

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

# **Predictive Maintenance of Rotary Draw Bending Machines Using Artificial Intelligence**

**Rui Afonso Patrício Sá Marques**



Mestrado Integrado em Engenharia Mecânica

Supervisor: Ana Rosanete Lourenço Reis

Second Supervisor: Pedro Nuno Fontes de Oliveira

October 1, 2021



# **Predictive Maintenance of Rotary Draw Bending Machines Using Artificial Intelligence**

**Rui Afonso Patrício Sá Marques**

Mestrado Integrado em Engenharia Mecânica

October 1, 2021



# Abstract

This dissertation was developed as part of an effort by AMOB in partnership with INEGI to bring its machines into the new era of manufacturing, commonly referred to as Industry 4.0. This refers to machines with the capacity for data collection, analysis and storage, which enable greater production capabilities and detailed insights into the production process. As technological advancements make the development of such machines possible, there is a big push towards the adoption of these solutions in manufacturing, as they result in more efficient, productive, lucrative and sustainable manufacturing.

The focus of this thesis is on establishing the infrastructure needed to make predictive maintenance possible in AMOB's raw draw bending machines through the use of artificial intelligence. To do this, research into the state of the art of raw draw bending machines, maintenance and artificial intelligence was conducted. The main concerns regarding failure of AMOB's machines were established and root cause analyses of these failures was conducted for a specific problem that was affecting one of the machines at the time. A framework and architecture for the systems needed for the implementation of an artificial intelligence based predictive maintenance system was devised based on the research. Data from the machines was collected and analysed in order to validate some choices regarding this system and test some hypotheses regarding how the wear of the machine can manifest itself in this data.



# Resumo

Esta dissertação foi desenvolvida no âmbito da intenção da AMOB, em parceria com o INEGI, de trazer as suas máquinas para a nova era de processos de fabrico, apelidada de Indústria 4.0. Este termo refere-se a máquinas com capacidade para recolha, análise e armazenamento de dados, o que permite maiores capacidades produtivas e um visão detalhada dos processos produtivos. À medida que o progresso tecnológico tornam o desenvolvimento de tais máquinas possível, há uma grande pressão para a adoção destas tecnologias, pois delas resultam uma maior eficiência, produtividade, lucro e sustentabilidade ambientais.

O foco desta tese consiste em estabelecer a infraestrutura necessária para tornar possível a manutenção preditiva das máquinas curvadoras de tubo da AMOB, através do uso de inteligência artificial. Para tal, pesquisas sobre o estado da arte das máquinas curvadoras, manutenção e inteligência artificial foram feitas. As principais preocupações em termos de falha das máquinas da AMOB foram estabelecidas e uma análise da causa raiz foi feita para o caso específico de uma máquina que se encontrava com problemas à época. Uma estrutura para os sistemas necessários para a implementação de uma estratégia de manutenção preditiva baseada em inteligência artificial foi criada, com base na pesquisa feita. Dados das máquinas foram recolhidos e analisados, de forma a validar algumas das escolhas feitas na estruturação do sistema e também para testar algumas hipóteses formuladas a respeito de como o desgaste da máquina poderia manifestar-se nestes dados.





# Acknowledgements

I would like to start by thanking AMOB, INEGI and FEUP, and everyone at these institutions that helped make this thesis.

I want to thank my thesis coordinators, professor Ana Reis and engineer Pedro Oliveira for the guidance in this thesis.

A special thanks to eng. Rui Amaral, eng. Miguel Fonseca, eng. Carlos Agra, professor Bessa Pacheco, eng. João Paulo Sousa, eng. Ricardo Carvalho, eng. João Martins, professor René Vinicio Sánchez and all others who, in their own way, contributed to this thesis.

I also want to thank AMOB for providing me with a space to conduct the work of this thesis and the machines used in testing, and INEGI for providing the sensors and equipment needed for those tests.

This thesis was developed under the POCI-01-0247-FEDER-038454 – Smart2B project, which is an integrated IoT platform solution for laser cutting. It was co-financed by the operational program for internationalization and competitiveness COMPETE 2020, through FEDER, European Fund for Regional Development, and the company AMOB. The work was developed in INEGI with the unit of advanced manufacturing processes project nº 38454.

Rui Marques





*“There can be 100 people in the room, and 99 don’t believe, but all it takes is one”*

Stefani Germanotta



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	The company . . . . .	2
1.3	Objectives . . . . .	2
<b>2</b>	<b>State of the art</b>	<b>3</b>
2.1	Metal Tube Conforming Machines . . . . .	3
2.1.1	Press Bending . . . . .	3
2.1.2	Roll Bending . . . . .	4
2.1.3	Compression bending . . . . .	4
2.1.4	Rotary Draw Bending . . . . .	4
2.2	Maintenance . . . . .	9
2.2.1	Preventive . . . . .	10
2.2.2	Condition-based . . . . .	11
2.2.3	Predictive . . . . .	12
2.2.4	Prescriptive . . . . .	13
2.3	Creation of a predictive maintenance system . . . . .	14
2.4	Data pre-processing . . . . .	15
2.4.1	Compression . . . . .	15
2.4.2	Signal Features . . . . .	17

2.5	Modeling techniques . . . . .	18
2.5.1	ML techniques . . . . .	19
2.6	Model validation . . . . .	21
2.6.1	Accuracy score . . . . .	21
2.6.2	Confusion Matrix . . . . .	22
2.7	Machine monitoring . . . . .	22
2.7.1	Wear and failure indicator parameters . . . . .	22
<b>3</b>	<b>Case Study</b>	<b>27</b>
3.1	The machines . . . . .	27
3.2	Current maintenance strategy of e-MOB machines . . . . .	29
3.3	Failure pain points . . . . .	29
3.3.1	Linear guides . . . . .	31
3.3.2	Ball Screws . . . . .	31
3.4	e-MOB 42 9611 with vertical axis problems . . . . .	32
3.4.1	Machine's history . . . . .	33
3.4.2	Problem . . . . .	34
3.4.3	Approach . . . . .	34
3.4.4	H Axis . . . . .	34
3.4.5	Ball screw analysis . . . . .	38
3.4.6	Failure causes . . . . .	39
3.4.7	Conclusions . . . . .	45
<b>4</b>	<b>Architecture of the PdM system</b>	<b>47</b>
4.1	Introduction . . . . .	47
4.2	Data gathering and sensorization . . . . .	48
4.2.1	Available data . . . . .	48

4.2.2	Sensorization . . . . .	49
4.3	Data Processing . . . . .	53
4.4	Data Storage . . . . .	57
4.5	Communication and data standards . . . . .	59
4.6	System Overview . . . . .	59
<b>5</b>	<b>Preliminary testing</b>	<b>61</b>
5.1	Vibration . . . . .	61
5.1.1	Objectives . . . . .	61
5.1.2	Theory . . . . .	62
5.1.3	Approach . . . . .	63
5.1.4	Results and discussion . . . . .	65
5.1.5	Conclusions . . . . .	70
5.2	Torque . . . . .	71
5.2.1	Theory and objectives . . . . .	71
5.2.2	Approach . . . . .	71
5.2.3	Results and discussion . . . . .	72
5.2.4	Conclusions . . . . .	77
<b>6</b>	<b>Conclusion</b>	<b>79</b>
6.1	Conclusion . . . . .	79
6.2	Future Work . . . . .	80
<b>A</b>		<b>85</b>
<b>B</b>		<b>95</b>
<b>C</b>		<b>99</b>





# List of Figures

2.1	Main parts of a rotary draw bending machine responsible for forming the tube (AMOB, b) adapted . . . . .	5
2.2	Spool bend die from 2 angles . . . . .	6
2.3	Simple plug mandrel . . . . .	8
2.4	Standard mandrel . . . . .	8
2.5	Wiper die from two different angles . . . . .	9
2.6	Workflow diagram for preventive maintenance based on time . . . . .	10
2.7	Workflow chart of condition-based maintenance . . . . .	12
2.8	Workflow diagram for predictive maintenance . . . . .	13
2.9	Workflow diagram of the creation of a PDM system . . . . .	15
2.10	Frequency type transforms and their properties (Lensu, 1998) . . . . .	16
2.11	Confusion Matrix . . . . .	22
2.12	Standard ISO 10816-8:2014 (Yasir et al., 2019) . . . . .	24
2.13	Typical characteristics of a piezoelectrical sensor and a MEMS sensor (Jung et al.)	25
2.14	a) Time-domain graph b) Frequency-domain graph . . . . .	25
3.1	e-MOB's axis designation (AMOB, a) . . . . .	28
3.2	Carriage damage due to manual collision in 9611 e-MOB axis . . . . .	30
3.3	Ball screw of the H axis of one of AMOB's machines . . . . .	32
3.4	e-MOB 42 9611 at the client . . . . .	33

3.5	Linear guide in one of AMOB's machines . . . . .	35
3.6	Bending head carriage with H axis ball-recirculating screw marked as 1 (AMOB, a) . . . . .	36
3.7	Engineering drawing of the assembly of the ball screw in the H axis of e-MOB 52 . . . . .	37
3.8	Places to lubricate when doing H axis maintenance (AMOB, a) . . . . .	37
3.9	Lubrication unit from one of AMOB's machines from two perspectives . . . . .	38
3.10	Replaced ball screw with severe wear signs . . . . .	39
3.11	Ishikawa diagram with the possible causes of the ball screw failure . . . . .	40
3.12	Machine with the parallelism planes colored . . . . .	42
3.13	Machine during the alignment procedures . . . . .	43
3.14	Ball screw during alignment procedure, before and after the panel is lowered . . . . .	44
3.15	Service life factor as a function of misalignment, adapted (HIWIN, 1998a) . . . . .	46
4.1	Wiring diagram of the ADXL345 sensors connected via I2C . . . . .	50
4.2	Main sensor mounted on the nut carrying part of the H axis . . . . .	51
4.3	Wiring diagram of the ADXL345 sensor connected via 4 wires SPI (nagimov, 2020) . . . . .	51
4.4	Data flow chart for the accelerometer . . . . .	56
4.5	Flow of data diagram between IoT Edge device and Microsoft Azure cloud services (an emma et al., 2020) . . . . .	58
4.6	Machine-Raspberry Pi communication diagram . . . . .	59
4.7	End-to-end data flow diagram of system . . . . .	60
5.1	Representation of angular deformation . . . . .	62
5.2	Examples of time-domain data from the ADXL345 connected via I2C for a sampling rate over 400 Hz . . . . .	65
5.3	Examples of frequency-domain data from the ADXL345 connected via I2C for a sampling rate over 400 Hz . . . . .	66
5.4	Examples of torque-position (on the left) and torque speed graphs (on the right) with key areas marked . . . . .	73

5.5	Torque-position (on the left) and torque speed graphs (on the right) for e-MOB 42 9611 . . . . .	74
5.6	Torque-position (on the left) and torque speed graphs (on the right) for e-MOB 42 9611 (blue) and the e-MOB 42 available at AMOB (orange) . . . . .	75
5.7	Torque-position for e-MOB 42 9611 (blue line) and the e-MOB 42 available at AMOB (orange line) with areas of interest colored . . . . .	76



# List of Tables

2.1	Ranking of time domain features of an accelerometer signal and accuracy after KNN using a ReliefF ranking algorithm (Sánchez) . . . . .	18
4.1	Condition indicators formulas . . . . .	55
5.1	Positions and velocities of vibration trials . . . . .	64
5.2	Condition indicators for the vibration trials in respect to the X axis of the sensor .	67
5.3	Condition indicators for the vibration trials in respect to the Y axis of the sensor .	68
5.4	Condition indicators for the vibration trials in respect to the Z axis of the sensor .	69
5.5	Metrics of torque for parts movements where the malfunctioning machine diverges from the control machine at AMOB . . . . .	76



# Symbols and Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
AQEC	Aerospace Qualified Electronic Component
CNC	Computer Numerical Control
CSV	Comma Separated Values
DCT	Discrete Cosine Transform
DWT	Discrete Wavelet Transform
FFPN	Feature Fusion Pyramid Network
FFT	Fast Fourier Transform
FIFO	First In First Out
HMI	Human Machine Interface
I2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
IoT	Internet of Things
IT	Information Technology
KNN	K-Nearest Neighbor
MEMS	MicroElectroMechanical Systems
LSB	Least Significant Bit
ML	Machine Learning
PC	Personal Computer
PdM	Predictive Maintenance
PNG	Portable Network Graphics
RPM	Rotations Per Minute
RUL	Remaining Useful Life
SPI	Serial Peripheral Interface
SQL	Structured Query Language
SVM	Support Vector Machine
VAT	Value Added Tax





# Chapter 1

## Introduction

With the advent of new technologies and the rise of the so called “Industry 4.0”, technology has become more and more ubiquitous in manufacturing processes. These technologies have gathered interest in recent times, as they have the potential to solve some of the most pressing problems of our times, such as environmental sustainability and value proposition in a global market.

One area that can benefit from these technologies is maintenance. One of the biggest challenges when making a machine is planning its maintenance, as the sheer number of variables that influence the failure of a part or system makes this decision mainly based on previous *in-situ* experience rather than a data driven process. Technologies such as big data, artificial intelligence (AI) and the convergence of information technology (IT) and operational technology (OT) are tools that make it possible to make this a data driven process and to improve the efficiency of this task, from both an economical and an environmental standpoint.

### 1.1 Motivation

This thesis comes as a response to an overall push in the industry towards the use of such technologies in improving the longevity of their products and their efficiency. An example of the tube bending industry also moving in this direction is the BLM Group, who is also working in this field, as evidenced by the 2019 thesis titled “Analysis of process signals from rotary draw bending operations” ([SORIANI, 2018/2019](#)).

In the specific case of AMOB, the company in which this thesis is being developed, predictive maintenance solutions help keep its products better compete against its competitors’ machines and also provide its customers more value with their machines.

## 1.2 The company

AMOB is a Portuguese company based in Vila Nova de Famalicão that specializes in the production of tube and pipe bending machines. Since its foundation in 1969, the family company has grown into one of the biggest players in the metalworking technology field, with operations in all five continents and in over 30 countries.

Their main plant is adjacent to its headquarters in Vila Nova de Famalicão, spanning 18000m<sup>2</sup> and with over 140 members on its staff.

AMOB is currently, in partnership with INEGI, developing solutions to make their machines connected to the cloud, in order to offer more comprehensive and intelligent solutions to their clients. This not only allows the client to see real time information about their machines and the productive process from anywhere, as allows AMOB to provide better support to their customers, both by collecting data from the machines during their lifetime to develop better maintenance strategies, provide remote support with greater insights into the problem without having to make an in person visit and also predict a monitor in real-time future needs of a client.

## 1.3 Objectives

The main objective of this work is to develop and test the basis of predictive maintenance tools for one of the axes of AMOB's e-MOB machines. The point of this is to serve as the foundation for building predictive maintenance systems into their future machines.

This requires an investigation into the state of the art of both rotary draw bending, maintenance and artificial intelligence, the main pillars of this thesis. The aim of this is to get a sense of the task ahead and to familiarize with the processes and tools available in order to make the best choices when devising the architecture of the system.

Another aim is to understand what are the main concerns in the e-MOB range and find out their causes. This will define the focus of the system being developed.

## Chapter 2

# State of the art

Research on maintenance of tube bending machines, especially more advanced types of maintenance such as predictive maintenance, is still in its infancy, not only on raw draw bending machines but on the use of technology to implement predictive maintenance systems.

Most manufacturers do not openly disclose their research and implementation of these systems, therefore, in this chapter, most of the findings and research presented is academic, and not existing solutions.

### 2.1 Metal Tube Conforming Machines

Metal tube conforming machines are machines whose function is to conform the tubular profiles into a given complex shape, by plastically deforming them. These conformed tubes are used in applications ranging from the aeronautical industry to agricultural machinery, with parts obtained through this process also being used in other transportation industries and architecture ([Pacheco et al., 2019](#)).

#### 2.1.1 Press Bending

Press bending, also known as ram bending, is one of the simplest and cheapest tube bending methods. It is based on the three-point-bending principle, in which a force applied between two support points causes the supported beam or tube to bend ([Pacheco et al., 2019](#)). This method can be applied manually, by using a manual press with a tool with the shape and radius of the tube to conform, or a mechanical press, which uses a hydraulic press to apply force to the conforming tool

(Pacheco et al., 2019). These tools look similar to an arch of a pulley. Different tools are needed in order to obtain different radii.

### 2.1.2 Roll Bending

Encyclopedia Britannica defines calendaring, another term used to refer to roll bending, as a: “process of smoothing and compressing a material [...] by passing a single continuous sheet through a number of pairs of [...] rolls.” (Britannica, 2007). In respect to tube bending, this process is slightly different, as it is a tube passing through three pulleys like tools. In these machines, the tools are arranged in an isosceles triangular formation, with the center pulley only slightly raised from the alignment of the other two. This disposition makes it impossible for the tube to pass between the sets of tools straight, therefore bending it. The radius of the arch of the tube is dictated by the distance between the two lower cylinders. Although this machine can conform different radii with the same set of tools, different tools are needed for different diameter tubes. This kind of machine is mostly used to conform large radii.

### 2.1.3 Compression bending

This process consists of conforming a tube by rotation of a tool, which can be straight or curved, around a pulley shaped tool that is fixed on the inner side of the curve (Pacheco et al., 2019). A support is needed in order to sustain one edge of the tube, acting as a point for the reaction force caused by the movement of the rotating tool to act upon (Pacheco et al., 2019). This forces the tube to conform to the shape of the inner tool instead of just rotating around it (Pacheco et al., 2019). The inner tool radius and the angle position of the rotating tool determine the radius and the angle of the tube curve respectively (Pacheco et al., 2019).

### 2.1.4 Rotary Draw Bending

Rotary draw bending is a technique that is mainly used for tight curves with relatively small radii and complex geometries, that require several bends at different angles and in different positions. It is one of the most commonly used types of tube bending techniques.

