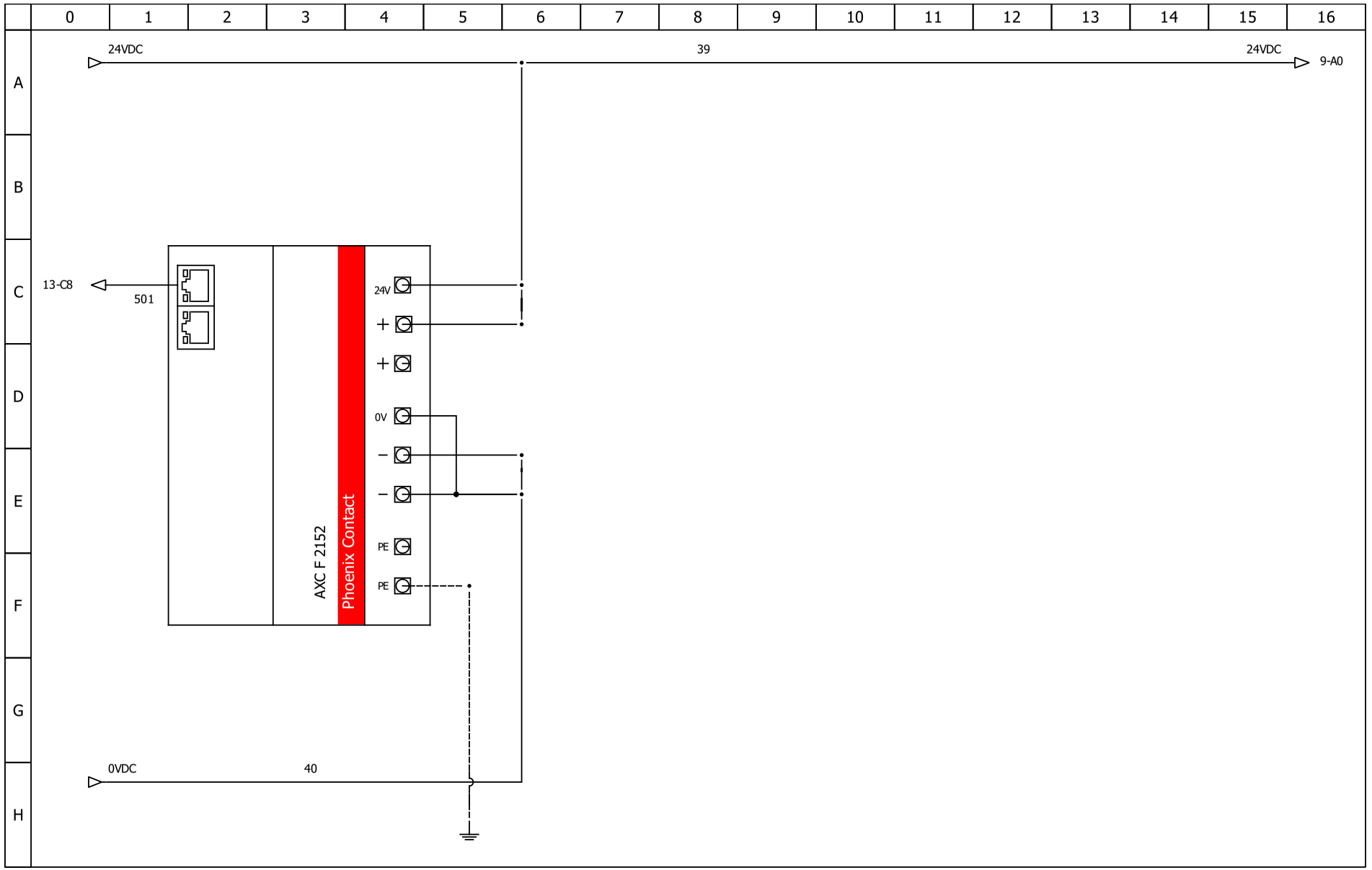

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
A
B
C
D
E
F
G
H
L1
L2
L3
N
21
22
23
24
L3
N
1 L1
3 L2
5 L3
QM03.01
Schneider
Motor Circuit Breaker
4A - 6.3A 3P
I>
I>
I>
2 T1
4 T2
5 T3
5-C3
30
31
32
L1
L2
L3
PE
VFD03.01
Variador
Goodrive20
1.5kW / 5A
5-C10
U
V
W
PE
33
34
36
XP01
4mm2
1
2
3
4
Grundfos Pump
750 W
M