Its working principle is very similar to the rotary compression bending machine, the main difference being that in this process the tool that rotates, clamp die, as the name indicates, restrains the tube, not allowing slippage. This, along with the pressure die, which controls the amount of material “fed” to the curve, force the tube to stretch along the bend die, taking its shape in the process.

The main advantages of this process compared to compression bending is that it allows for more control over the process and allows for tighter bends (smaller curve radius) (STAM). Besides this, the use of complementary tools in this process, such as wiper dies and mandrels, allow for tighter dimensional tolerances, less defects and a better cosmetic finishing than competing processes (STAM).

#### 2.1.4.1 Parts and tools

These machines have five main components that directly bend the tube, which can be seen in figure 2.1, namely (Köseoğlu and Parlak, 2012):

- **Bend die;**
- **Clamp die;**
- **Pressure die;**
- **Mandrel;**
- **Wiper die.**

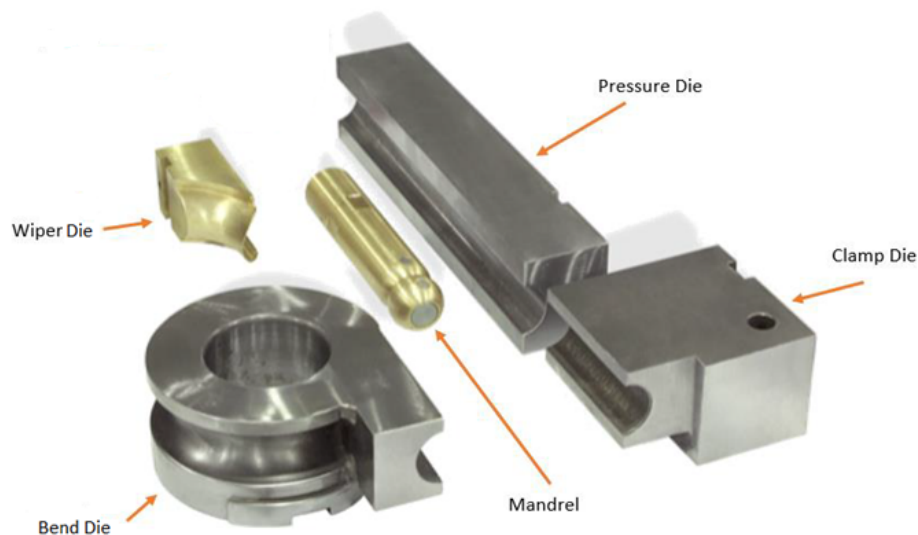


Figure 2.1: Main parts of a rotary draw bending machine responsible for forming the tube (AMOB, b) adapted

The first three tools mentioned are responsible for bending the tube per se, as the others are responsible for ensuring that the tube bends and conforms properly, by preventing its wrinkling, rupture,

ovalization and thickness reduction on the outside of the bend of the tube (STAM; Pacheco et al., 2019).

The remainder of the machine mainly serves three functions: to support the tube, actuate the mentioned tools and to position the tools and tube for each bend. The systems responsible for these functions are dependent on the machine's type, meaning whether it is hydraulically or electrically actuated.

### Bend die

The bend die is one of the most important tools of a raw draw bending machine. It is the tool around which the tube wraps when it is being bent. Therefore, this is the tool that imposes the curve radius to the tube. This tool is usually either bolted or fixed using flanges to the machine and can be stacked on each other.

The dimensions of these dies is dictated by the parameters of the bend, namely: the outside diameter (OD), the center line radius (CLR) and the maximum degree of bend (DOB). As the main material requirement for this part is toughness, shock resistant steels are commonly used. There are also several variants of this tool, each with a particular geometry. The main ones are (OMNI-X, 2016):

- **Spool** — This tool is made of two components: the grip, which holds the tube during the bending, and the die, the part which gives the tube its shape. These parts are separate and therefore can be replaced individually. Different grips are available depending on the surface topology wanted for the tube. It is mounted on a tool post that fits through the middle of the tool. By far the most popular solution as the grip and die can be replaced separately. An example of a spool can be seen in figure 2.2.

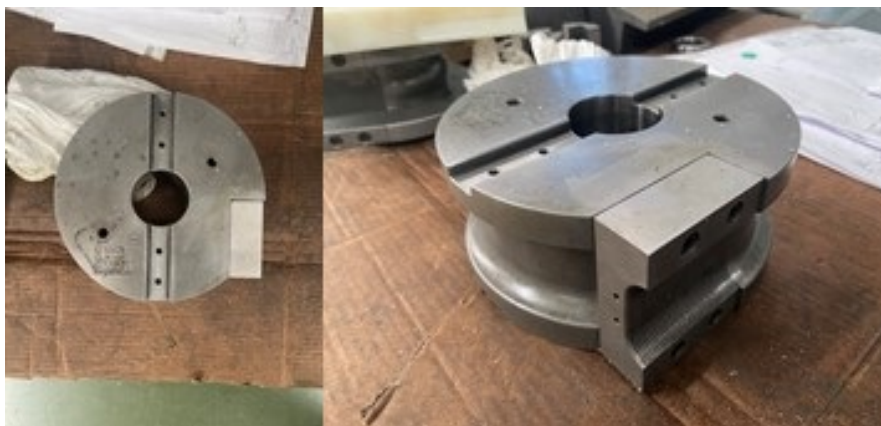


Figure 2.2: Spool bend die from 2 angles

- **One piece** — Similar to the spool but the grip is non-removable and cannot be separated from the rest of the tool.
- **Partial platform** — Similar to the one piece die but has extra material on one side of the die in order to support itself on the machine.
- **Full platform** — Same as partial platform dies but has extra material on both sides.
- **Flange** — Does not have a hole in the middle and is not mounted on a tool post, instead it is fixed to the machine through bolts and nuts. Used in small machines that cannot accommodate a tool post.

### Clamp die

The clamp die is the tool responsible for applying the pressure that holds the tube between itself and the seizure part of the bend die during bending (OMNI-X, 2016). It has the same finish and length of the clamping part of the clamp die, usually a grip (OMNI-X, 2016). Its objective is to prevent any slippage from happening (OMNI-X, 2016).

### Pressure die

The pressure die is a tool that accompanies the tube, sitting parallel to it, making sure that pressure between the tool and the bend die is constant throughout the bending process (OMNI-X, 2016).

### Mandrel

The mandrel is a very important tool when tube bending, as it is the main tool responsible for the prevention of defects and imperfections such as crinkling and ovalization. This is achieved by providing the tube with a solid support that counteracts the forces that lead to these defects. It is of the utmost importance that the mandrel is properly fitted into the tube, as large gaps leave room for deformation.

The type of mandrel used is dependent on how demanding the bend is and the likelihood of defects.

There are several types of mandrels, such as (OMNI-X, 2016):

- **Simple plug** — A straight cylindrical rod with a slightly smaller diameter than the inner diameter of the tube. Used for large radii bends. One can be seen in figure 2.3.



Figure 2.3: Simple plug mandrel

- **Simple formed plug** — A straight cylindrical rod with a slight curvature towards the end in the shape of the bend. Provides more support to the outer wall of the tube during bending.
- **Standard mandrel** — Cylindrical metal rod with a series of balls at the end attached to each other with a similar radius to the rest of the mandrel, as shown in figure 2.4. These balls act as rotulas and allow the end of the mandrel to bend and, therefore, for the mandrel to penetrate deeper into the tube, allowing it to support the tube for longer. The balls are connected to each other and to the rest of the mandrel via links, a small plug with a spherical ending that fits into a socket with a similar shape on the other ball or the rest of the mandrel. It is the most common type of mandrel as it is able to support most bends.



Figure 2.4: Standard mandrel



- **Close pitch, thin-walled mandrel** — This is an iteration of the standard mandrel for, as the name suggests, small radii bends and thin wall tubes. In this mandrel the rotulas are closer together.

The mandrels also vary in material, with aluminium-bronze and hard-chrome plated steel being the most used materials. This is mostly due to the very good cost-performance relationship of these materials for this specific application.

### Wiper die

The wiper die has a similar function to the mandrel, complementing it by also helping minimize crinkling on the tube's inner side during bending. Because of this overlap of functions, it is not always used, being mostly used for tubes with thin walls and tight bends with small radii. These are usually made of the same material as the mandrel. In figure 2.5 a wiper die made by AMOB can be seen.

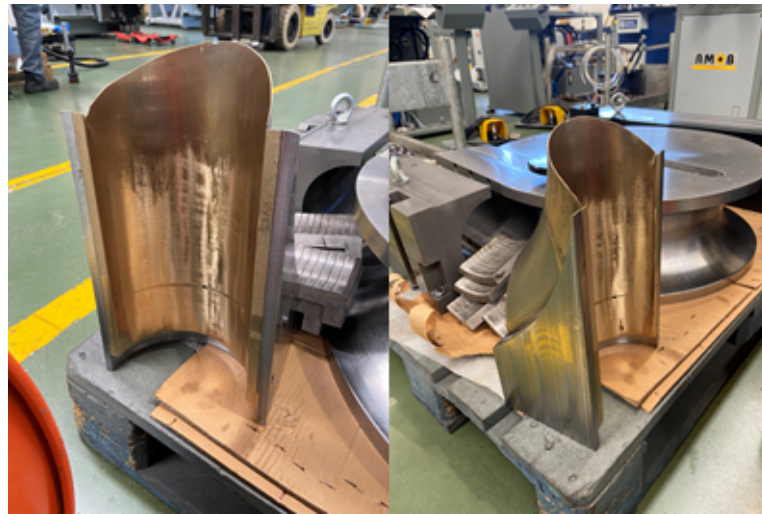


Figure 2.5: Wiper die from two different angles

## 2.2 Maintenance

In the European Standard EN 13306:2017 maintenance is defined as “[...] the combination of all technical, administrative and managerial actions during the life cycle of an item intended to retain it in, or restore it to, a state in which it can perform the required function”. In simpler terms, the aim of maintenance is to maintain the good functioning of a given machine and to prevent it

from breaking down. This makes maintenance an important part of any company, as it can affect parameters such as productivity, quality and costs.

Maintenance affects these parameters in two ways, as lack of maintenance leads to machine failure, which in turn means stopping the production line until the problem is fixed or the machine replaced, which normally is not a quick process and carries heavy costs, but also over cautious maintenance can lead to inefficiencies, as maintenance procedures take time away from working time, as it is common for the machine having to be out of use during these procedures, but also because parts will be replaced when they can still endure more time in service.

Maintenance can be divided into two big groups: active maintenance, where the machine is proactively maintained in order to avoid failure, and reactive maintenance, where no effort is made to prevent failure and the machine is only serviced after failure (Sullivan et al., 2010). In this dissertation the focus will be on the first kind of maintenance, which is subdivided into preventive and predictive maintenance.

### 2.2.1 Preventive

Preventive maintenance is one of the more conventional types of maintenance, which is based on scheduled inspections and repairs triggered by reaching a predetermined value of a parameter, such as time in service or mileage. The guidelines and timing for these routine inspections and repairs is often based on empirical data gathered from previous tests executed by the manufacturer. This type of maintenance can be summarized by the workflow diagram on figure 2.6.

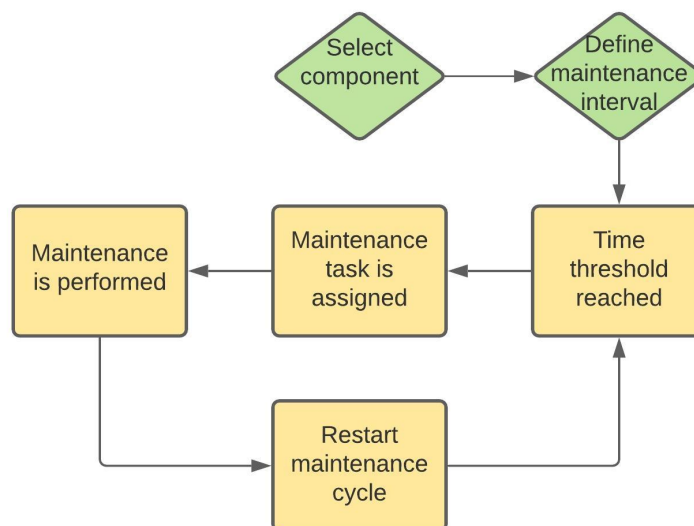


Figure 2.6: Workflow diagram for preventive maintenance based on time

Therefore, this kind of maintenance is planned assuming a given set of use conditions, commonly referred to as normal wear. If the conditions of actual use are different from the ones considered when planning for machine maintenance, the plan made may no longer be appropriate. In order to assure that the machine is getting enough maintenance even if the conditions of use vary within a reasonable margin, it is common practice to anticipate maintenance procedures in comparison to the results obtained in testing. This results in a lower failure rate of the machines in actual use but inserts more inefficiency and waste through the machine's lifetime, as parts which still had useful life ahead of them might be discarded and replaced sooner than needed, by turn increasing maintenance costs during the machine's lifetime.

Another problem that arises from this process not considering the actual use of the machine being serviced is that the system is unable to respond to unpredicted events that may affect the machine's good functioning. This may lead to machine failure between scheduled maintenances, which can be very costly.

Despite its many shortcomings, preventive maintenance is still widely used as it does not require additional equipment in each machine to monitor its functioning and is relatively easy to implement, especially in simple machines. This makes it a very low entry cost solution.

This makes this solution optimal for low-cost short lifetime machines, in which the cost of implementing other solutions highly impacts the price of the machine and the wasted potential of the replaced and repaired parts does not constitute a significant cost in the overall cost of the machine over its lifetime.

### **2.2.2 Condition-based**

Similarly to predictive maintenance, condition-based maintenance is based on real-time data collected by the machine's sensors. It is usually a precursor method to predictive maintenance as it is simpler and faster to implement, while having a lower cost .

Condition-based maintenance relies on real-time data to monitor the machine's function, emitting a warning when a parameter reaches a previously calculated threshold. The main difference to predictive maintenance is that instead of giving an estimation of how much use can we expect from the part or machine before failure, it only warns when failure is impending. This means that in this method there is no need to keep a record of the machine's data, which means a simpler system. It also bypasses the process of creating a model of the machine's lifetime, which requires large amounts of data and, in machines whose expected life is very extended, take a lot of time to collect. Therefore, this method is quicker to implement, which is why it can be thought of as a precursor to predictive maintenance, as it can be implemented while data is being gathered to model the machine's lifetime.

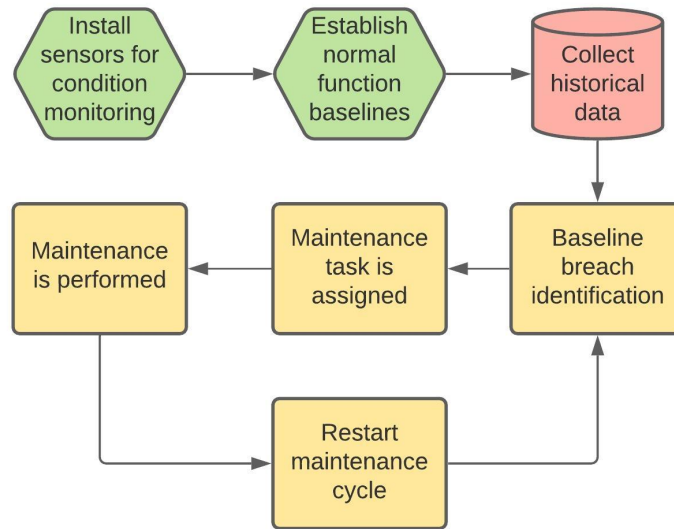


Figure 2.7: Workflow chart of condition-based maintenance

### 2.2.3 Predictive

Predictive maintenance (PdM) uses a different approach into the maintenance problem, where through data, gathered by sensors and the machine itself, a model is established for the useful life of the machine. The model obtained allows for the prediction of the remaining useful lifetime of the machine, which in turn allows maintenance procedures to be done only when the part is approaching end-of-life. This leads to significant savings, with the parts only being replaced exactly when needed.

The workflow diagram of this type of maintenance is in every way similar to the one of the condition-based maintenance, with the exception of how the processing and analysis of data are done. In predictive maintenance, this monitoring of the condition based on data is done through software, which establishes the thresholds of normal function, as in condition-based maintenance this is done prior by a human. With the use of tools such as machine learning (ML), a feedback loop in the data collection and establishment of the baselines can be created, as the data collected during the machine's operation can be added to the pool of data used to train and assess the model.

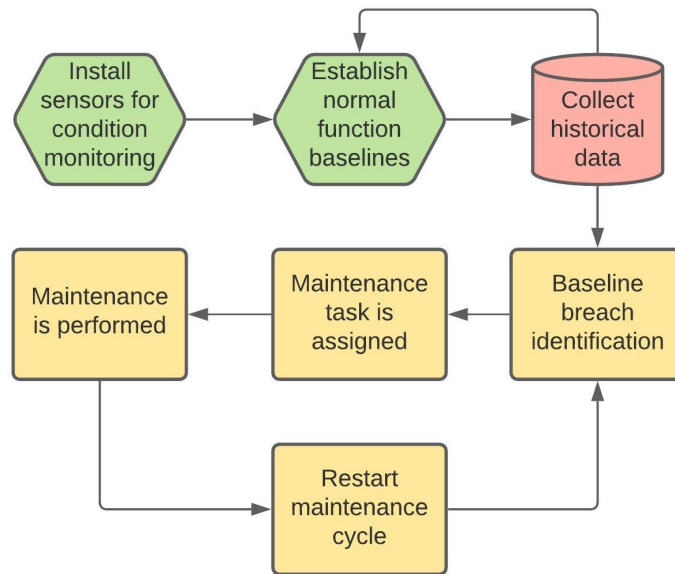


Figure 2.8: Workflow diagram for predictive maintenance

The drawbacks of this kind of maintenance are its high initial costs, since the machine has to be equipped with the necessary sensors, and also the fact that not every machine is able to be adapted to this kind of maintenance. This can be because of computerization of the machine, for example hydraulic machines that use analog controls, or inability to isolate and understand the parameters that can serve as indicators of failure. It can also happen that the cost of adapting the machine to a digital environment, although possible, reveals to costly to justify it.