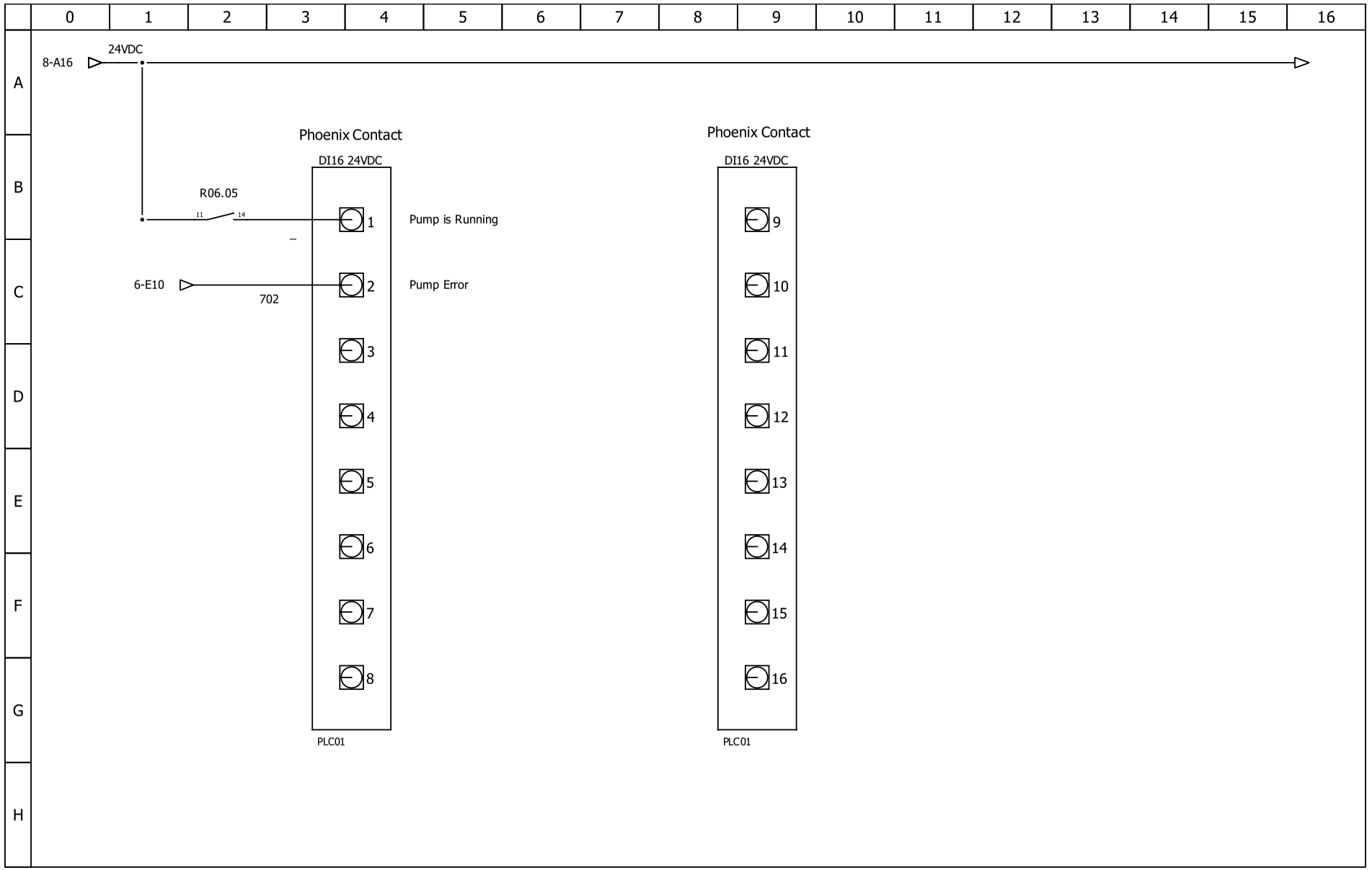

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
A
B
C
D
E
F
G
H
23
L3
24
N
Q06.03
Hager
Circuit Breaker
6A C 1P+N
1
N
2
N
PE
PE
40
39
0VDC
24VDC
5-H0
5-A0
UNO POWER
DC Power Source
230VAC / 24VDC 10A