Overall, some estimates put the savings of adopting this kind of maintenance compared with reactive maintenance in the 25% to 30% range (Sullivan et al., 2010). When compared with the 12% to 18% savings the authors estimate for preventive maintenance (Sullivan et al., 2010), we can see that this process almost doubles the savings of just implementing a standard maintenance strategy.

#### 2.2.4 Prescriptive

This is an emerging type of maintenance which builds on predictive maintenance by automatically adjusting parameters and tacking actions to prolong the machine's remaining useful life (RUL) and minimize machine downtime due to maintenance tasks.

This involves giving the machine the ability to perform some maintenance tasks, bypassing the human operator. However, the point of this kind of maintenance is not to replace entirely the human operator, but instead to collaborate and complement its work.

An illustrative example of this type of maintenance being used in a machine that is able to self-regulate the amount of lubrication used based on the wear of the part in question, but still needing to be replaced by an operator when it reaches the end of its useful life.

This process is only now starting to be implemented as a high level of automation, computer power and connectivity is needed for this sort of solutions, which was only made possible with the rise of technologies such as internet of things (IoT), ML and 5G, which has latencies low enough to allow the processing of the amounts of data involved in this process possible.

### **2.3 Creation of a predictive maintenance system**

In order to implement a predictive maintenance system, it is necessary to conceive models for the machine lifetime and failure.

The first step in establishing this model is to understand how the failure occurs and what variables might play a role and indicate it. To do this it is necessary to understand how the machine works. If the machine is already in use, a starting point is to query the operators about the failure and to study the current maintenance strategy guiding parameters. This may help in formulating hypotheses about how to tackle the problem and in selecting, or excluding, variables to retrieve data about.

Once established the variables to study, it is necessary to start collecting the data. In order for this to be possible, sensors may need to be selected and installed. The selection and installation of these sensors must consider: the space and part where the sensor will be mounted, its range and accuracy, the price and availability.

At the data collection stage, how this data is acquired has to be defined. This includes defining the sampling rate, the measurement duration, the scale and the bandwidth. It is important at this stage to choose a convenient storage format, in order to later facilitate its analysis.

For PdM algorithms, historical data from throughout the lifetime of the part is needed. In fact, for a more accurate model data from multiple identical parts is desirable. Since in an industrial setting parts are usually dimensioned for a lifetime of 20000 hours, this process takes a long time. In fact, it is the most time consuming part of creating a PdM system and may take years. One benefit of the use ML models is that they can be optimized by each new reading, possibly reducing the amount of data needed to deploy the system and therefore the time needed. However, complete data from at least one lifetime of one part is still needed as a basis.

Next, how the data will be analyzed must be decided. This means choosing what the inputs of the algorithm will be. One option is to analyze the signal as a whole. Because this means very large amounts of data compression is needed. Compression algorithms can be divided into two,

depending on whether or not all of the data can be later retrieved: lossless and lossy compression algorithms. Another alternative is to only analyze key features of the signal that are related to failure. Signal features are numbers that help characterize the signal, such as the average or variance. If a feature is related to the wear or good functioning of a machine is called a condition indicator. Depending on the type of analysis, for example whether it is a time domain analysis or frequency domain analysis, different signal features will be good condition indicators. These condition indicators can be extracted locally in real time, foregoing the need to store the signal itself. This means a very low amount of data needs to be stored. In order for a correct choice of condition indicators the ML algorithm that will be used to create the model must be chosen.

Finally, tests should be made in order to validate the model obtained. There are two key metrics in evaluating a model that should be established: the confusion matrix and the accuracy score. This allows the assessment of how good the model is at predicting the state of the part or machine.

The steps involved in the creation of a PdM system are summarized in the diagram of figure 2.9.

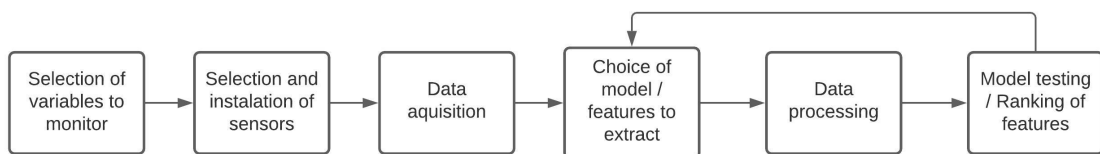


Figure 2.9: Workflow diagram of the creation of a PdM system

## 2.4 Data pre-processing

### 2.4.1 Compression

The objective of compression is to make a file smaller without incurring on information loss (Wayner, 2009). This is done by identifying redundancies and irrelevancies in data and reducing them (Note, 2011). Lossless compression is based on redundancy minimization while lossy compression is on irrelevancy minimization (Note, 2011).

Frequency type transforms are a method of pre-processing time dependent data input to a computer, allowing for the computation of signal spectrums (Lensu, 1998).

There are several different types of frequency type transforms, depending on the base sinusoid function used to form the data recorded (Lensu, 1998). The most common transforms and their

properties are shown in figure 2.10.

Transform	$p_I$	Properties	Applications
discrete Fourier	3	requires complex arithmetic, very good energy compaction	digital signal processing, convolution, digital filtering, analysis
discrete cosine	3	requires real operations, near optimal substitute for the KL transform, excellent energy compaction	transform coders, Wiener filters
discrete sine	4	symmetric, requires real operations, yields fast KL transform, very good energy compaction	block processing algorithms, coding, filtering, performance estimation
Hadamard	4	no multiplications, difficult to analyze, good energy compaction	digital hardware implementations, image data compression, filtering, design of codes
Haar	5	very good energy compaction	feature extraction, image coding, image analysis
Slant	3	has "image-like" basis, very good energy compaction	image coding
Karhunen-Loève	1	optimal in many ways, best energy compaction in the MS sense	performance evaluation, useful for small size vectors
fast KL	2	better performance than independent block-by-block processing	recursive-block processing techniques, adaptive techniques
sinusoidal	3	excellent energy compaction for the optimum-fast transform	practical substitutes for the KL transform, analysis, mathematical modelling of signals
singular value decomposition	1	best energy-packing efficiency for any image, varies drastically from image to image	separable FIR filters, LS and minimum norm solutions of linear equations, rank of large matrices, image restoration, power spectrum estimation, data compression

Figure 2.10: Frequency type transforms and their properties (Lensu, 1998)

Some of the most used transforms for data processing are DCT and discrete wavelet transform (DWT). Because image capture is one of the most used tools both in science and in day-to-day life, and the need for more and better images, most studies done in this field were done using images, which are a 2-D or 3-D signals, meaning a 2-D or 3-D array is needed to store the signals, in contrast to the end use of this paper, sensor data compression, which is a 1-D signal, where the quantity recorded is stored as a 1-D signal dependent only on time (Lensu, 1998). Nonetheless, the conclusions obtained for two and three-dimensional signals can be ported to 1-D signals.

DCT is often considered the best compression technique, as its performance is nearly identical to the Karhunen–Loève transform (KLT), which is regarded as the benchmark transform for noise reduction as it has a very good rate-distortion (RD) performance (Chen and Zeng, 2012), while being faster and more efficient.



DWT is considered an alternative to DCT that offers better data quality (Bansal and Dubey, 2013). It decomposes the signal into two by filtering out frequencies, a low frequency signal obtained using a low pass filter and a high frequency signal using a high pass filter (Katharotiya, Patel and Mahesh 2011). The high frequency is regarded as the detailed part of the signal and the low frequency as the approximation (Katharotiya et al., 2011).

In the paper titled “Comparative Analysis between DCT & DWT Techniques of Image Compression” (Katharotiya et al., 2011) the authors concluded a significantly higher compression ratio is obtained through DWT (Cr = 1.9 to 2.3) then with DCT (Cr=1.6) with no significant information loss, albeit DCT being more time efficient than DWT.

### 2.4.2 Signal Features

Some types of algorithms, such as support vector machine (SVM) algorithms, do not use the curve of the data itself as input, only needing key features of its distribution in order to characterize the data. Therefore, there is only need to extract and store these features from the signals and not the entire dataset. Some of these features are: arithmetic mean, 50th percentile, trimmed mean, interquartile range, mean absolute deviation, range, variance, standard deviation, root mean square, kurtosis, skewness, crest factor, impulse factor and margin factor (Lee et al., 2019). From these, the ones that present changes with the wear of the tool or the machine should be kept and used as input on the chosen algorithm (Lee et al., 2019). For oscillatory signals, features are grouped depending on the analysis being made, with features specific for time domain, frequency domain and frequency time domain. Using the methodology described in the paper titled “Multi-fault Diagnosis of Rotating Machinery by Using Feature Ranking Methods and SVM-based Classifiers” (Sánchez et al., 2017), and analysing 64 time domain features and 24 frequency domain features, with a total of 88 time-frequency domain features, but using KNN instead of SVM and FFPN as classification algorithms, René-Vinicio Sanchez and his colleagues reached the rankings displayed in table 2.1 using the ReliefF algorithm to rank the features. This algorithm builds on the Relief algorithm, which is used for binary classification, and evaluates how good a feature is at classifying neighboring instances. In the table 2.1 the evolution of the kNN algorithm with the increase of features used, surpassing 90% accuracy with the top 6 ranked features.

Table 2.1: Ranking of time domain features of an accelerometer signal and accuracy after KNN using a ReliefF ranking algorithm (Sánchez)

Ranking	Feature	KNN (%)
1	Average	11.15
2	Skewness	18.99
3	Slope sign change	52.27
4	Zero crossing	79.44
5	Histogram upper limit	89.10
6	Margin index	90.01
7	Kurtosis	91.41
8	Latitude factor	91.66
9	Slack factor	90.00
10	Mean deviation ratio	91.82

## 2.5 Modeling techniques

One way of creating a model of a machine's lifetime would be to run consecutive tests from new until failure and use the data collected to train a ML algorithm. This approach is very reliable and simple.

However, as mentioned before, due to the long lifetime of some components, this approach may not be viable. An alternative is to train a ML model with data from the machine in normal working conditions, in order for it to be able to detect abnormal functioning of the machine.

One approach to this is called anomaly or novelty detection. A baseline of the parameters being monitored is established and an ML algorithm that once trained and implemented is able to monitor data from the machine for spikes and dips that fall out of the normal function range and make a maintenance warning. By using ML, the algorithm keeps learning from the monitoring data, becoming more accurate over time. This approach is more suitable for monitoring unexpected changes in a machine and not progressive wear and tear. This approach models the normal functioning of a machine or component instead of its wear and tear.

An approach more suitable for monitoring wear and tear is by establishing wear conditions for the tool in question and using AI to process data from the machine and assign a wear state to the machine (e.g. normal functioning, approaching end of life, imminent failure).

Both wear and tear prediction and anomaly detection can be used to create a robust PdM system, since they tackle different facets of the maintenance problem, complementing each other.

### 2.5.1 ML techniques

Machine learning algorithms are sets of instructions that models a problem and solves it without direct instructions. It is a branch of AI that focuses on finding patterns in data in order to predict outcomes. This makes ML a powerful tool to tackle complex problems with ever-changing data that are otherwise too complex to model.

Machine learning can be classified in 3 ways: supervised learning, where the datasets that train the algorithm are already labeled; unsupervised learning, here the algorithm is trained with un-categorised data, with it having to find patterns and clusters; and reinforced learning, where the algorithm “learns” by analyzing the outcome of the decisions it made, operating in a feedback loop, not requiring previous training ([Azure, 2021](#)).

Some ML algorithms are: anomaly or novelty detection algorithms, where the goal is to identify points that fall out of the “normal” range; regression algorithms, which use previous data to extrapolate the value of future data; time series algorithms, which predict the evolution of the data, identifying trends and other cyclical patterns; clustering algorithms, which segregate data into groups based on similarities; and classification algorithms, which classify data into predefined groups ([Azure, 2021](#)).

This chapter is a concise presentation of some of the most commonly used ML algorithms.

#### 2.5.1.1 Linear regression

One of the most common types of regression and ML in general, linear regression aims to establish a straight line that best characterizes the progression of the data. It works on two dimensional datasets. It qualifies as a progression type of ML algorithm, as it has a numerical target variable. This means the objective of this type of algorithm is to allow for extrapolation and interpolation of the data, allowing us to find the target variable value for values of the independent variable which are not in the training dataset.

Other regressions exist for when the data fits a curve (e.g. parabola) different than a straight line. All these regressions function similarly to the linear regression, just with a different formula for the appropriate curve and different parameters to set.

Most regression algorithms resort to the mean square error (MSE) as a cost function. A cost function is a function that assesses the performance of a ML algorithm by measuring the error of the ML model, meaning the discrepancy between the prediction and reality.

### 2.5.1.2 Decision tree

This is a classification algorithm, meaning the objective is to sort the target variable(s) into categories. They allow the computer to "understand" based on characteristics of an object what that object is (i.e. what category it belongs to).

Decision trees divide the data into two or more subsets using if-then functions. They owe their name to their visual representation, which branches out the further down gone. Each leaf represents a class and each branch a set of conditions, usually written as logical operators.

These are one of the most common types of ML algorithms due to their simplicity and their immediacy of comprehension.

This algorithm is also the basis of more advanced and complex algorithms such as the random forest, which creates several trees at random and congregates the results of each one in order to reach a final result.

Gradient boosting algorithms are also usually associated with decision trees, as it combines different models in order to improve the performance of the algorithm. The most common use is to join weak models, meaning models that are only marginally better than pure chance, such as decision trees, in a process called ensembling.

### 2.5.1.3 K-nearest neighbor

Like the decision-tree, this is also a classification algorithm. It is also a non-parametric algorithm, meaning it is not based on probability distribution families, which makes it more flexible to classify data whose distribution does not resemble any known probability distribution family. Finally, it is a non-linear algorithm, meaning it can make non-linear relationships and therefore does not use hyperplanes (see section 2.5.1.4) to divide the data into categories. It can work both as a supervised and an unsupervised algorithm.

This algorithm models data by considering the data close to the point being analyzed and classifying it, either by the most common occurrence in its vicinity or, alternatively, by assigning different weights to each data point based on its distance to the point in analysis. This is based on the assumption that similar objects tend to exist in a similar space. Its name comes from  $k$  being a user-imputed variable that sets the number of data points nearest to the test point the algorithm makes to find the boundaries of each class. The most frequent class in those  $k$  points will be the class of that test point.

#### 2.5.1.4 Support vector machine

These are models based on algorithms that classify data into classes based on datasets from which it learns. In this algorithm, data is interpreted as a  $n$  dimension vector, and its goal is to separate this data into  $n - 1$  spaces, called hyperplanes.

This algorithm is used in supervised learning, where the dataset used to train the algorithm are already labeled, although it can also be used in unsupervised learning where the algorithm tries to establish classes of data with similar characteristics within the class and different characteristics from other classes.

Although similar to kNN, SVM is more limited in what patterns it can find in the data, with the advantage being it is less computationally demanding.

## 2.6 Model validation

In order to implement a model, it is necessary to know how well it describes real life. This is called model validation.

There are several tools to compare the model with the actual data and to assess the model's accuracy. These tools help in the comparison of models in order to choose the one that more closely predicts future data.

### 2.6.1 Accuracy score

This is a percentage that results from supervised algorithms testing its model against the whole dataset. This value is not particularly useful to evaluate the model in what pertains to the real world, but it is mostly used by the algorithms themselves internally to improve their models. This is so as many ML algorithms work by creating a random model based on part of the dataset, seeing how well it fits the whole dataset, getting this score and repeating. If the score of the current model is greater than the previous one, that model is kept. This is done until the accuracy value over a number of models stabilizes. It is not uncommon for simple models (e.g. two classes) to have an accuracy score of 100%, meaning the model perfectly classifies the whole dataset, which does not mean in real life the model is 100% accurate.

## 2.6.2 Confusion Matrix

A confusion matrix is a visual representation of the accuracy of the model against a given set of data. It allows for quickly checking the number of times an algorithm correctly labels a data point and the number of times it labels in incorrectly in each of the other classes. This makes it easier to see how well the model fits that data, compare it to others and identify weaknesses in the model, such as confusion of some classes (mislabeling).

In figure 2.11 a confusion matrix model for two classes, positive and negative, is presented. The green and red cells should be filled with the number of times the event within them happens.

		Predicted Condition	
		Positive	Negative
Actual Condition	Positive	True Positive	False Negative
	Negative	False Positive	True Negative

Figure 2.11: Confusion Matrix

## 2.7 Machine monitoring

In this chapter relevant topics about which parameters are usually monitored for machine failure and wear are monitored, as well as common methods employed to do so. The focus will be on parameters and methods of monitoring that can be deployed in the machine and part that will be discussed in the case study section of this thesis.

### 2.7.1 Wear and failure indicator parameters

Even though failure can have innumerable causes, wear is usually due to grinding between moving parts in common. This can be mostly avoided by correct lubrication, which reduces friction between parts and therefore reduces wear. This is why the main focus of many maintenance strategies is in maintaining adequate lubrication.

By understanding the mechanisms of wear and tear of mechanical parts, possible indicators of this phenomenon can be hypothesized, namely by understanding the byproducts of grinding due to poor lubrication or excessive wear.

Wear, for the most part, can be defined as damage caused by use, which mostly consists of erosion of contact surfaces between parts, which lead to higher tolerances between them and altered surface characteristics, such as roughness. Both of these factors can manifest themselves during use as noise and abnormal vibration.

#### **2.7.1.1 Torque**

As gaps between parts widen, debris from the wear parts accumulates and roughness gets more uneven, the motors powering the moving parts will have to output more power to compensate for these impediments to the movement. This is also true when lubrication is insufficient, to account for the increase in friction, and when there is failure due to other unexpected causes, as the motor will always try to overcome impediments to its functioning. The way motors do this is by increasing their torque output, outputting more force to try to keep its commanded functioning parameters (achieve a certain position or speed).

Therefore, monitoring motor torque is a good way of monitoring machine wear and failure, as both of these will manifest themselves as an increase of torque. This is an especially simple way of monitoring these conditions in machines equipped with servomotors, as this information is readily available. In fact, this allows even for the establishment of limits to this value and real time analysis of its progression, which in turn allows the motor to automatically stop when in a dangerous situation, preventing further damage to the machine.

#### **2.7.1.2 Vibration**

Vibration can be defined as an alternating oscillatory movement around a reference position. Due to the movement of different parts of a machine, and, more importantly, to the movement of rotating parts, vibrations are produced.

Vibration is a very good indicator of how well a machine is functioning as the degradation of components in a machine lead the motors and other actuators to try and compensate the losses, which involve tweaking parameters such as velocity, frequency, torque and force. A different set of forces actuation on the machine will produce a different vibration from normal functioning. One of the motives that makes vibration such a good parameter to monitor a machine's or part's condition is how quickly it responds to equipment aging and degradation([Jung et al.](#)). In fact, standard ISO

10816-8:2014 establishes normal vibration measurements for rotating and non-rotating parts, as can be seen in figure 2.12.

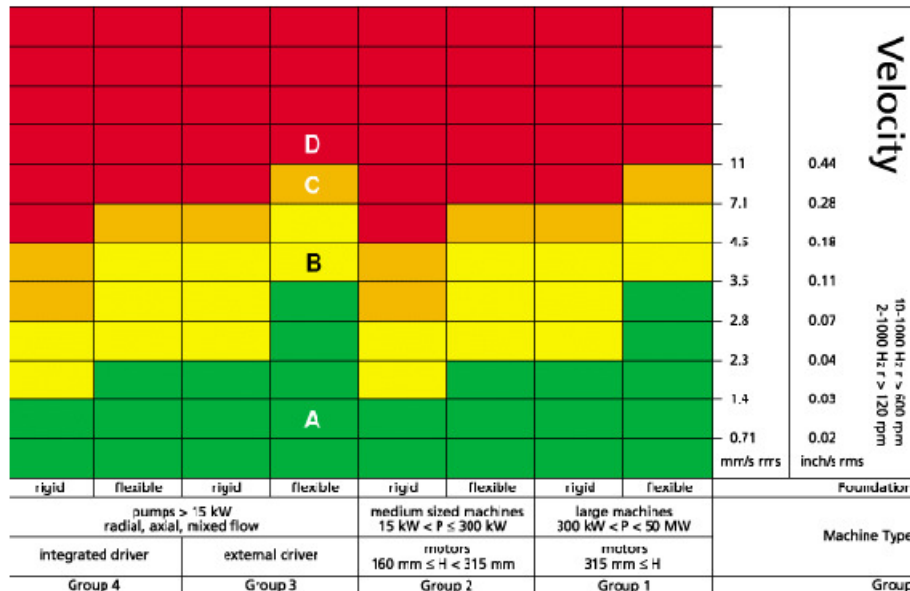


Figure 2.12: Standard ISO 10816-8:2014 (Yasir et al., 2019)

In order to use vibration as an indicator of failure, sensors able to measure it are needed. This has only recently been made viable, as sensor became smaller and easier to implement, without compromising the machine’s normal functioning, due to advancements of technology. One of these technologies are MEMS accelerometers, which stands for MicroElectroMechanical Systems, which are small inertial accelerometers that are able to detect changes to acceleration with high precision and at a very low cost. Compared to the previously available alternative, piezoelectric accelerometers, MEMS are much smaller, consume less power and are cheaper, making this the technology of choice for this kind of application(Jung et al.). Its small size allows it to be installed in tight spaces and without disturbing the machine’s functioning, its low power consumption allows it to run out of the power source of an IoT device, and its low price makes the implementation of systems depending on it more affordable and the replacement in case of malfunction less costly (Jung et al.). Bellow, we can see a chart 2.13 comparing typical characteristics of a MEMS sensor and a piezoelectrical one.



	Piezo Sensor	MEMS Sensor
Price	US\$ 300+	within US\$ 10+
Power	27mW	3mW
Size	1.97×0.98×1 inch	0.2×0.2×0.05 inch
Noise Density	700 $\mu$ g	4000 $\mu$ g
Resonance freq.	20 kHz	22 kHz
Accelerate range	10g	100g

Figure 2.13: Typical characteristics of a piezoelectrical sensor and a MEMS sensor (Jung et al.)

There are also different ways to analyze vibration data: time-domain analysis and frequency-domain analysis.

Time-domain analysis is the plotting of the signal intensity or corresponding acceleration coming from the sensor against time, which translates to a waveform being graphed, showing how the signal evolved through time. On figure 2.14 a) a time-domain graph can be seen.

On the other hand, frequency-domain analysis shows how much of each frequency is present in the vibration being monitored. In other to do this one of the transforms addressed on a previous chapter is used. The result of this analysis is what is called a spectrum of vibration. Frequency-domain analysis allows the isolation of certain frequencies associated with the phenomena or part being studied, which is helpful, for example, for monitoring different components of a machine using just one sensor. An example of this type of graph can be seen in figure 2.14 b).

Both of these analyses complement each other and give different perspectives on the vibration being monitored.

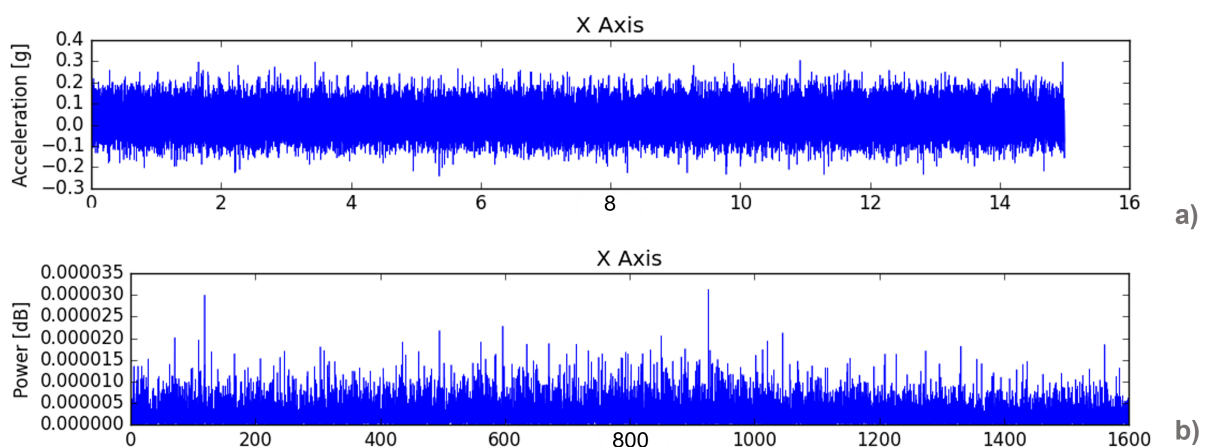


Figure 2.14: a) Time-domain graph b) Frequency-domain graph



## Chapter 3

# Case Study

In this chapter, the e-MOB machines of AMOB are the basis of this case study, whose aim is to determine the causes and how failure occurs.

The main causes of failure will be showcased along with the current maintenance strategy for the parts affected. From there, the focus will be on a severe undiagnosed problem that is happening at the time. Root cause analysis of this case will be conducted. Suggestions on how to avoid this problem and conclusions will be drawn.

This case will serve as the starting point for devising and testing a PdM system architecture, as it is the case where most data is available, as AMOB lacks any historical data about their machines from a PdM point of view.

### 3.1 The machines

AMOB intends on deploying this system on their e-MOB series machines. These are the flagship rotary draw bending (RDB) machines of AMOB. They are fully electric CNC machines.

This series has a tube diameter range from 6 *mm* to 225 *mm* and is able to make bends with radii as tight as the diameter of the tube.

Some of the series features include: tool multi-staking, for easy production of tubes with different radii bends and complex geometries; booster cart, to help achieve a better finishing and tighter bends; up to 8 times more energy efficient when compared to the previous flagships ([Pacheco et al., 2019](#)); high precision and a better human-machine interface (HMI), in big part due to the included proprietary AMOB 3D bending software.

e-MOB machines have 9 independently operated axis, all electrically powered, controlled by servomotors, as seen in the figure 3.1.

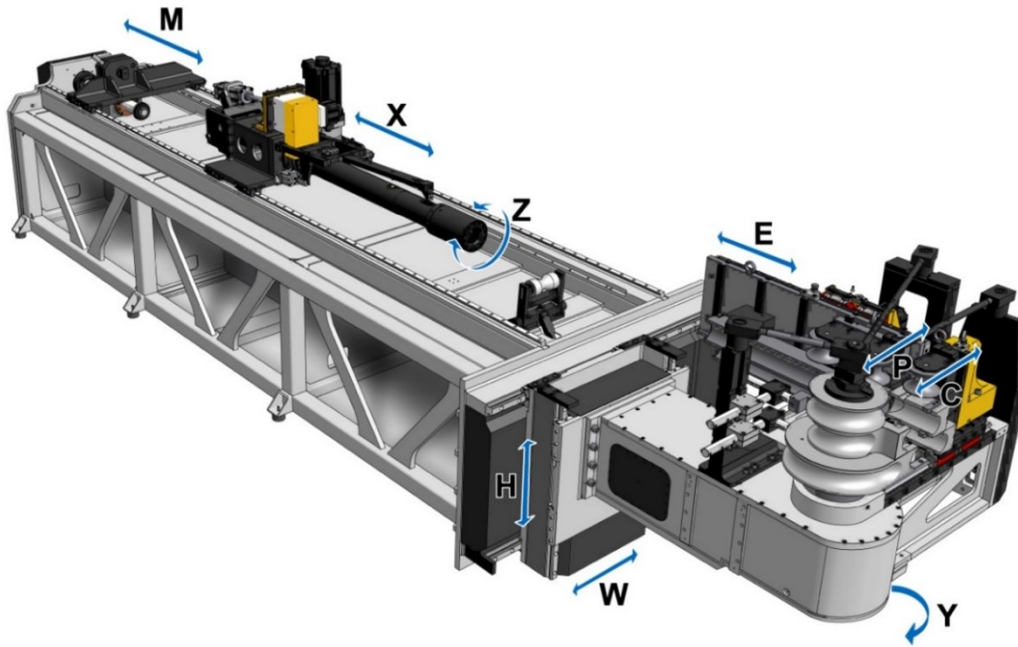


Figure 3.1: e-MOB's axis designation (AMOB, a)

Each of these axes correspond to the following movements:

- **Axis X**—Tube displacement carriage;
- **Axis Y**—Tube bending arm;
- **Axis Z**—Tube rotation plane;
- **Axis H**—Vertical shift for changing average radius;
- **Axis W**—Horizontal head shift for changing average radius;
- **Axis M**—Mandrel extraction;
- **Axis C**— Tube clamping;
- **Axis P**—Pressure die;
- **Axis E**— Horizontal pressure die follower.

## 3.2 Current maintenance strategy of e-MOB machines

The current maintenance strategy is entirely based on a mix of reactive and preventive maintenance strategies. This strategy consists of 8 main procedures, namely (AMOB, a)

- Lubricants list, capacities and corresponding information
- Lubricate ball recirculation systems
- Check the ball screws' automatic lubricating units
- Inspection of toothed belts and tensional procedure
- Check the position and fastening of limit switches
- Lubricate the positioner rack
- Restore the mandrel lubrication system
- Replace the lubricant of the bending axis gearbox.

In terms of any information or work done towards the implementation of predictive maintenance strategies it is nonexistent, with no historical data of any kind available. Very little information about service requests and previous failure of their machines. Since AMOB sells to clients all over the world, much of the maintenance and support is outsourced to third parties by the AMOB local or regional office. There is no standardization of communication between AMOB and these third party service providers. Occasionally reports are made by AMOB's regional or local offices and send to the headquarters but not for each occurrence. Most communication is informal (e.g. messaging apps) and not kept by AMOB. Root cause analysis is rarely done, with the most common practice being replacing the faulty part for a new one.

## 3.3 Failure pain points

From the available records of repair requests from customers and talks with the engineers in AMOB's client service department, the most common types of failure of its e-MOB machines were identified. It was concluded that the most common types of failure were:

- Wear of linear guides
- Wear of ball screws

- Mechanical damage of linear guides due to improper use
- Mechanical damage of linear guides due to the operator not identifying obstacles in the machine's path.

The failures mentioned in the two last points occur when the machine is operated in manual mode, where each task is inputted by the operator one at the time. Because, in this mode, the e-MOB machines have yet very few safeguards against collisions with themselves, collisions happen fairly frequently as described in the last point. One example of this is when the carriage unit (X axis) advances without returning the clamp die to its initial position after bending (Y axis to 0), causing the tube to collide with the clamp die, which causes forces that actuate on the X axis guides.

Another issue is when operations are not imputed in a feasible order, for example, changing the tool without first ordering the clamp die to open. This puts excessive loads onto the guides, which causes damage. This is the kind of scenario that fall under the third point.

AMOB's engineering team knows this and is already working on implementing more software-based solutions to this problem. As they are not the result of degradation or wear of the parts but miss use they have no relevancy from the point of view of this thesis. It is also important to mention that while the damage to the guides is the one that impacts machine functioning and therefore mentioned above, these failures also cause damage to other parts, but mostly cosmetic damage that does not impact the machine functioning, as can be seen in figure 3.2.



Figure 3.2: Carriage damage due to manual collision in 9611 e-MOB axis

This means that the main pain points in terms of wear are the linear guides and ball screws. Therefore, when choosing which components to monitor for failure, these would be the ones to start with, since they are the ones who would benefit the client the most in terms of lowering downtime due to failure. Therefore, the PdM system that is presented in this thesis will focus on one of these parts.

### **3.3.1 Linear guides**

Linear guides consist of two main components: a rail and a block that slides along it. Their purpose is to both guide the movement of whatever is mounted on the cart and to ease the motion of that same part, by reducing the resistance to the movement due to friction.

### **3.3.2 Ball Screws**

Ball screws are mechanical linear actuators that translate rotational motion, for example of a motor, into linear motion. These work by pushing a chain of balls inside the ball nut that track the outside of a helical screw inside the actuator per se, which causes the ball nut to move up or down, depending on which direction the motor is working. An alternative to this sort of working is the screw itself is powered by the motor, spinning, while the ball nut has that degree of freedom locked, being unable to spin. This forces the balls to track up or down the screw, pushing the ball nut up or down with it. In this case, AMOB uses this second solution, where the screw spins, and the balls track a closed circuit inside the ball nut, called a ball recirculating screw.

These mechanisms allow for a very efficient conversion of power from a motor into linear movement, with high precision and control. Its main disadvantages are its comparatively big size, as the actuator has to have space for the tracks for the balls, and that, although able to support heavy loads, it can be back-driven, meaning the actuator can force the balls to move. This is specially concerning when the motor driving it needs power to maintain a certain position, as a power outage can mean that the supported load will be dropped.

AMOB also uses HIWIN ball screws for their e-MOB machines.



Figure 3.3: Ball screw of the H axis of one of AMOB's machines

The servomotors AMOB uses to power these screws are almost exclusively OMRON 1S series with built-in EtherCAT® communications. These servomotors can be connected to a PC running Sysmac Studio via EtherCAT®, which allows for access to the working parameters set for the servomotor and data gathered by the servomotor itself, such as torque and velocity. This data can come in two ways: Process Data Objects (PDO), which relay real time data every time the servomotor is sampled, such as position; and Service Data Objects (SDO), which is a channel to relay sporadic information, such as error codes. This data is used by AMOB's own software to control the production process and can be exported for analysis if requested, in the form of data traces.

### 3.4 e-MOB 42 9611 with vertical axis problems

As mentioned before, it is important to understand what leads to failure before devising maintenance strategies. Also, a specific current case allows for the collection of data, although limited to what can be retrieved remotely with the capabilities the machine currently has, as it is overseas. Therefore, this thesis will focus on the parts affected by this problem on this machine, as a basis to test the systems here devised. The idea being that those can be applied to the rest of the machine in the future with small adjustments.



In this section a machine already in use at a client that was having problems will serve as the basis of this case study.

### 3.4.1 Machine's history

During the 12 months of this machine's operation, the first breakdown happened after four months of use. After that the ball screw failed three more times, being replaced each time with a similar one. The data extracted from it helps define what are normal function parameter values and what are failure indicating values, despite the fact that the failure was not due to wear and tear because it failed much earlier than expected.

This machine had completed 224944 parts and 901992 bends at the time of the second service intervention, with the main car going a total distance of 524.16 km. Mechanically, this machine is an e-MOB 52 incorporated with an automated cell that works 24 hours a day, 7 days a week.

Figure 3.4 shows the machine in question at the client's facilities, which was taken by the service team that revised the equipment during diagnostics.



Figure 3.4: e-MOB 42 9611 at the client

This specific model is not capable to perform bends in the opposite direction, since the head of the machine can not rotate on itself, like the AMOB 2Bend machines.

This particular machine was equipped with just one tool, the OD 25x1,5 CLR 2D, where OD stands for the outer diameter and CLR for center line radius.

### 3.4.2 Problem

The customer contacted AMOB as they noticed increasing noise when the H axis was triggered and a sudden "hiccup" when passing a certain point in that axis. The spindle of the ball screw also looked very worn out.

AMOB observed a very abrasive atmosphere at the plant when they replaced the entire ball screw. As mentioned before, up to June 2021 this ball screw had already been replaced 3 times, with an average life for each ball screw of around two months. This last time, for the failure in July, AMOB is planning on replacing the ball screw for a different ball screw, namely a Bosch Rexroth FEM-E-C 40x10Ex6-6 with T9 tolerance ball screw. This ball screw has a dynamic load rating of 86500 N (around 8826.5 kgf) and a static load rating of 132200 N. This screw was chosen as there was no room within the machine for a higher diameter screw and, even though it is significantly more expensive than the original ball screw, it has almost double the dynamic load rating.

### 3.4.3 Approach

In order to do the root cause analysis of this problem, a thorough characterization of the parts involved in this axis movement will be done. All possible causes will be listed and their plausibility evaluated based on the analysis of one of the replaced ball screws, data from the machine and project data. In the end, all causes that contributed to this problem will be listed.