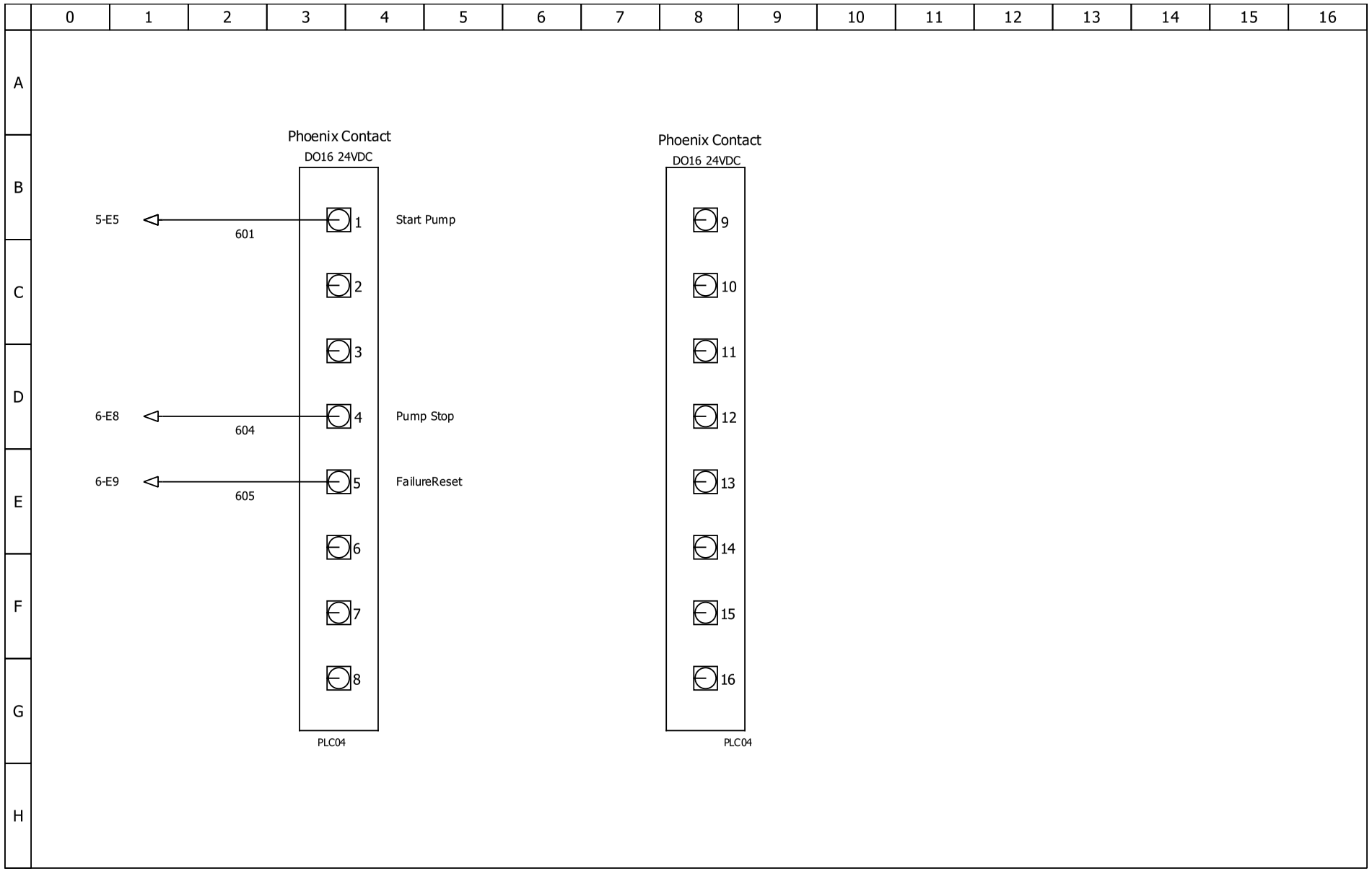

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
A B C D E F G H
4-D15
24VDC
39
6-A0
QM03.01
PMR01.01
101
R05.01
S05.00
M-0-A
5-D9
Auto
S05.00
103
104
S06.10
Parar
R05.02
105
S06.01
Arrancar
R05.03
106
107
VFD04.01
VFD Driving Pump
CMY
Y1
R06.01
108
R06.05
Pump Running
109
10-B1
601
R05.01
Relay 24VDC
AllOkay
R05.02
Relay 24VDC
Auto Run
R05.03
Relay 24VDC
Pump Start
R06.05
Relay 24VDC
Pump Running
S06.10
Pump
Defect
RD
S06.01
Pump
Running
GN
40
0VDC
6-H0
5-B7
5-C12
5-F8
6-E5
5-C14
9-B2