### 3.4.4 H Axis

The head of the machine, which includes all the components responsible for bending the tube per se, is mounted on a carriage capable of horizontal (W axis) and vertical (H axis) movement. This is needed in order to bend the tube with different tools for different radii, therefore allowing for quick and easy alternation of the bending tools.

While some configurations of this machine include a gas spring to aid in this movement, the machine experiencing this problem does not include one. The same ball screw is used in both the

machine with the gas spring and without. The inclusion of the gas spring is to save energy, being offered as an option to the client.

This machine also does not include a mechanical brake in this axis, therefore needing the servomotor to work continuously to sustain the machine's head in any given position.

In this axis, high rigidity ball type linear guides from HIWIN's HG series, namely the HIWIN HGW35HCZA, are used, as can be seen in figure 3.5. This series main feature is being able to withstand high loads and having a high rigidity in all directions. It is also a self-aligning model, which minimizes installation errors. This configuration in particular (HGW35HCZA) is a combination flange (drilled and tap holes) type standard block with top or bottom mounting, medium preload, for super heavy loads with normal particle ingress protection (HIWIN, 1998b).



Figure 3.5: Linear guide in one of AMOB's machines

In respect to maintenance the manufacturer recommends checking the grease every 100 km or every 3 to 6 months (HIWIN, 1998b). As for the nominal life of the guide, the manufacturer indicates the following equation for its calculation:

$$L = \left( \frac{f_h \times f_t \times C}{f_w \times P} \right)^{\frac{10}{3}} \times 100 \quad [km] \quad (3.1)$$

L : Nominal life	$f_h$ : Hardness factor
C : Basic dynamic load rating	$f_t$ : Temperature factor
P : Actual load	$f_w$ : Load factor

This gives us a conservative estimate of the lifetime of the guide, as in the manufacturer's tests this value was exceeded without incurring in damage, such as fatigue flaking, when this value was surpassed. The manufacturer provides a nominal life calculator on their website, which was the tool used by AMOB when dimensioning this part. Considering the maximum value of velocity reached  $65 \text{ mm/s}$ , since the average maximum velocity reached on the  $100 \text{ mm}$  to  $145 \text{ mm}$  movement of the H axis is around  $60 \text{ mm/s}$ , and awarding this value a slight safety margin, an accelerating time of 0.45 seconds, calculated by dividing the number of indexes when the velocity goes from 0 to  $60 \text{ m/s}$  and dividing it by the number of samples collected per second, averaging several of these values, with a stroke of  $45 \text{ mm}$  and a frequency of 2 times a minute, value considering also the data trace conditions and values.

Using these values, a nominal life of  $15513.56 \text{ km}$  or  $1436440.5$  hours was calculated, with a static safety factor of 9.84.

This axis is controlled by a brushless servomotor, namely OMRON's R88M-1M2K020C-BS2 with  $9,55 \text{ Nm}$  of rated torque, with an angular reducer attached, Dynabox's 45 i=5,2 J<5 H1, which transmits the motion to the carriage by means of a ball-recirculating screw, a type of ball-screw, which allows for rotational motion to be converted into linear with very little friction (AMOB, a). This particular machine uses a HIWIN R40-10K4-FSCDIN ball screw to transfer movement to this axis. This assembly can be seen in figure 3.6, with the ball-recirculating screw marked as 1.

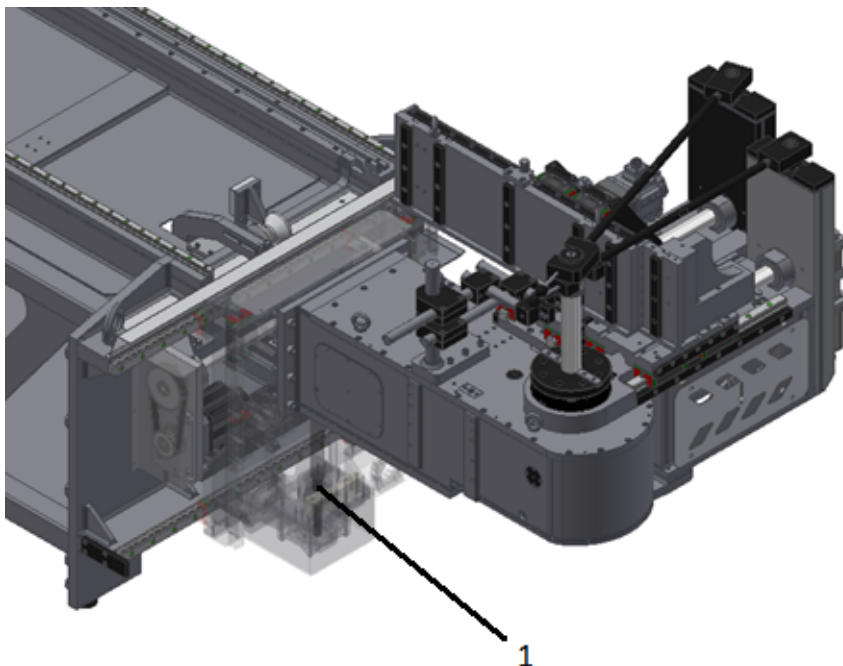


Figure 3.6: Bending head carriage with H axis ball-recirculating screw marked as 1 (AMOB, a)

This screw is held by two bearings at the lower edge, acting as a fixed joint, with the exception of the rotation of the screw. The upper edge is free. In figure 3.7 the drawing of the assembly of the ball screw in this machine can be seen.

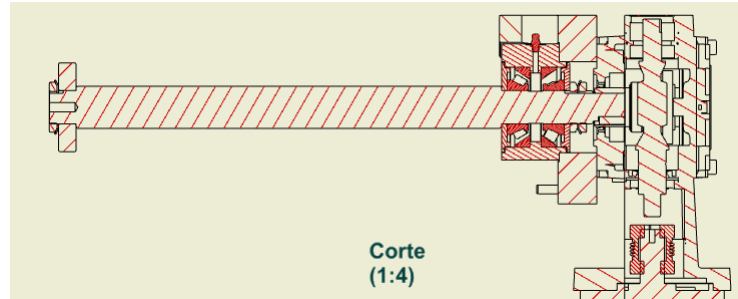


Figure 3.7: Engineering drawing of the assembly of the ball screw in the H axis of e-MOB 52

The current maintenance plan for this system is for each of the 4 linear guides of the vertical movement of the machine's head, which can be seen in the picture below, to be lubricated every 170 work hours or once a week with grease Kernite NGLI 2 (AMOB, a).

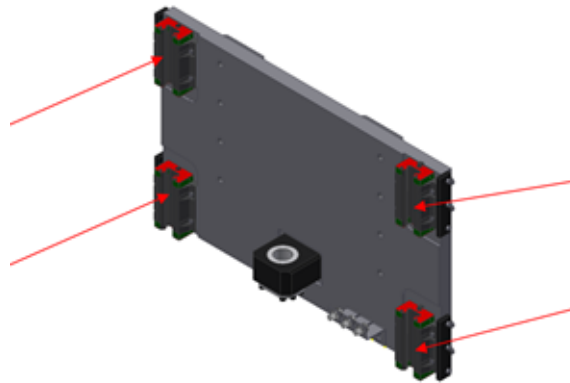


Figure 3.8: Places to lubricate when doing H axis maintenance (AMOB, a)

The ball screws automatic lubricating unit should be checked at the same frequency as the linear guides, with the amount of grease dispensed being adjusted based on the visual inspection of the ball screw (AMOB, a).

The automatic lubricating unit used by AMOB is Oberrecht's simalube® automatic lubricator. These long-term stepless automatic lubricators consist of a cylinder that is filled with grease or oil with a gas generator, a dry cell that produces hydrogen, that pushes a piston, which in turn pushes oil or grease out, at a set rate. This rate can be manually adjusted using an allen key to rotate the rate indicator at the top of the cylinder, as can be seen in the figure 3.9.



Figure 3.9: Lubrication unit from one of AMOB's machines from two perspectives

This setting is dependent on the amount of grease needed daily, resistance from grease lines, ambient pressure and the viscosity of the grease used.

### 3.4.5 Ball screw analysis

When one of the replaced ball screws came in from the client it was inspected for clues into what the problem might be.

The ball screw was not broken or noticeably bent out of shape and no balls appeared to be missing. The gaskets on the nut seemed to be undamaged.

Visually, there was more than enough lubricant on the ball screw, even though this lubricant was very dirty, as it had a sparkly dark grey tone to it, instead of its normal cream color. There were also significant signs of wear on the tracks, with some of them being almost entirely removed, as can be seen in figure 3.10.



Figure 3.10: Replaced ball screw with severe wear signs

When manually moving the ball screw nut up and down increased difficulty in moving the nut with a downwards force applied was noticed. More noise and trepidation could also be detected. When an upwards force was applied none of these signs were observed.

### 3.4.6 Failure causes

There were identified four main areas where mistakes may result in this problem arising:

- Faulty Parts
- Lubrication
- Project
- Assembly

These were then broken further down into what specific actions may have contributed to or caused the problem. Those actions are codified in the Ishikawa diagram as shown in figure 3.11.

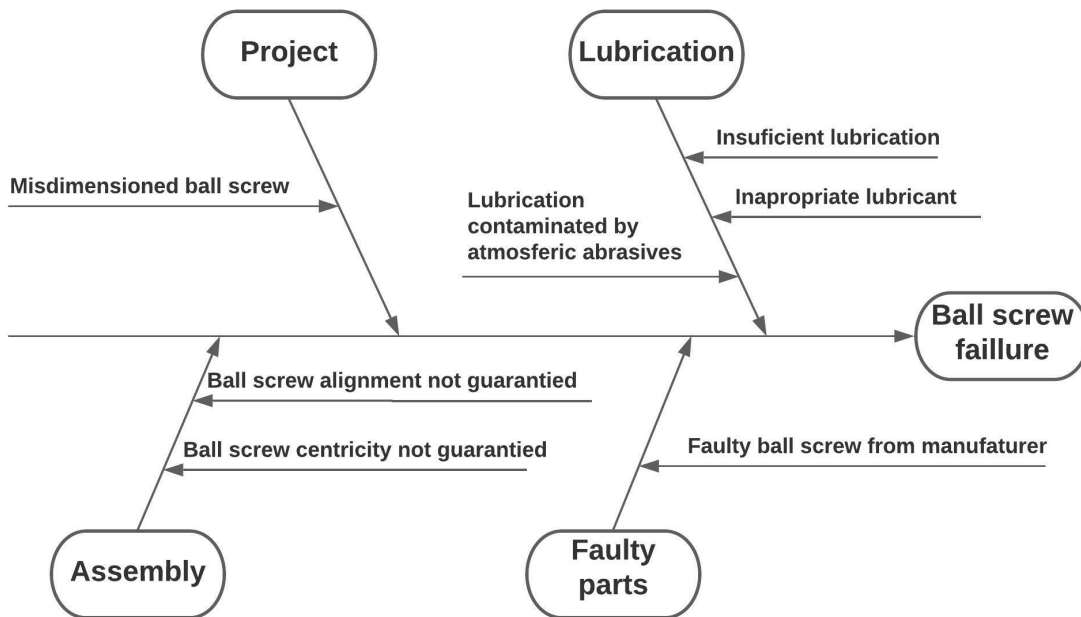


Figure 3.11: Ishikawa diagram with the possible causes of the ball screw failure

### 3.4.6.1 Lubrication

As mentioned before incorrect lubrication leads to higher wear of moving parts. In the case of this ball screw, there are three things that may have been done incorrectly in terms of lubrication that could explain the failure: not enough lubrication, not the right lubricant and contamination of the lubricant.

As for the lack of lubrication, that can be excluded by virtue of the visual analysis of the screw, where there could even be observed excess lubrication, which would not lead to this problem.

As for the lubricant itself, not only is this the lubricant used in all of the ball screws of AMOB machine's as it has the recommended viscosity by the manufacturer for the use. Another possibility was the lubricant interacting with the material of the gaskets leading to their degradation and abrasive particles getting in. Again, visually, the gaskets show no sign of damage.

As for the particles that contaminated the lubricant that were seen in the visual inspection leading to increased wear, it is unlikely that it has any significant impact. Firstly, because the screw nut is equipped with a NBR finger wiper, which removes coarser material thanks to its hard plastic fingers, and finer particles have almost no impact due the lubricant grease. Also, if abrasive particles present on the factory atmosphere were the culprit, all the other axis in the machine would show similar problems or at least increased wear which has not happened.



Therefore, the problem is unrelated with lubrication. This also means AMOB's hypothesis for the cause of the problem is wrong.

### 3.4.6.2 Faulty parts

This is the easiest option to discard. If the problem was by chance having one ball screw that left the factory defective, replacing it should have resolved the problem. The other possibility would be that it was a wider problem that the manufacturer had all of their ball screws for this series were somehow defective. If this was the case the manufacturer would quickly realize it and issue a recall, which did not happen.

Therefore, it is certain that a defective ball screw did not cause this failure.

### 3.4.6.3 Project

In terms of project mistakes that may lead to the failure of the ball screw it is limited to the choice and dimensioning of said ball screw. With that said, dimensioning any part is a big challenge, where there is a lot of margin for mistakes.

Its highly likely that there was poor dimensioning as the flattening on the replaced ball screw that was analyzed are associated with excessive load. Since the load in this axis is not variable, the ball screw must be undersized.

It is therefore important to know how AMOB dimensioned this ball screw in the first place, in order to understand where mistakes and misassumptions might have happened.

To choose the screw AMOB used a sizing tool provided by HIWIN in their deutsch website ([Hiwin, 2021, https://www.hiwin.de/en/service/auslegungstool](https://www.hiwin.de/en/service/auslegungstool)). Considering that the machine's head plus tools weighs 1070 kg, AMOB decided in this tool to leave a safety margin and consider 1500 kg. The spindle was set to rolled and the nut type to flange. The spindle length to 300 mm, the free length to 280 mm, and the maximum stroke to 240 mm. The service life was set to 125 km. All the other parameters were left at standard values. This indicated them the screw to use, R40-10K4-FSDIN.

Even though this gave them what screw to use, it did not indicate its service life, only that is greater than 125 km. So, in order to know this value AMOB used the life calculation tool that HIWIN provides on their taiwanese website ([Hiwin, 2016, https://www.hiwin.tw/support/bs\\_lifecalculate.aspx](https://www.hiwin.tw/support/bs_lifecalculate.aspx)). However, the ball screw chosen is not available in this tool, so AMOB used the 38-10K4-FSCDIN. Although this ball screw is dimensionally identical to the one the 40-10K4-FSCDIN, its dynamic load rating is of 5660 kgf. Compared to the dynamic load rating

of the 40-10K4-FSCDIN of 4550 *kgf*, the dynamic load of ball screw chosen for the service life calculation is almost 25% higher. When calculating the service life for the ball screw that was actually installed, based on the formulas available in the HIWIN catalog and keeping all the parameters (loads, cycle, configuration) the same as the ones used by AMOB, which can be consulted in appendix D, the service life obtained was of approximately 160km. This value is significantly lower than the 295 *km* calculated by AMOB (see appendix D).

However, this alone does not explain fully the failure, as even when considering the correct service life, the ball screw should last at least a year before failing. This considering that over the course of 18 days the distance traveled in this axis at the client was recorded and was 5330 meters, which averages at about 300 meters per day.

#### 3.4.6.4 Assembly

The main factors in the assembly process that can result in accelerated wear and failure of the ball screw are the center of the screw being misaligned with the center of the nut, causing a radial force on the screw that may lead to seizure and the mentioned wear, and the alignment of the screw, meaning its parallelism towards the plane of movement of the head (the plane of the H and W axis, i.e. vertical and horizontal movement) and its perpendicular plane that also contains the H axis. These planes are represented on figure 3.12 in blue and red respectively.

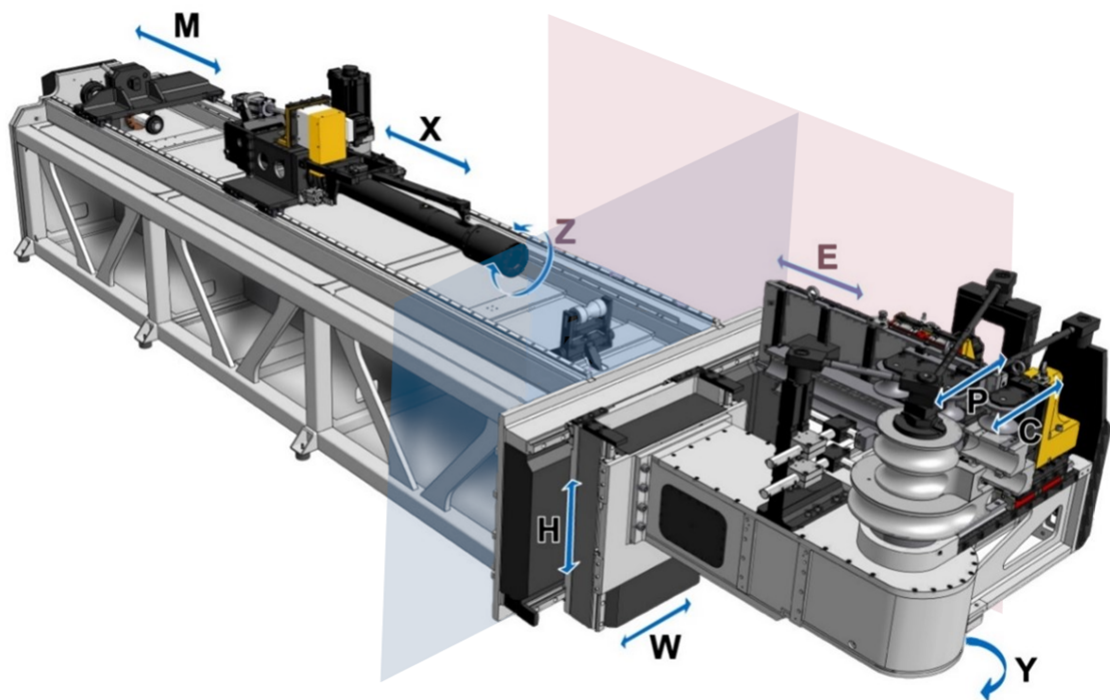


Figure 3.12: Machine with the parallelism planes colored

### Alignment procedures

The current method used to assure both of these conditions is done when mounting the machine's head in its body. A overhead crane holds the panel that will support the machine's head. While the guide blocks of said panel are already mounted on the guide rails in the machine's body, therefore imposing the trajectory of the movement, the assembly operator lowers the plate until the nut snaps into the nut carrying part on the machine's head panel. If when this happens there is noticeable movement of the screw the assembly operator tweaks the screws that hold the support of the spindle of the ball screw. This process is repeated until there is no movement of the spindle or the operator deems that movement small enough. This procedure can be seen in figures 3.13 and 3.14. Doing this for any one position should assure the centricity of the nut and the part on the machine's head panel.

As for the parallelism of the spindle with the guide's imposed trajectory, a similar procedure must be conducted on a second point along the ball screw. To increase the accuracy of these procedures, these points must be as far away from each other as possible, as for the same misalignment, the movement of the spindle will be greater with increasing distance. Therefore, it is common practice to conduct the first procedure (assures centricity) with the ball screw nut the bottommost position of the spindle and the second procedure (assures alignment) at the upmost position. This second procedure works by assuring the centricity in two different positions on the same axis, which means the distance between the guides and the spindle is the same at two different points, therefore assuring their parallelism. Any misalignment with the plane perpendicular to this one (red plane in figure 3.12) would also produce movement of the screw in this second procedure, being the parallelism of the screw towards this plane also assured in this procedure.



Figure 3.13: Machine during the alignment procedures

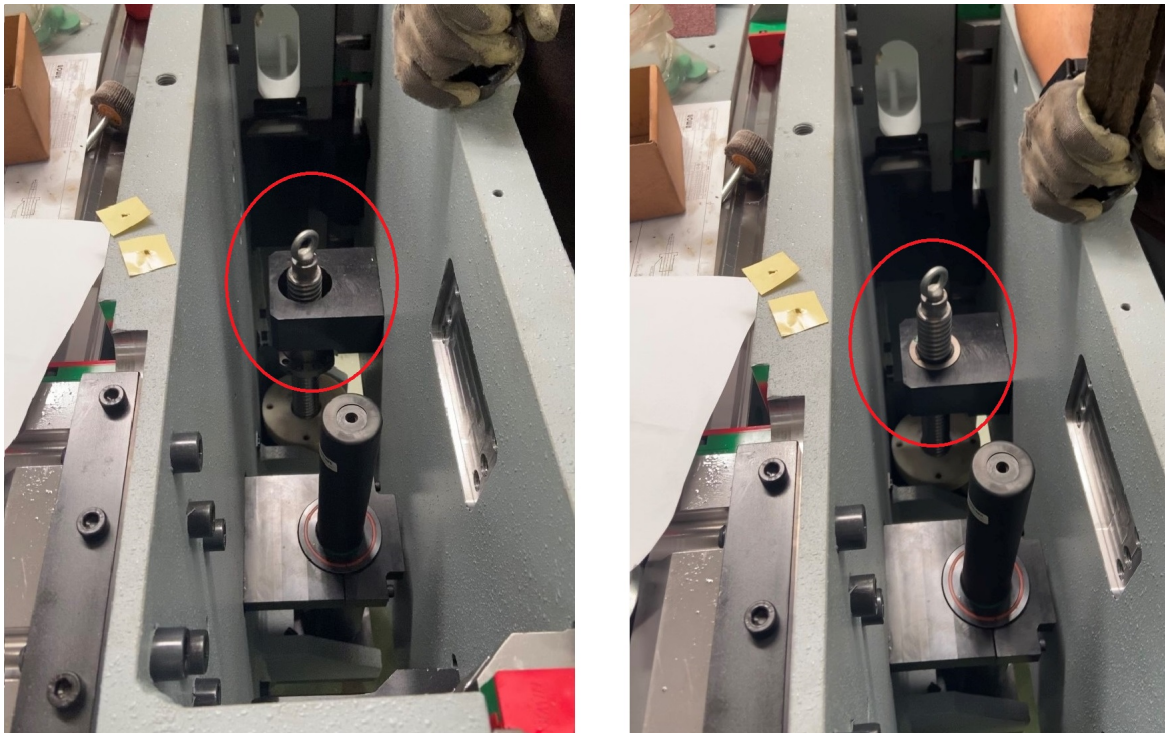


Figure 3.14: Ball screw during alignment procedure, before and after the panel is lowered

There are two big flaws with this procedure. First of all, when tweaking the position of the screw for the second procedure there is no guarantee that the centricity of the first procedure is not lost. This will lead to the screw still not being parallel to the guides and the red plane despite the procedures. Sometimes the procedure is repeated in an intermediary position if the operator thinks it is necessary, but this does nothing to avoid this issue. Secondly, because the length of the screw is not very long (around 300 *mm*) and the validation is dependent on the sensibility and experience of the person assembling the machine, there is no guarantee that all screws that pass these procedures are within the tolerances defined by the manufacturer.

### Screw tolerances

For the model of screw used by AMOB there are two available tolerance classes available. These classes are defined by the maximum admissible path deviation over a 300 *mm* length. For this ball screw they are: T5, which corresponds to 0.023 *mm*, and T7, which corresponds to 0.052 *mm*. AMOB in every substitution used the T7 tolerance ball screw.

### Checking misalignment

In an operation conducted with the service office, the machine's head was disassembled from the ball screw, to a configuration similar to the one shown in figure 3.13.

A dial gauge was fixed to the metal plate where the machine's head was assembled with its tip against the ball screw nut. The screw nut was manually rotated so it would travel up 200 *mm*. The metal plate was risen the same distance using the overhead crane. The results were validated by zeroing the dial gauge with the nut at the same position and lowering the plate just enough so some of the nut enters the nut carrying piece. The dial gauge with the tip at the bottom of the nut's body should indicate the adjustment of the screw so it is centered with this piece. This adjustment is equal in value to the misalignment of the ball screw. This second procedure can also give the misalignment in any plane other than the plane of the guide rails (blue plane in the figure 3.12).

For the case study machine the misalignment in that plane (over 200 *mm*) was 0.210 *mm*. This is clearly in excess of the tolerances of the ball screw. On the perpendicular plane (red plane in figure 3.12) it was just 0.008 *mm*, a insignificant amount well within the tolerances that can be disregarded given the scale of the misalignment in the other plane.

In fact, this misalignment is so severe that the fault for the problems of this ball screw can be mainly attributed to it.

### 3.4.7 Conclusions

As seem before, it can be concluded that bad dimensioning and misalignment of the ball screw were the causes of its repeated failure.

How each of this problems contributed to the ball screw failure can be seen by analyzing this graph made by ball screw manufacturer.

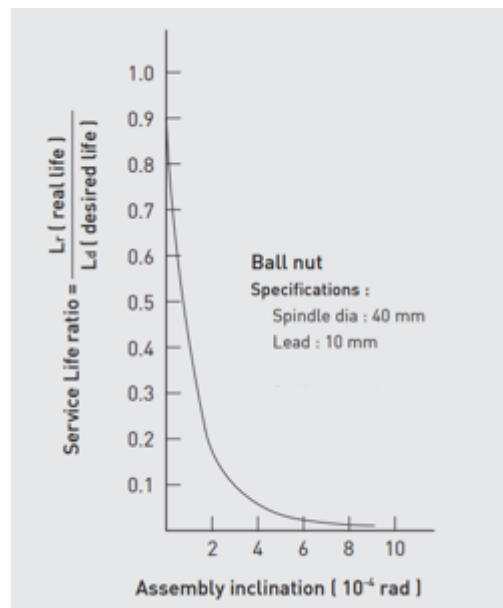


Figure 3.15: Service life factor as a function of misalignment, adapted (HIWIN, 1998a)

A misalignment of  $0.21 \text{ mm}$  over a  $200 \text{ mm}$  path is the equivalent of a  $10.5 \times 10^{-4}$  radians, which is a misalignment so severe that falls out of this graph. Still, it can be assumed that the service life factor is lower than 0.1. Considering that the correctly calculated life is about  $160 \text{ km}$  and not  $295 \text{ km}$  as previously thought, the service life of this ball screw is lower than  $16 \text{ km}$ . Given that based on the measured average of  $300 \text{ m}$  per day traveled in this axis, and that the machine operates every day, including holidays and weekends, it would travel about  $9 \text{ km}$  a month. This means the ball screw should fail around the two month mark, which is exactly what was happening in reality.

While the choice for a ball screw with a larger dynamic load rating and inclination tolerances (Bosch Rexroth FEM-E-C 40x10Ex6-6 with T9 tolerance) for the next repair addresses to a great extent the problem, until the ball screw is properly realigned the service life of this ball screw will always be significantly lower than what is indicated by the manufacturer and what is expected by the client. Based on these findings AMOB also realigned the ball screw when they changed it for this Bosch ball screw.

## **Chapter 4**

# **Architecture of the PdM system**

In this chapter the architecture of the predictive maintenance system will be outlined, from the sensorization and data acquisition to data storage. Different options will be presented and the reasoning behind the choices made explained.

### **4.1 Introduction**

In broad terms, a PdM system can be broken down into three parts: sensorization and other data gathering equipment, data processing and data storage.

Between these parts, there needs to be communication. That requires data formats to be chosen and standardized across the system and communication protocols to be chosen.

For these systems, there are two main types of implementation: local or on the cloud. A local system runs in the machine itself or in devices installed for that purpose parallel to the machine, and data is stored locally. While they may be accessed remotely, these systems are designed to provide information to the user on site. This is similar to the solution currently available in AMOB's machines, which consists of TeamViewer 11 license that is installed in every e-MOB machine. This is a remote desktop software that allows for remote control of the computer that controls the machine. This solution has several drawbacks such as: the data is only available while the machine is turned on and online, due to the lack of storage of the controlling computers data can only be obtained at request and not continuously, the data that can be obtained is limited to the outputs of AMOB 3D bending software and the user interface can't be optimized for the user and platform that is being used. This means this solution is currently only available for AMOB to help troubleshoot any problems a customer might be experiencing.

What is intended with this project with INEGI is to implement a cloud based solution, which means data storage and possibly part of data processing is done in servers connected to the internet. This opens up possibilities such as continuous data gathering, which makes PdM possible, and tailor made customer solutions, which allows AMOB to offer its clients machine's metrics and analysis anywhere as a service.

The objectives of this project are:

- Create a PdM structure
- Monitor the process
- Process optimization

This thesis focuses on the first of these goals.

## **4.2 Data gathering and sensorization**

In this section it will be analyzed what data can be retrieved from the machine as it is. For relevant data that can not be obtained this way, sensorization systems will be designed.

### **4.2.1 Available data**

From the machine's software several operational variables can be obtained from the servomotors. The most relevant of these variables are: command position, actual position, command velocity, actual velocity and torque. These are available for every axis as all axis are controlled by servomotors. The servomotor communicates them as process data objects (PDO), which is a protocol for real time data, which the the servomotor controller software, which is integrated in AMOB's 3D bending software, can turn into an float variable. These variables can be written and exported either as comma separated value file (.csv) or as Structured Query Language (SQL) data.

Apart from this data, by implementing counter blocks in AMOB's 3D bending software functional block diagram (FBD) that controls the servomotors, the number of hours of use, the distance traveled in a given axis and the number of cycles done can also be obtained. The date and time elapsed since the last maintenance procedure can also be obtained given that a second counter block for time is present that is reset every time the machine is serviced.



## 4.2.2 Sensorization

Besides the available torque data, as mentioned in section 2.7.1.2, vibration data is also useful for PdM of rotating parts. For that, a sensor capable of recording vibration signals is needed. The most commonly used type of sensor for this application is accelerometers. These sensors offer comparable performance to acoustic emission sensors while being less susceptible to noise from the surroundings. Withing accelerometers, as explained in chapter 2.7.1.2, MEMS sensors are much more advantageous for this kind of applications than piezoelectric sensors.

Because, as will be discussed later in this chapter, the backbone of the PdM system will be a Raspberry Pi 4, the solution considered was on which was compatible with this device. In a 2017 paper titled “Development of vibration spectrum analyzer using the Raspberry Pi microcomputer and 3-axis digital MEMS accelerometer ADXL345” (Iwaniec et al.), the authors study the feasibility of using a ADXL345 sensor in conjunction with a raspberry pi in vibration analysis, with positive results, as the rig is able to do real time accurate analysis of the signal in both domains.

### 4.2.2.1 Sensor characteristics

In terms of characteristics, the ADXL345 is an ultralow power device, consuming only  $23 \mu A$  in measuring mode at  $V_S = 2.5V$ ; high resolution, 13-bit full resolution at  $\pm 16 g$  (10-bit fixed resolution) and a rated sensitivity of  $3,9 mg/LSB$  across the measurement range ; thin and light MEMS sensor that connects via I2C or SPI (3 or 4 wires) (Devices, 2013). This sensor is also AQEC standard certified, meaning it can be used in defense and aerospace applications, and has extended industrial temperature range from  $-55 C$  to  $+105 C$  (Devices, 2013). Its output is formatted as 16-bit twos complements and has a FIFO buffer to minimize host processor activity, in this case the load on the Raspberry Pi’s processor (Devices, 2013). It measures just  $3 mm \times 5 mm \times 1 mm$  and can be configured for 4 different measuring ranges:  $\pm 2 g$  ( $3,9 mg/LSB$ ),  $\pm 4 g$  ( $7,8 mg/LSB$ ),  $\pm 8 g$  ( $15,6 mg/LSB$ ) and  $\pm 16 g$  ( $31,2 mg/LSB$ ) (Devices, 2013; Iwaniec et al.). Based on the paper entitled “Failure Diagnosis System for a Ball-Screw by Using Vibration Signals” (Lee et al., 2015), it can be presumed that, based on the range of frequencies establish in this paper for ball screw vibration and also their relationship to its speed, that this sensor capabilities in terms of range of frequencies detected are well beyond the necessary for this application, since the maximum speed of the ball screw is 9,6 rotations per second or 576 RPM.

All things considered, the ADLX345 sensor was chosen for the collection of vibration data of the H axis ball screw. Suppliers were researched, as the sensor used in this process was purchased ElectroFun, a local technology shop, for 5,85€ plus VAT (7,20€ including VAT). However, if the intention were to outfit all axis of the machine with this sensor or multiple machines this supplier offers progressive quantity discounts.

#### 4.2.2.2 Sensor set-up

As mentioned before there are two different ways of connecting this sensor to the raspberry pi. I2C is the simpler of the two to code for, which can be done without using any pre-existing libraries. It also only needs 2 wires to communicate, making the wiring process very simple. However, the ADXL345 is limited to 2 addresses, which is what identifies each I2C device in communications, therefore limiting it to a maximum of 2 of these sensors per Raspberry Pi. Another disadvantage of this standard is its bus speed of  $100\text{ kbit/s}$ , which can be altered in the raspberry and in the sensor to the  $400\text{ kbit/s}$  fast mode. Because when connected via I2C the sensor does not use its FIFO chip, for sampling frequencies over  $200\text{ Hz}$  (for fast mode) and  $50\text{ Hz}$  (for standard mode) the signal becomes unusable due to digital noise. The second sensor was used as a dummy to reduce the interference from noise and vibration by other components. In figure 4.1 a diagram on how the sensors were connected can be seen. The main sensor was mounted on the nut carrying part and the dummy close by on the machine body. Both sensors were screwed using the holes that come drilled into them and the bottom part had to be isolated using tape, as short circuiting certain circuits with the machine's body was a hard to diagnose problem that occurred during testing. To connect the sensor the pins were soldered onto the sensor and custom made cable with 8 female to female jumper wires, with the needed with approximately 1.5 meters was made (in order to allow for both SPI and I2C connections).

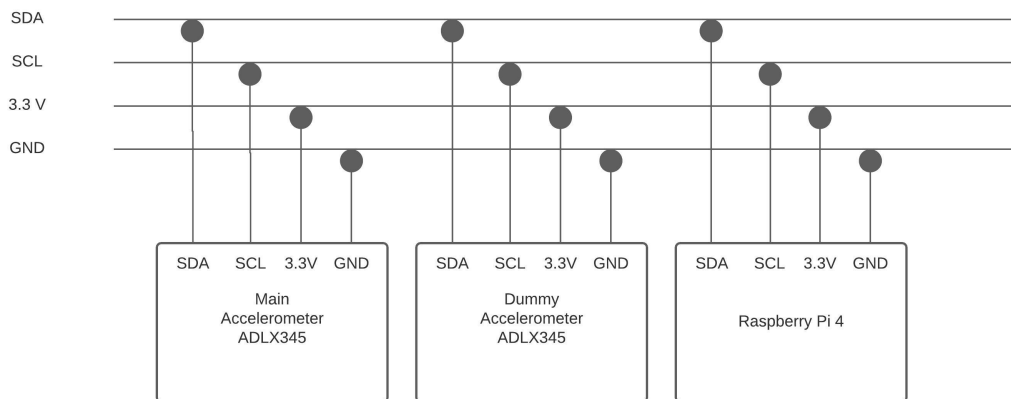


Figure 4.1: Wiring diagram of the ADXL345 sensors connected via I2C

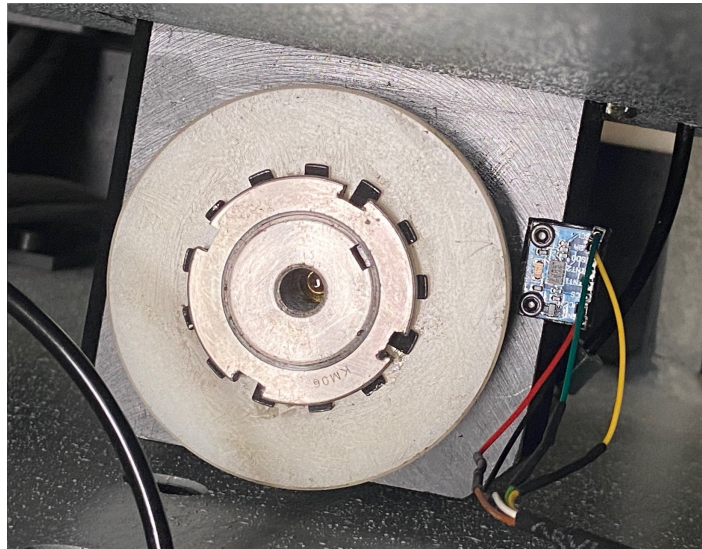


Figure 4.2: Main sensor mounted on the nut carrying part of the H axis

As an alternative, the 4 wire SPI connection was also tested. This allows the verification whether there are frequencies above that 200  $Hz$  threshold that are of importance, by utilizing the full 1600  $Hz$  capacity of this sensor. This is more complex to code for, therefore the code for this connection used as a basis an open source library posted on GitHub ([nagimov, 2020](#)). Because this library is made for just one sensor only the main sensor was connected via SPI. The position of the sensor is the same as via I2C. The connection to the raspberry itself is also more complicated as can be seen in wiring diagram of figure 4.3.