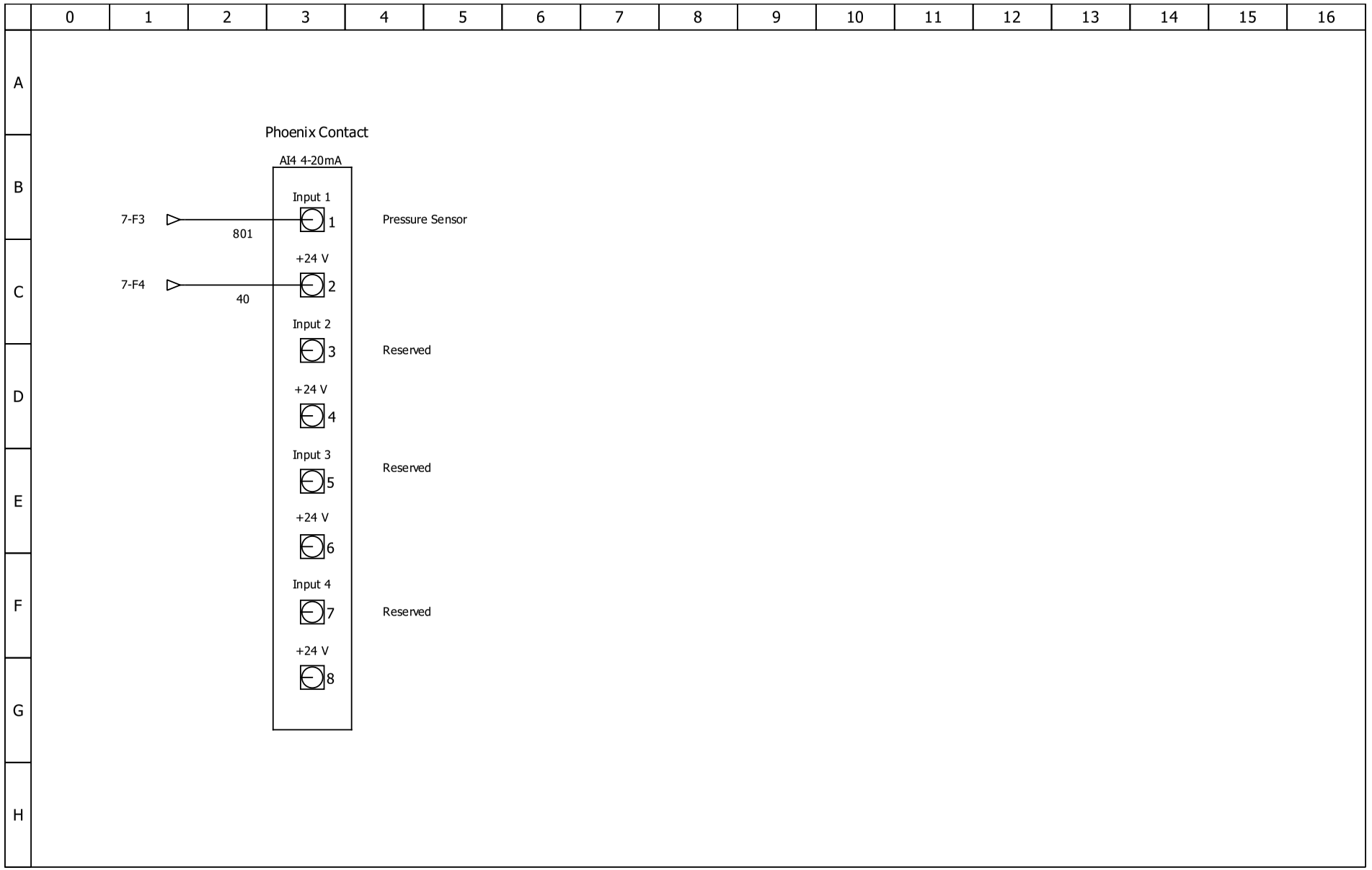

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
A
B
C
D
E
F
G
H
5-A15
24VDC
39
24VDC
7-A0
40
39
39
12-E5
905
XLOL7
4-20mA
PLC-VFD
Current-Voltage Signal Circuit
24VDC
0V
XLOL06
0-10V
201
R05.03
11
14
200
10-D1
604
10-E1
605
9-C1
702
S1
VFD
Foward
S2
Forward
Jogging
S3
Pump
Stop
S4
Failure
Reset
Y1
Pump Error
Signal
AO1
VFD Present
Frequency
AI2
VFD Set
Frequency
PWD
COM
Goodrive20
40
40
40
40
40
40
40
12-C4
0VDC
0VDC
5-H16
7-B2
40

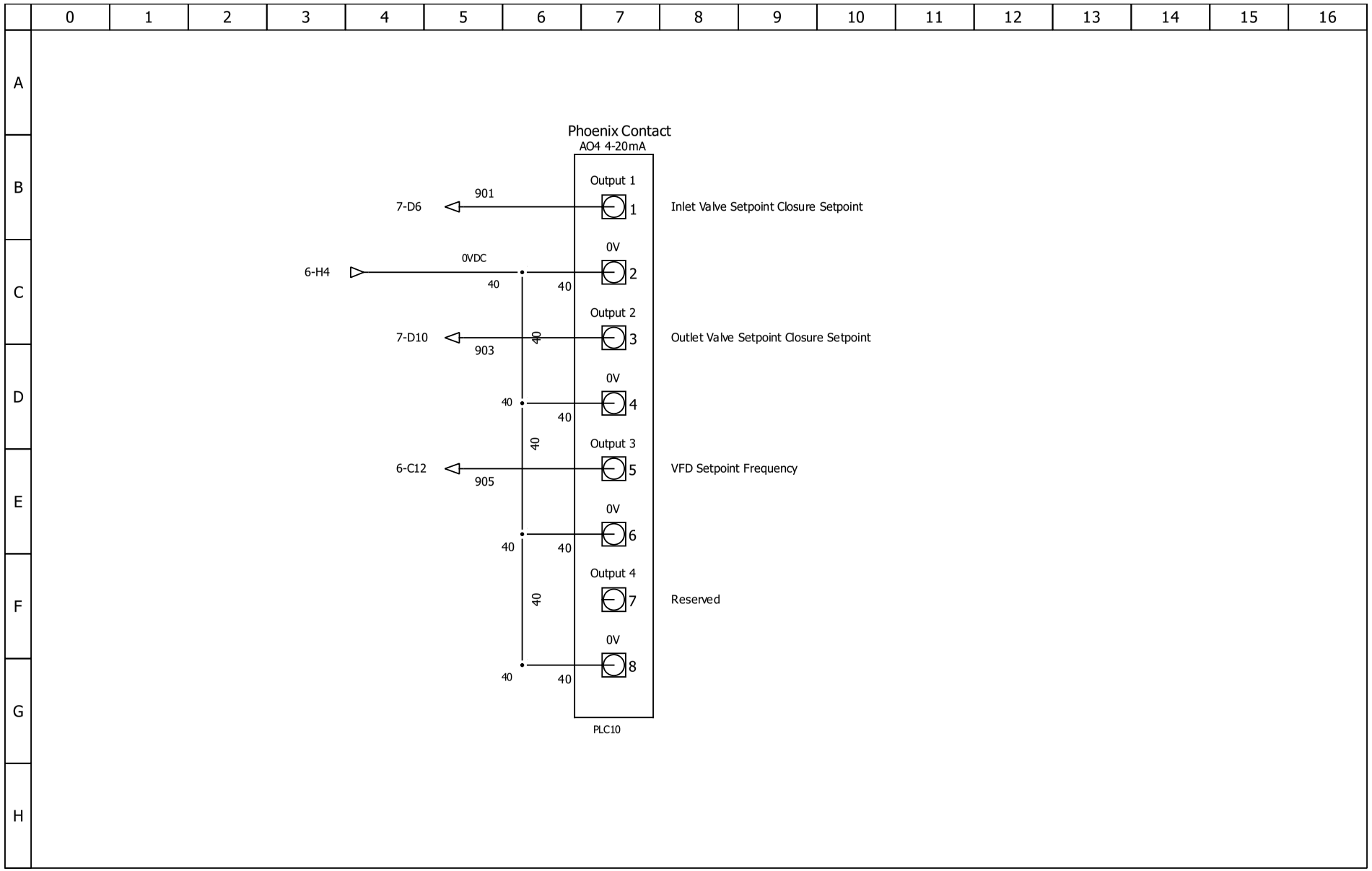

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
A
B
C
D
E
F
G
H
6-A15
24VDC.
24VDC.
13-E2
39
40
39
39
6-H15
0VDC
0VDC
13-F2
40
40
40
39
39
40
IN+
IN-
+24V
GND
OUT+
OUT-
+24V
GND
12-B5
12-C5
XS05
39
40
901
903
40
40
Pressure Sensor
4-20mA
4-20mA
PE
801
11-B1
11-C1
Output 4-20mA
Common 4-20mA
Input 4-20mA
0 VDC
24 VDC
M
Inlet Valve
4-20mA
Outlet Valve
4-20mA