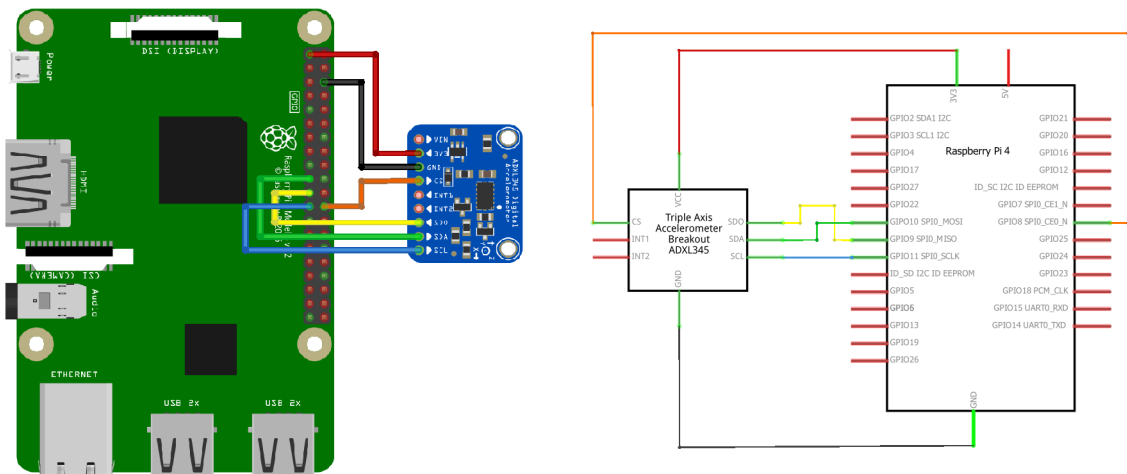


Figure 4.3: Wiring diagram of the ADXL345 sensor connected via 4 wires SPI ([nagimov, 2020](#))

### 4.2.2.3 Sensor code

For coding the language chosen was python. This language was used because it is a language commonly used for raspberry pi applications since its operating system (OS) has a python Integrated Development Environment (IDE) built in (Thonny). It also has a lot of open source libraries available, which facilitate the development of new software.

For the sensor connected via I2C smbus, which stands for System Management Bus and manages data between the sensor and the raspberry, time and datetime libraries were used. The first step was creating variables with all the hexadecimal codes to set the sensor parameters (i.e. scale, bandwidth and addresses). Right after functions for setting each parameter were created. These will bridge convert the user input into the correct hexadecimal code to send to the sensor. This are useful for the code that was used to test the sensors implementation and gather the data shown in the next chapter but for the PdM they would be set to a fixed value. The same applies to the next section of the code, the user inputs. This allows the user to set the sensor parameters when running the code by typing the value at the prompts at the command bar. Next the hexadecimal codes corresponding to the inputs are sent to the sensor. There is a verification and confirmation that these values were set correctly.

Having implemented the code and installed the sensor in the machine the data acquisition itself can start. This was programmed using a loop that probes the sensor for data on each of its axis (x, y and z; hexadecimal codes 0x32 to 0x37) every time the time between the last the loop ran and current time is bigger than the period set when setting the sampling frequency. This loop runs while the time is inferior to the loop end time, which consists of the time when sampling started plus the duration of the run. The data from the sensor, along with a timestamp each time a sample is taken are stored in a list. The values on this list are the outputs of the sensor. Those values are then converted into acceleration values in  $g$ 's.

The data from each of the sensors is then stored on separate .csv files.

This code was based on what the publication [Cassel Barbosa et al. \(2020\)](#) and is in full in appendix [A](#).

As for the SPI connection a entirely different script was needed. As mentioned in the section [4.2.2.2](#) this code was based on nagimov's ([nagimov, 2020](#)) code. After downloading the library and importing it there is only need to set the length of the run, the sample rate and the name of the output .csv file. The f' at the beginning of calling function allows the code to substitute the variables within braces to be called and not just read as is, as can be seen bellow.

```
os.system(f'sudo adxl345spi -t {length_s} -f {sample_rate_Hz} -s {name}.csv')
```

The full SPI code can be seen in full in appendix B.

## 4.3 Data Processing

For the data obtained by both types of connection the processing is very similar. For this processing, three libraries are crucial: numpy, matplotlib and pandas. These are all part of SciPy, "a Python-based ecosystem of open-source software for mathematics, science, and engineering", as it cites in its official website ([Scipy, 2021](#)).

Numpy consists of a package for mathematical and scientific computing. One of its main advantages is the possibility of working with arrays instead of lists, which are more efficient when working with the data. In fact, one of the most attractive features of this library is its efficiency. This library is free to use.

The matplotlib library is a python library that allows data to be plotted, in a similar fashion to the plotting tools used in Matlab, a commonly used software for data analysis in scientific research. Not only this library allows for the data to be presented in many different ways right within a python script, it is also free.

Pandas is another free and open source library that is used for data manipulation and source data analysis. It is important in importing and exporting data, as it has functions that easily convert data from one format to another.

As for the data processing itself, time domain, frequency domain and time-frequency domain graphs were instantly plotted after the data is acquired, as integrated in the data acquisition scripts. Each type of graph is plotted in a separate figure, which can be saved as .png files. This allows for a quick and easy way to visualize the data, for quick troubleshooting of any problems.

For the time domain graph the signal is simply plotted against the timestamps of each point.

As for the frequency domain graph, the data has to be transformed first. For that, the fast Fourier transform (FFT) is used. This is an algorithm that calculates the discrete Fourier transform (DFT). This transform converts a signal from its original domain, whether it is space or time, into the frequency domain. It decomposes the signal into its frequencies, allowing for a quick visual perception of the influence of each frequency in the signal. The FFT does this rapidly, being very popular in science and engineering uses.

In python, the implementation of the FFT is done through the FFT function in numpy:

```
(numpy.fft.fft(a[, n, axis, norm])
```

This function computes a one dimensional discrete Fourier transform. It takes as inputs an array consisting of the signal,  $a$ , and, optionally, the length of the axis of the output,  $n$ , meaning the frequencies up to which the DFT is calculated, the axis of the input array where the signal is stored,  $axis$ , and the  $norm$ , which basically serves to scale the output.

This numpy function defines the DFT as:

$$A_k = \sum_{m=0}^{n-1} a_m \exp\left\{-2\pi i \frac{mk}{n}\right\} \quad k = 0, \dots, n-1 \quad (4.1)$$

As for the time-frequency domain, there is no need to further transform the data, as this is just the frequency data, that resulted from the FFT, plotted through time. Since this difference is only in the way the data is plotted, the `specgram` function of `matplotlib` is used instead of the regular `plot` function. A spectrogram is a graphic representation of the frequency spectrum of a signal throughout time, it usually uses color and brightness to represent the spectrum of frequencies.

```
matplotlib.pyplot.specgram(x, NFFT=None, Fs=None, cmap=None)
```

The `specgram` function above has four main arguments: the array containing the result of the FFT of the signal,  $x$ , the number of samples of that dataset,  $NFFT$ , which is the sampling frequency times the duration of the sampling run, the sampling frequency,  $Fs$ , and the colormap,  $cmap$ , which is usually set to `Spectral` and sets the color gamut used to represent the frequencies.

Apart from plotting these graphs, the condition indicators mentioned in table 2.1 are calculated. To do this, a function was coded in python. That function can be consulted in appendix C. The code to calculate each indicator was based on the formulas in the table below:

Table 4.1: Condition indicators formulas

Condition Indicator	Formula
Average	$F_1 = \frac{1}{N} \sum_{i=1}^N x_i$
Skewness	$F_2 = \frac{1}{N} \sum_{i=1}^N (x_i - F_1)^2$
Slope sign change	$F_3 = \sum_{i=2}^N f[(x_i - x_{i-1}) + (x_i - x_{i+1})]$
Zero crossing	$F_4 = \sum_{i=1}^N \text{step}[\text{sign}(-x_i * x_{i+1})]$
Histogram upper limit	$F_5 = \max(x) + \frac{1}{2} \frac{\max(x) - \min(x)}{N-1}$
Margin index	$F_6 = \left( \frac{\max(x_i)}{\frac{1}{N} \sum_{i=1}^N \sqrt{x_i}} \right)^2$
Kurtosis	$F_7 = \frac{N \sum_{i=1}^N (x_i - F_1)^4}{\left[ \sum_{i=1}^N (x_i - F_1)^2 \right]^2}$
Latitude factor	$F_8 = \frac{\max( x_i )}{\frac{1}{N} \left( \sum_{i=1}^N \sqrt{ x_i } \right)^2}$
Slack factor	$F_9 = \frac{\max(x_i)}{\frac{1}{N} \left( \sum_{i=1}^N x_i^2 \right)}$
Mean deviation ratio	$F_{10} = \frac{F_1}{F_2}$

where:

$$f = \begin{cases} 1 & \text{if } x \geq \text{threshold} \\ 0 & \text{if } x < \text{threshold} \end{cases} \quad \text{Threshold} = 0.2 \quad (4.2)$$

$$\text{step} = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases} \quad (4.3)$$

$$\text{sign} = \begin{cases} 1 & \text{if } x > 0 \\ 0.5 & \text{if } x = 0 \\ 0 & \text{if } x < 0 \end{cases} \quad (4.4)$$

## Data Processing Overview

After processing, the data from the accelerometer is available as: the raw list data from the sensor, a .csv file with that data, time domain, frequency domain and time-frequency domain graphs and the condition indicators.

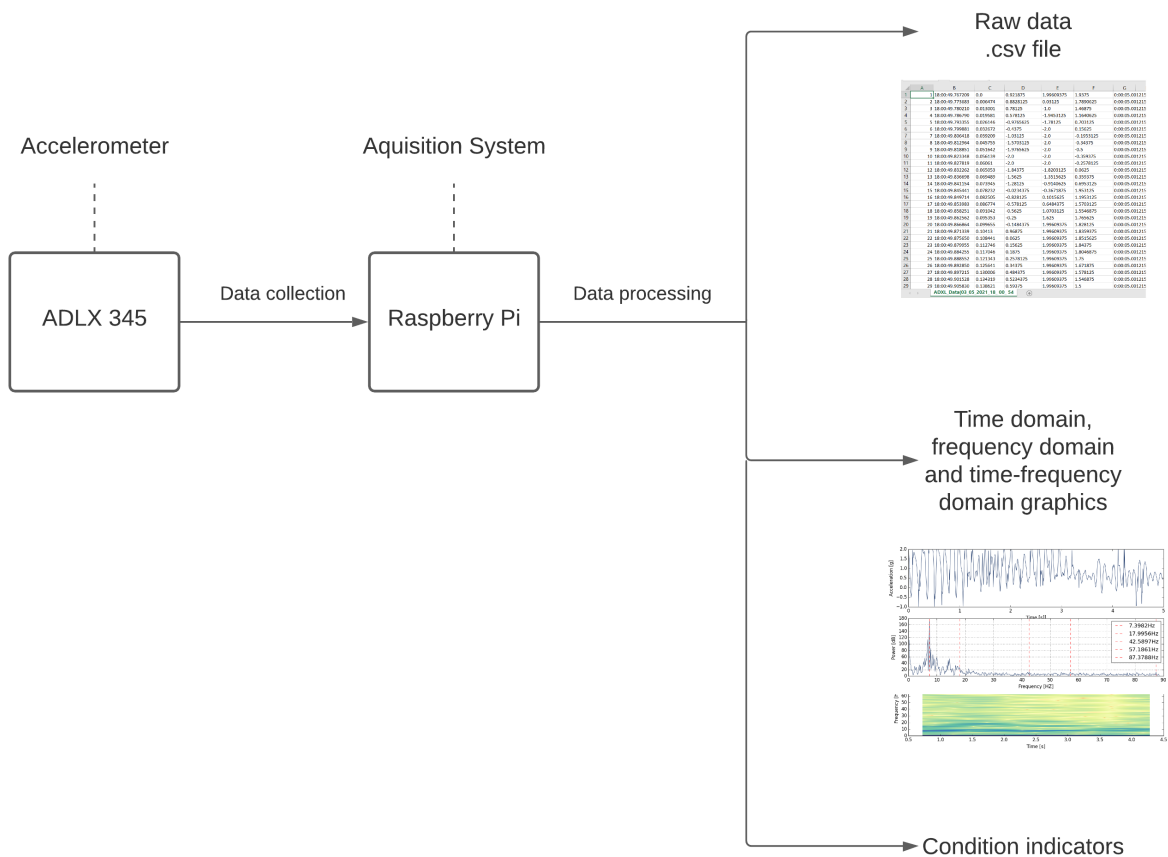


Figure 4.4: Data flow chart for the accelerometer

As for the torque data from the machine's servomotor, since this data is of a different nature, the data is not suitable for any treatment other than plotting it against time, position and velocity. In terms of storage for later constructing the PdM model, it only makes sense to keep the raw data, as these graphs can be easily obtained later and are only relevant in comparison similar to graphs from different points in the machine's life and not on their own. Because this raw data through the machine's life adds up to very large amounts of data, and given the limited nature of storage, as the more storage needed the more expensive the storage solution becomes, compression algorithms such as the ones presented in section 2.4.1 may be needed.



## 4.4 Data Storage

As above mentioned, there are two types of storage solutions: local and on the cloud. Because of the limitations of local storage, a cloud solution was chosen.

Within cloud solutions there is also an option between it being handled by AMOB itself or a contracted third party. A first party system would require AMOB to have and maintain its own servers, with all the upfront and maintenance costs it involves and the development of all customer faced solutions and data analysis tools to be done from scratch. Given the small dimension of AMOB and the lack of economies of scale, this solution is not feasible. By contracting these services to a third party not only are the IT infrastructure problems solved as most of these storage services come bundled with advanced data processing tools and development solutions which result in easier and faster development of their IT products.

INEGI searched the market for solutions and in conjunction with AMOB settled for the Microsoft Azure, which is Microsoft's cloud computing and data storage service for businesses. Azure is a very encompassing solution that offers software as a service (SaaS), meaning software applications that are contracted as a subscription service, platform as a service (PaaS), which allows code made by the client, in this case INEGI and AMOB, to be ran at Microsoft computers, and infrastructure as a service (IaaS), which provides high level tools that facilitate the development of software, such as APIs.

There are three of these bundled solutions that made Azure a particularly interesting solution for this application: Azure IoT Hub, Azure Stream Analytics, Power BI.

The IoT Hub allows for communication of the Raspberry Pi with the rest of Azure cloud services and for over-the-air upgrades to the code running on the Raspberry Pi.

Stream Analytics allows access to real time analytics calculated on the cloud, by using ML and other technologies. These analytics help the end clients make better decisions and optimizing their production processes. It allows the selection of which metrics to import by creating simple SQL queries.

Power BI allows for easy visualization of all of this data. It acts as a hub for monitoring the machine and process with simple dashboards and interactive reports. This is the basis of the interface the end client will interact with.

In the figure 4.5 there is a scheme of flow of data between an IoT edge device, such as a Raspberry Pi, the other IoT devices and sensors and the cloud.

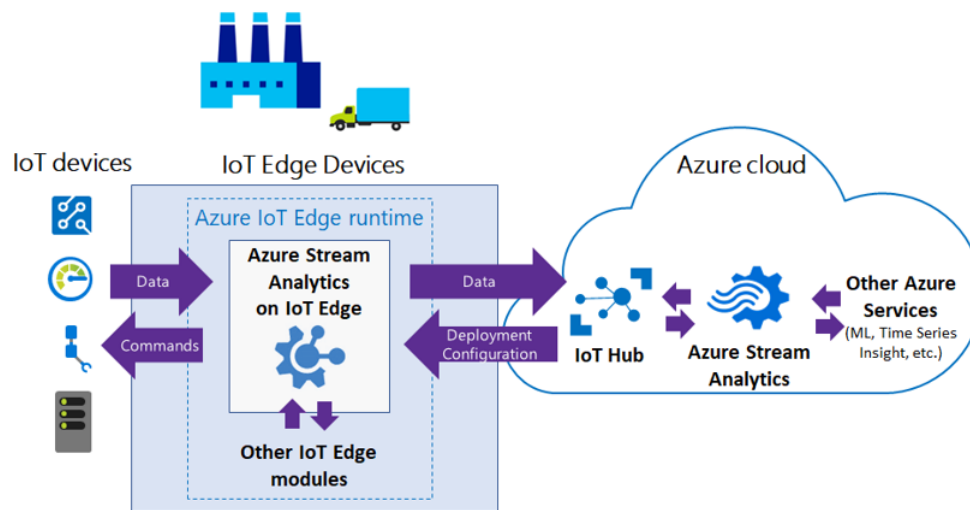


Figure 4.5: Flow of data diagram between IoT Edge device and Microsoft Azure cloud services (an emma et al., 2020)

The Azure Cloud services are priced, among other things, by the amount of storage available. Microsoft subdivides the storage into three access levels: hot, cold and archive. Hot storage is meant for data that is accessed frequently and that is meant to be kept for a very short period of time. This is mainly used to keep data relevant to real time operations and can act as a place to hold data while it is waiting to be processed and transferred to cold storage. Cold storage is meant for data that are more scarcely accessed and that is stored for longer periods of time, at least 30 days. This is useful for holding large data sets for future analysis, such as the data for PdM modeling. Last but not least, archive is very similar to cold storage in terms of uses but for data that is going to be kept for at least 180 days and does not need to be accessed at command. The reason is that there is a one day latency to access the data, as while it is in archive it is not online, having to be fetched.