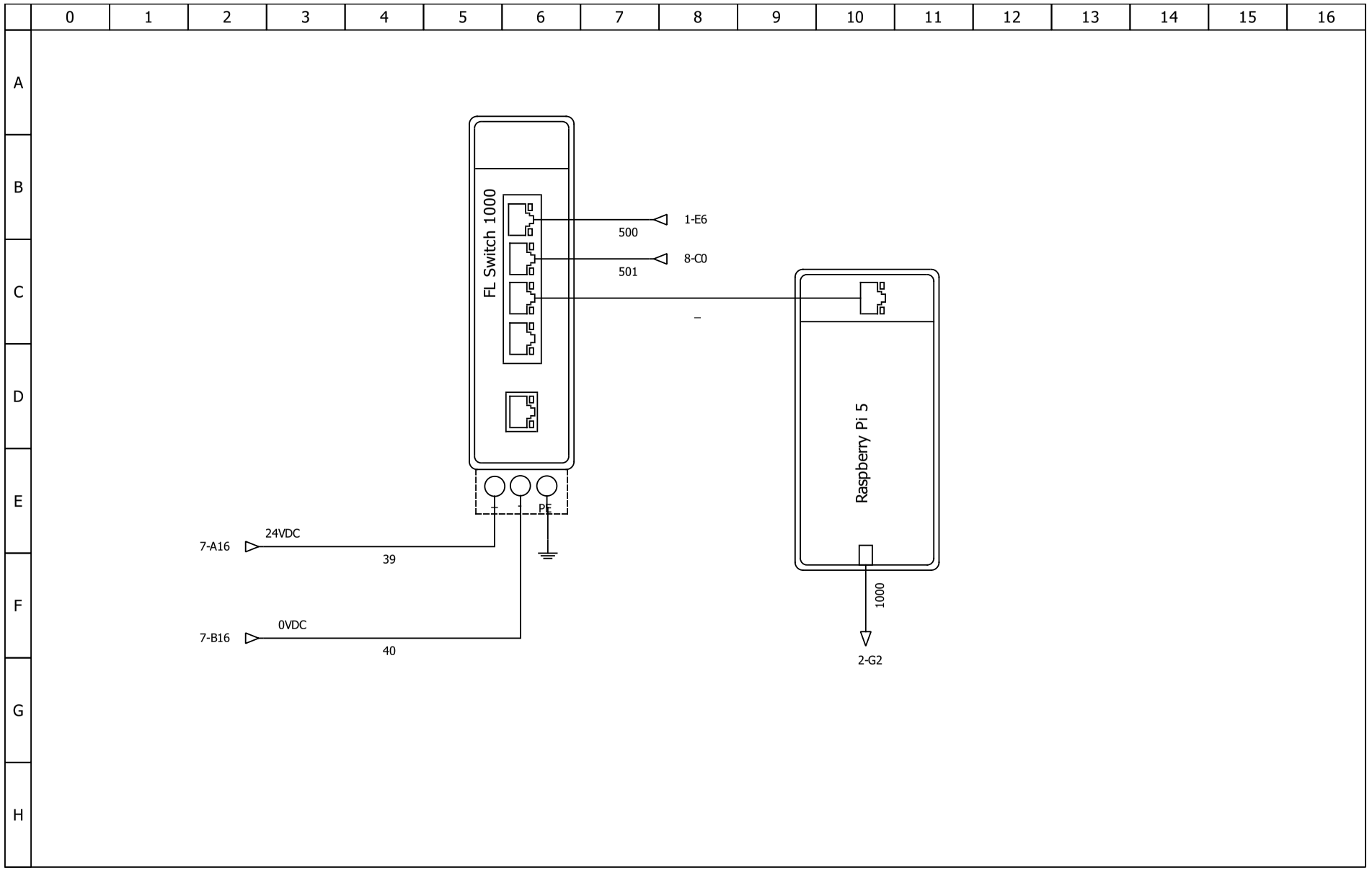

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
A
B
C
D
E
F
G
H
24VDC
39
24VDC
9-A0
13-C8
501
24V
+
+
0V
−
−
PE
PE
AXC F 2152
Phoenix Contact
0VDC
40


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
A
B
C
D
E
F
G
H
8-A16
24VDC
Phoenix Contact
DI16 24VDC
R06.05
11
14
1
Pump is Running
6-E10
2
Pump Error
702
3
4
5
6
7
8
PLC01
Phoenix Contact
DI16 24VDC
9
10
11
12
13
14
15
16
PLC01


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
A
B
C
D
E
F
G
H
Phoenix Contact
DO16 24VDC
5-E5
601
1
Start Pump
2
3
6-E8
604
4
Pump Stop
6-E9
605
5
FailureReset
6
7
8
PLC04
Phoenix Contact
DO16 24VDC
9
10
11
12
13
14
15
16
PLC04


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
A
B
C
D
E
F
G
H
Phoenix Contact
AI4 4-20mA
Input 1
7-F3
801
1
Pressure Sensor
+24 V
7-F4
40
2
Input 2
3
Reserved
+24 V
4
Input 3
5
Reserved
+24 V
6
Input 4
7
Reserved
+24 V
8


Phoenix Contact
AO4 4-20mA
Output 1
901
7-D6
1
Inlet Valve Setpoint Closure Setpoint
0V
0VDC
6-H4
2
40
40
Output 2
7-D10
40
3
Outlet Valve Setpoint Closure Setpoint
903
0V
40
4
40
40
Output 3
6-C12
5
VFD Setpoint Frequency
905
0V
40
6
40
Output 4
40
7
Reserved
0V
40
8
40
PLC10


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
A
B
C
D
E
F
G
H
FL Switch 1000
1-E6
500
8-C0
501
Raspberry Pi 5
1000
2-G2
PE
24VDC
7-A16
39
0VDC
7-B16
40

B

DAQ and Inference Code for Runtime

Below, in the next series of pages, is a single python script that runs on the Raspberry Pi, able to take measurements from the system built in this thesis and infer with it using a model trained with data having been preprocessed using any of the combinations of methods described in Table 3.8.

For it to work the API would have to be running and the asset documents, created before, at the time of training, would have to be present.

It boils down to the take_measurements() and the infer_with_pipeline() functions:

- The first function is essentially what was used for the data acquisition process.
- The second one is what worked so well for inferring, as shown in Table 6.3.

```
import subprocess
from pymodbus.client import ModbusTcpClient
import datetime
import json
from PyPlcnextRsc import Device
from PyPlcnextRsc.Arp.Plc.Gds.Services import IDataAccessService
from datetime import datetime
import pandas as pd
import requests
import pickle
import numpy as np
import struct


RPi_IP="127.0.0.1"
API_port=5000



with open('/path/to/plc_credentials.txt', 'rb') as file:
    credential_data = file.read()

cred = json.loads(credential_data)
secureInfoSupplier = lambda:(cred['loginword'],cred['passwrd'])
PLC_IP='XX.XX.XX.XX'



#EEM770 - Based code ( energy measurements)
##-------------------------------------------------------
def import_modbusenergy_registertable(file_path,sheet_name):
    # Read the filepath excel file in the sheetname sheet
    cleaner_df = pd.read_excel(file_path, sheet_name=sheet_name)
    cleaner_df['Name (EN)']=cleaner_df['Name (EN)'].apply(lambda x: x.
        replace(' ','_'))
    noduplicates_df=cleaner_df.drop_duplicates(subset='Name (EN)')
    return noduplicates_df

def ieee_754_converter(reg1, reg2):
    # Convert the integers to 16-bit binary strings
    bin1 = f"{reg1:016b}"
    bin2 = f"{reg2:016b}"

    # Extract the parts
    sign = int(bin1[0])
    exponent = int(bin1[1:9], 2)
    fraction_bits = bin1[9:] + bin2  # 7 bits from bin1 and 16 bits from
        bin2
    fraction = int(fraction_bits, 2) / (2**23) # turning it into fraction (
        if fraction_bits was 101, then fraction should be dec(0.101), this
        could also be done by dividing 5/2**3)

    # Compute the result
    result = ((-1) ** sign) * (1 + fraction) * (2 ** (exponent - 127))

    return result
```

```
def read_modbus_register(register, count, data_type, divider,client):
    result = client.read_holding_registers(register, count)
    #print(f'result={result.registers}')
    if result.isError():
        return None
    if data_type == 'Fl32' and count == 2:
        raw_value = ieee_754_converter(result.registers[0], result.
            registers[0])
    elif data_type == 'Uint16' and count == 1:
        raw_value = result.registers[0]
    elif data_type == 'Int16' and count == 1:
        raw_value = struct.unpack('>h', struct.pack('>H', result.registers
            [0]))[0]
    elif data_type == 'SInt16' and count == 1:
        raw_value = struct.unpack('>h', struct.pack('>H', result.registers
            [0]))[0]
    else:
        raise ValueError(f"Unsupported data type or count: {data_type}, {
            count}")
    return raw_value / divider if divider else raw_value

def convert_divider(value):
    try:
        return float(value)
    except (ValueError, TypeError):
        print(f'Value Error: {ValueError} | Type Error: {TypeError},
            turning divider to 1')
        return 1
```

```python
def extract_values_from_modbus(register_df,client):
    # Dictionary to store the values
    register_values = {}

    # Iterate through the DataFrame,
    for index, row in register_df.iterrows():
        if pd.notnull(row['Dec']) and row['R/W'] == 'R':
            register = int(row['Dec'])
            count = int(row['Count'])
            data_type = row['Datatype']
            divider = convert_divider(row['Divider'])
            name = row['Name (EN)']
            value = read_modbus_register(register, count, data_type,
                divider, client)
            if value is not None:
                register_values[name] = value

    client.close()

    # Convert the results to a DataFrame and save to a new Excel file
    result_df = pd.DataFrame(list(register_values.items()), columns=['Name'
        , 'Value'])

    pretty_result = result_df.merge(register_df[['Name (EN)', 'Unit', '
        Description (EN)']], left_on='Name', right_on='Name (EN)', how='left
        ').set_index('Name')

    # Drop the redundant 'Name (EN)' column and set Name as the index
    pretty_result = pretty_result.drop(columns=['Name (EN)'])
    # create the dict with the EEM-MA700 values
    eem_values={index: row["Value"] for index, row in pretty_result.
        iterrows()}
    # create the dict with  EEM-MA700 variable info
    eem_info=pretty_result.drop(columns=['Value']).to_dict('index')
    #pretty_result.to_excel('Pretty_Modbus_Values.xlsx', index=False)
    return eem_values, eem_info


# Modbus client setup
EEMMA770_IP = 'XX.XX.XX.YY'
filename='Clean_RegisterTable.xlsx'
path2file='/path/to/excelfile/with/modbusregisters'
file_path = path2file+filename
sheet_name='ForInference'
modbus_df = import_modbusenergy_registertable(file_path, sheet_name)
modbus_client = ModbusTcpClient(EEMMA770_IP)  # EEM-MA770
###-------------------------------------------------------
#getting  a pipeline from which to predict

# Load the pickle file
def load_results(filename):
    with open(filename, 'rb') as f:
        return pickle.load(f)
```

```python
# Sort all pipelines by macro F1 score
def sort_pipelines_by_f1_score(all_results):
    sorted_pipelines = []

    # Iterate over all results and collect the pipelines with their F1 scores
    for result in all_results:
        for model_name, model_info in result['models'].items():
            classification_report = model_info['classification_report']
            f1_score_macro = classification_report['macro avg']['f1-score']

            # Store relevant info in the sorted list
            sorted_pipelines.append({
                'model_name': model_name,
                'f1_score_macro': f1_score_macro,
                'pipeline': model_info['best_pipeline'],
                'params': model_info['best_params'],
                'result_info': result
            })

    # Sort the list by F1 score in descending order
    sorted_pipelines.sort(key=lambda x: x['f1_score_macro'], reverse=True)

    return sorted_pipelines


# Get a pipeline based on the row position in the sorted list
def get_pipeline_by_position(sorted_pipelines, position):
    if 0 <= position < len(sorted_pipelines):
        pipeline_info = sorted_pipelines[position]
        print(f"Model: {pipeline_info['model_name']}, F1 Score (Macro): {pipeline_info['f1_score_macro']}")
        #print(f"F1 Score (Macro Avg): {pipeline_info['f1_score_macro']}")
        #print(f"Best Params: {pipeline_info['params']}")
        return pipeline_info['pipeline']
    else:
        print("Invalid position. Please provide a position within the range.")
        return None
```

```
##-------------------------------------------------------
#preprocessing functions for vibration and inference functions


def process_fft(vibration, sampling_frequency):
    series = vibration

    fft_result = np.fft.fft(np.array(series, dtype=float))
    magnitude = np.abs(fft_result)

    frequencies = np.fft.fftfreq(len(series), d=1/sampling_frequency)
    return magnitude, frequencies

# RMS function remains unchanged
def rms(values):
    return np.sqrt(np.mean(np.square(values)))

# Sequential band_series function (without Parallel)
def band_series(series, series_size, band_size, type='mean'):
    band_size = int(band_size)
    if type.lower() == 'mean':
        banded_series = [np.mean(series[i:i+band_size]) for i in range(1,
            series_size-band_size, band_size)]
    elif type.lower() == 'rms':
        banded_series = [rms(series[i:i+band_size]) for i in range(1,
            series_size-band_size, band_size)]
    return banded_series

#normalising function
def normalise_logarithmically(series):
    series = np.array(series)
    log_series = np.log1p(series)  # log1p is used to avoid log(0) issues
    normalized_series = log_series / np.max(log_series)
    return normalized_series

#Finds the n highest peaks in each FFT
def find_inference_peaks(number_of_peaks, frequencies, fft):

    # Find indices where fft values are peaks (i.e., greater than their
        neighbors)
    peak_indices = [i for i in range(1, len(fft) - 1) if fft[i] > fft[i -
        1] and fft[i] > fft[i + 1]]

    # Sort the peaks by their FFT values in descending order
    sorted_peak_indices = sorted(peak_indices, key=lambda i: fft[i],
        reverse=True)

    # Select the top n peaks
    top_peak_indices = sorted_peak_indices[:number_of_peaks]

        # Append the frequencies and FFT values for the top peaks
    peaks_list = [(float(frequencies[i]), float(fft[i])) for i in
        top_peak_indices]

    return peaks_list
```

```
def create_doublecol_df(peaks_list):
    col_names=[]
    for i in range(0,len(peaks_list[0])):
        col_names.append(f"{i+1}_highest_peak_frequency")
        col_names.append(f"{i+1}_highest_peak_magnitude")

    # Create an empty DataFrame with the required number of columns
    double_column_df = pd.DataFrame(columns=col_names)

    # Populate the DataFrame
    for i, peaks in enumerate(peaks_list):
        row_data = []
        for peak in peaks:
            row_data.extend([peak[0], peak[1]])
        double_column_df.loc[i] = row_data
    return double_column_df

# Replace the row values in the sparse_tables with corresponding values from the tuple list
def create_sparsetable_df(sparse_tables, tuple_list):
    for freq, value in tuple_list:
        col_name = f"f{freq}".replace(".", ",")  # Convert to match column naming format
        if col_name in sparse_tables.columns:
            sparse_tables.at[0, col_name] = value  # Replace the row value in the column
    return sparse_tables
```

```python
def infer_with_pipeline(pipeline,
                        vibration,
                        normalisation,
                        banding,
                        dataframing,
                        additional_features_df,
                        number_of_peaks_dict,
                        sparse_tables_dict,
                        unique_pairs_dict,
                        sampling_rate=100000):
    """
    Uses the provided pipeline to infer on new data.
    """

    max_freq = int(sampling_rate / 2)

    # Process FFT for vibration data
    fft, frequencies = process_fft(vibration, sampling_rate)

    banded_frequencies = band_series(frequencies, max_freq, banding, type='mean')
    banded_fft = band_series(fft, max_freq, banding, type='rms')

    if str(normalisation) == "1":
        normalized_fft = normalise_logarithmically(banded_fft)
    else:
        normalized_fft = banded_fft

    inference_peaks = find_inference_peaks(number_of_peaks_dict[str(normalisation)][str(banding)], banded_frequencies, normalized_fft)

    # Split the data into training and testing sets
    if dataframing == "double_column":
        df = create_doublecol_df([inference_peaks])

    elif dataframing == "sparse_table":
        df = create_sparsetable_df(sparse_tables_dict[str(normalisation)][str(banding)], inference_peaks)

    df = df.reset_index(drop=True)
    additional_features_df = additional_features_df.reset_index(drop=True)

    X = pd.concat([df, additional_features_df], axis=1)

    if pipeline is not None:
        prediction = pipeline.predict(X)
        return unique_pairs_dict[str(prediction[0])]
    else:
        print("No valid pipeline provided!")
        return None
```

```
1
2  ##-----------------------------------------------------------
3  def take_measurements(PLC_IP,secureInfoSupplier, modbus_df, client):
4      ispumping, pressure, measured_freq = readsystemvariables(PLC_IP,
           secureInfoSupplier)
5      if ispumping==False:
6          print("pump isn't even running!")
7          return
8      vibration  = get_vibration(RPi_IP,  API_port)
9      vibration_sampling=vibration
10     eem_values, eem_info = extract_values_from_modbus(modbus_df,client)
11     return pressure, eem_values, vibration_sampling["voltages"]
12
13
14
15 ###-----------------------------------------------------------
16 def get_vibration(API_IP = "attentia.local", API_port = 5000, sensor_id =
       1):
17     """
18     Triggers the ESP to take a new vibration sample and fetches it.
19
20     :param sensor_id: ID of the vibration sensor.
21     :param esp_url: Base URL of the API handling the ESP communication.
22     :param timeout: Request timeout in seconds.
23     :return: Sample data as a dictionary.
24     """
25     timeout = 5
26
27     base_url = f"http://{API_IP}:{API_port}"
28     # Trigger the ESP to take a new sample
29     trigger_endpoint = f"{base_url}/take_sample?sensor_id={sensor_id}"
30     fetch_endpoint = f"{base_url}/fetch_samples?sensor_id={sensor_id}&count
           =1"
31
32     try:
33         # Step 1: Send a command to trigger the ESP
34         trigger_response = requests.get(trigger_endpoint, timeout=timeout)
35         if trigger_response.status_code != 200:
36             raise Exception(f"Trigger request failed: {trigger_response.
                   text}")
37
38         # Step 2: Retrieve the sample
39         fetch_response = requests.get(fetch_endpoint, timeout=timeout)
40         if fetch_response.status_code != 200:
41             raise Exception(f"Fetch request failed: {fetch_response.text}")
42
43         data = fetch_response.json()
44         if not data.get("samples"):
45             raise ValueError("No samples returned by the ESP.")
46
47         return data["samples"][0]
48
49     except Exception as e:
50         print(f"Error in get_vibration: {e}")
51         return None
```

```
###-----------------------------------------------------

def readsystemvariables(IP,login_credentials):
    with Device(IP, secureInfoSupplier=login_credentials) as device:
        data_access_service = IDataAccessService(device)
        # read one variable
        read_item = data_access_service.ReadSingle("Arp.Plc.Eclr/VFD1.
            RunningPump")
        # get the value of the read_item
        ispumping = read_item.Value.GetValue()
        # get the data type of the read_item
        ispumpingtype = read_item.Value.GetType()

        read_item = data_access_service.ReadSingle("Arp.Plc.Eclr/Valveset1.
            OutletPressure")
        pressure = read_item.Value.GetValue()
        pressuretype = read_item.Value.GetType()

        read_item = data_access_service.ReadSingle("Arp.Plc.Eclr/VFD1.
            OperatingFrequency")
        measured_freq = read_item.Value.GetValue()
        freqtype = read_item.Value.GetType()

        return ispumping, pressure, measured_freq



def measure_and_predict(pipeline_used = 2):
#pipeline used is the position of the pipeline in the sorted_pipelines list
    . The highest is the best (0). 4 is working for now (sparse table isn't
    working)

    all_results = load_results('./assets/all_models.pkl')
    sorted_pipelines = sort_pipelines_by_f1_score(all_results)
    number_of_peaks_dict = load_results('./assets/number_of_peaks.pkl')
    sparse_tables_dict=load_results('./assets/sparse_tables.pkl')
    unique_pairs_dict=load_results('./assets/encoded_labels.pkl')
    pressureandenergy_order = load_results('./assets/
        energyandpressure_order.pkl')

    pipeline_to_use = get_pipeline_by_position(sorted_pipelines,
        pipeline_used)

    api_process = subprocess.Popen(['python', 'API.py'])
    pressureval, eem_values, vibrationlist=take_measurements(PLC_IP,
        secureInfoSupplier, modbus_df, modbus_client)

    pressureandenergy_df = pd.DataFrame()
    pressureandenergy_df['pressure'] = [pressureval]
    for key, value in eem_values.items():
        pressureandenergy_df[key] = [value]

    pressureandenergy_df = pressureandenergy_df[pressureandenergy_order] #
        reorder the columns to match the model's expectations
```

```
1
2       api_process.terminate()
3       print("API has been STOPPED")
4       predicted_result = infer_with_pipeline(pipeline=pipeline_to_use,
5                                              vibration = vibrationlist,
6                                              normalisation =
                                                   sorted_pipelines[
                                                   pipeline_used]["result_info"
                                                   ]["normalisation"],
7                                              banding = sorted_pipelines[
                                                   pipeline_used]["result_info"
                                                   ]["band_width"],
8                                              dataframing = sorted_pipelines[
                                                   pipeline_used]["result_info"
                                                   ]["dataframing"],
9                                              additional_features_df =
                                                   pressureandenergy_df,
10                                             number_of_peaks_dict =
                                                   number_of_peaks_dict,
11                                             sparse_tables_dict =
                                                   sparse_tables_dict,
12                                             unique_pairs_dict =
                                                   unique_pairs_dict,
13                                             sampling_rate=100000)
14      datadict={'datetime':datetime.now(),'pumpstate':predicted_result,'
           model_used (see in results_df)':pipeline_used}
15
16
17      return predicted_result
18
19
20
21  if __name__ == "__main__":
22      measure_and_predict()
```