The less sporadic the accessed to stored data the cheaper it is. Therefore, archive data is the cheapest and hot data the most expensive. The contracted storage capacity has not been decided yet.

For this application, cold storage is the most relevant. As a .csv file with data from one axis weighs 3.3 Mb per minute of trial at a sampling rate of 300 Hz for servomotor data (sampling rate mostly used by AMOB when collecting data traces from their machines), and considering one minute of data collected per day daily for 2 years, 2.4 Gb of storage would be needed for the servo data of just one axis. As for the vibration data, at a sampling rate of 3200 Hz, a 15 second trial occupies 1.7 Mb. Considering again 15 seconds of this data stored a day for 2 years approximately 1.2 Gb would be needed for one axis. This means that around 3.6 Gb of cold storage are needed for each axis that is meant to be modeled. As for record keeping once the system is deployed, this would mean a 1.8 Gb per axis per year would be needed, as archive storage, if no compression is done to

the data.

## 4.5 Communication and data standards

AMOB's machines already use OPC UA, an open source GPL 2.0 license machine to machine industrial communications protocol, to communicate between the computer running the HMI and the servomotors.

Data from the machine would be stored in an SQL database. That database could be then directly accessed by the Raspberry Pi. However, this direct access presents security vulnerabilities to the machine. A solution is a python script running on the machine to access the machine's database and communicate securely with the Raspberry Pi, acting as a buffer. This would allow the communications to be encrypted by the script, and therefore secure. A graphic representation of this process can be seen in figure 4.6.

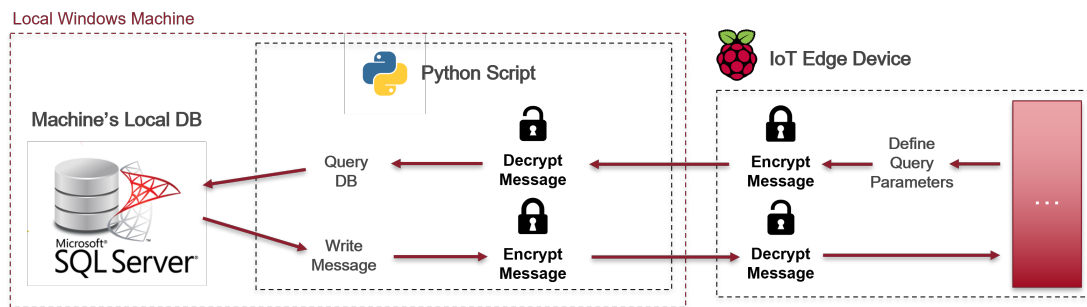


Figure 4.6: Machine-Raspberry Pi communication diagram

This communication between the machine and the Raspberry Pi would also be done using OPC UA, by using the OPC UA library on a python script or via file sharing using the File Transfer Protocol (FTC). The Raspberry Pi would need to be connected to the internet in order to communicate with the cloud, which can be done either wirelessly or using Ethernet. These communications are handled by the Azure IoT Edge.

## 4.6 System Overview

During the data gathering phase, estimated around two years, data is gathered daily using the system presented in this chapter and synthesized in the figure 4.7.

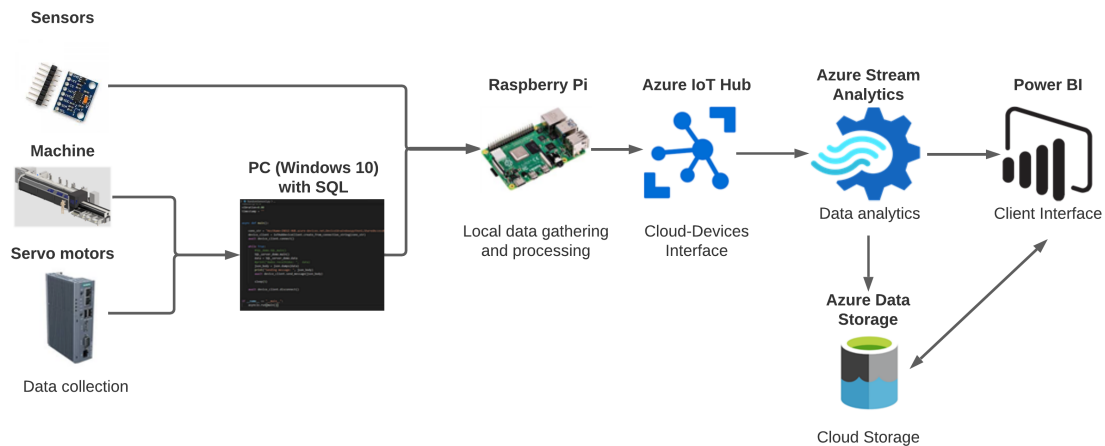


Figure 4.7: End-to-end data flow diagram of system

One set of data will be gathered every day, i.e. one minute of data traces from the servomotor and 15 seconds of vibration data, which is enough to capture the vibrations for the whole movement for speeds as low as 10% of the nominal motor speed. The vibration data must be gathered while the axis is in movement, which can easily be achieved through code. The positions during that time must also be kept, although they may be done at a much lower frequency than in the data traces. This allows for when preparing the data to train the ML model only to include the relevant parts (i.e. when the ball screw is being actuated). These are then uploaded to the cloud. This registers allow for the parameterization of the aging and degradation of the ball screw through-out time by the ML algorithms.

Once the historical data is gathered and used to train an ML algorithm, such as the ones presented in section 2.5.1, the resulting model is implemented as a python script on the Raspberry Pi. That script will have live data from the axis, both in terms of torque and vibration, as inputs and will output a estimated remaining useful life and a general condition state. These variables are then stored as hot data in the cloud for a short period of time. These stats can then be checked using a Power BI based application by the final customer. Some of this data can be archived so that the model can be assessed and, if needed, updated over time.

# Chapter 5

## Preliminary testing

The system presented in the previous chapter had as a basis the assumptions that torque and vibration could be used as indicators of degradation and failure. In this chapter, those assumptions, as well as a prototype of the acquisition system previously presented will be tested. The results of these tests will be presented and discussed.

The chapter will be divided into 2 main parts: vibration analysis and torque analysis.

### 5.1 Vibration

#### 5.1.1 Objectives

The objectives with this section of the thesis are:

- To validate sensor choice, placement, and overall proof of concept;
- To collect, filter and analyze time-domain and frequency domain data;
- To identify relevant frequencies of the system for future monitoring;
- Establish a baseline of vibrations for a well assembled machine;
- To test the hypothesis proposed of the influence of misalignment on vibrations.

### 5.1.2 Theory

The hypothesis tested in this section is that when assembling the machine, the ball screw is not correctly aligned, which causes notable changes in the vibration of the screw throughout its course.

This misalignment would cause radial strain between the screw and the nut, that will be referred to as force  $P$ .

Because the screw is fixed on one end and free on the other, and is elastic, which is translated as the Young modulus of the material of the screw, there can be deformation of the screw, which is represented by  $\theta$  as shown in the picture below.

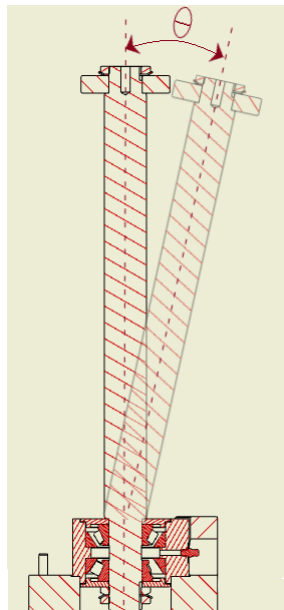


Figure 5.1: Representation of angular deformation

This bending is inversely proportional to the stiffness ( $k$ ) and proportional to the distance to the fixed end ( $l$ ) to the power of three, as shown by the equations of stiffness for a fixed-free beam (equations 5.1 and 5.2) presented below.

$$\theta = \frac{P \times l^3}{3EI} \quad (5.1)$$

$$k = \frac{P}{\theta} = \frac{3EI}{l^3} \quad (5.2)$$

$$I = \frac{\pi D^4}{64} \quad (5.3)$$

Therefore, the closer the ball nut is to the fixed end, the smaller  $l$  is, even for the same  $P$  force caused by the misalignment, the smaller is the ability of the beam to accommodate this force by deforming ( $\sigma$ ). This is aggravated by the fact that for a beam with constant section, as the screw is being considered, the moment of inertia ( $I$ ) for a circular section beam is proportional to the diameter to the power of 4, as shown in equation 5.3. Considering that the diameter of the screw of the ball screw in this case is exceptionally large for the application (40 mm), the stiffness ( $k$ ) of the screw is very high, aggravating the problem of not being able to deform closer to the fixed edge absorbing any forces due to misalignment.

Because the struggle between the force pushing the ball nut down due to the screw rotating and radial forces caused by the misalignment, that are less and less be absorbed by the screw, pressure and friction between the screw, balls and ball nut increases. This increasing struggle will, according to the hypothesis in this section, provoke increased vibrations as the ball nut approaches the lower part of the screw, and the added pressure and friction will accelerate the degradation and failure of these parts. This would cause faster wear of the screw and, as seen before, is a cause of concern in terms of maintenance of these machines.

### 5.1.3 Approach

In a machine in every way equal to the one that presented in section 3.4, that was present at AMOB, two ADXL345 sensors were installed and connected to a Raspberry Pi running the adequate code. Using the same procedure as in chapter 3, the misalignment was measured at 0.10 mm per 200 mm of course, or 0.15 mm per 300 mm. While this is less than half the misalignment on the ball screw previously presented it is still well above the T7 tolerance, at almost double its value.

Preliminary trials using both SPI and I2C at different sampling rates and scales were ran to test both protocols and see which one was more appropriate.

Using SPI at 3200 Hz sampling rate 23 trials were ran with different velocities and positions to test the influence of these parameters in the results.

In terms of positions the trials ran between the two most extreme positions, 35 mm and 274 mm, and also stopping in the middle, at the 155 mm position. This allows the comparison between the vibrations on the upper part of the ball screw and the lower part.

In terms of velocity, the trials were done at: 150%, 105%, 75%, 30% and 15% of the nominal velocity of the motor (i.e. 2000 rpm, equivalent of 64 mm/s ball screw nut velocity). On automatic mode, the way these machines operate most of the time, the velocity defaults to 150% of the servomotor velocity, or 96 mm/s linear velocity of the nut. The objective of these trials is to see the influence of the velocity on the vibrations and whether certain velocities make more evident

features of interest of the signal.

On the table 5.1 shown bellow, the parameters for each trial are listed.

Table 5.1: Positions and velocities of vibration trials

<b>Trial</b>	1	2	3	4	5	6	7	8	9	10
<b>Initial Position [mm]</b>	35	35	274	35	274	35	274	35	274	35
<b>Final Position [mm]</b>	35	274	35	274	35	274	35	274	35	274
<b>Velocity [% of nominal velocity]</b>	0	150	150	105	105	75	75	30	30	15
<b>Nut Velocity [mm/s]</b>	0	96	96	67,2	67,2	48	48	19,2	19,2	9,6
<b>Trial</b>	11	12	13	14	15	16	17	18	19	20
<b>Initial Position [mm]</b>	274	35	155	274	155	35	155	274	155	35
<b>Final Position [mm]</b>	35	155	274	155	35	155	274	155	35	155
<b>Velocity [% of nominal velocity]</b>	15	150	150	150	150	75	75	75	75	15
<b>Nut Velocity [mm/s]</b>	9,6	96	96	96	96	48	48	48	48	9,6
<b>Trial</b>	21	22	23							
<b>Initial Position [mm]</b>	155	274	155							
<b>Final Position [mm]</b>	274	155	35							
<b>Velocity [% of nominal velocity]</b>	15	15	15							
<b>Nut Velocity [mm/s]</b>	9,6	9,6	9,6							



### 5.1.4 Results and discussion

Concerning the data from I2C connected sensor, as suspected, the data for sampling rates over 400  $Hz$  appear as the one in the figure bellow.

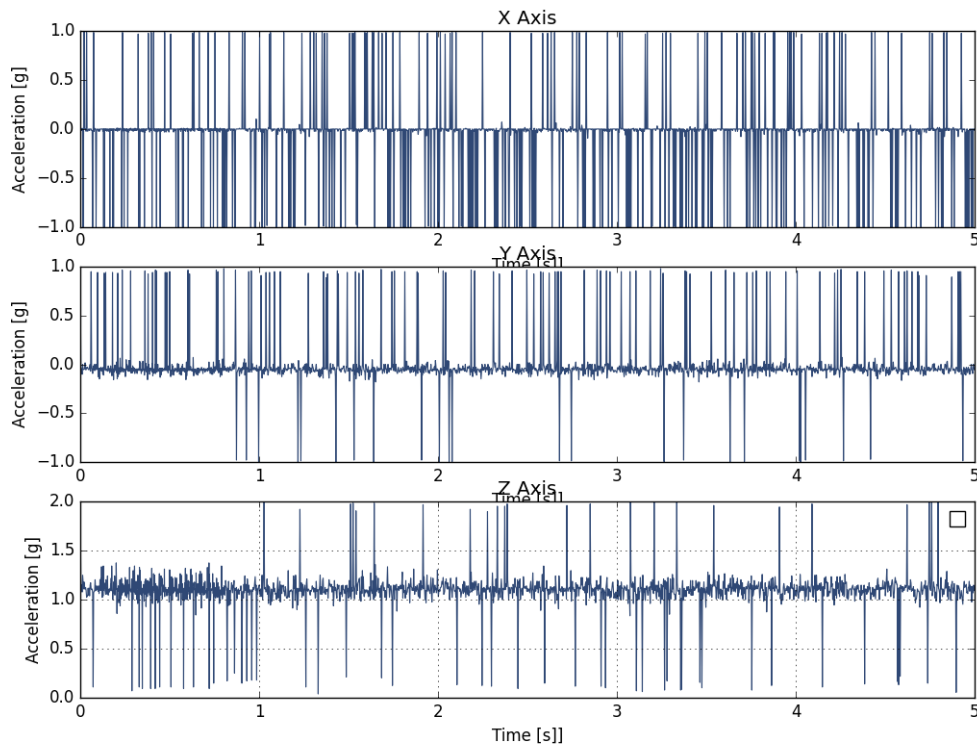


Figure 5.2: Examples of time-domain data from the ADXL345 connected via I2C for a sampling rate over 400  $Hz$

As can be seen in figure 5.2, the data for the x and y axis alternates between the extremes of the plot, in this case 1  $g$  and -1  $g$ , as the range was set for 2  $g$  in this particular trial. Several trials changing the range and sampling rate resulted in similar results for rates over 400  $Hz$ . This occurs because the Raspberry probes the sensor for data faster than the speed of the connection which generates the overpowering noise that can be seen in the signal.

When analyzing the frequency domain graphs for sampling frequencies over 400  $Hz$  it also becomes apparent that, as expected, regardless of the sampling rate the graph stops at 200  $Hz$ , as can be seen in the picture below.

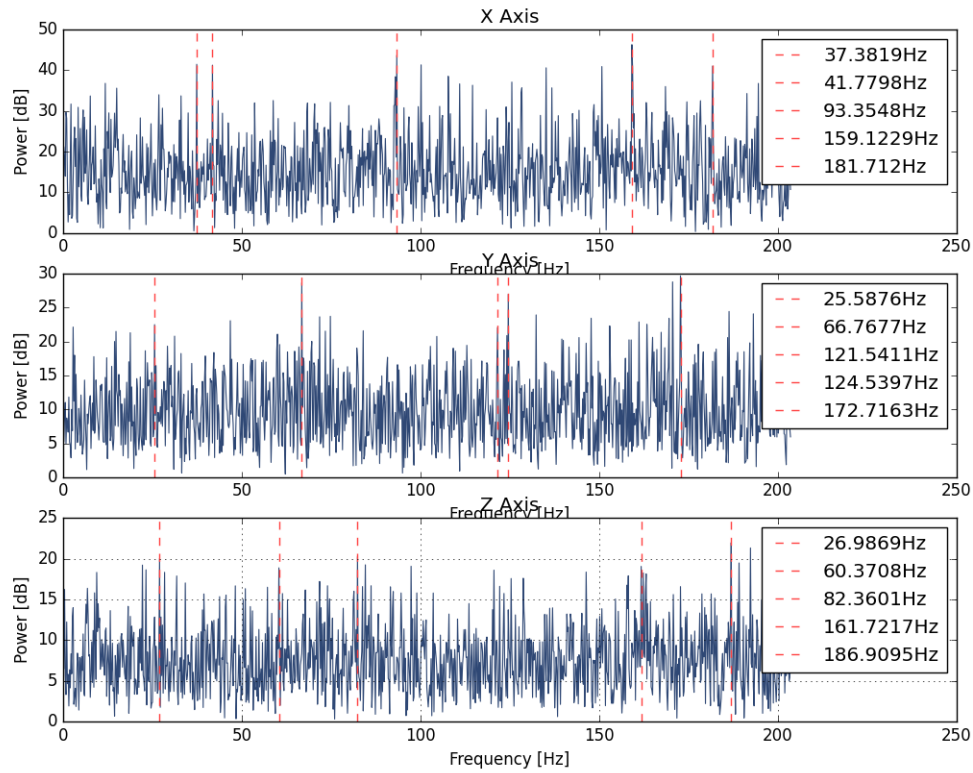


Figure 5.3: Examples of frequency-domain data from the ADXL345 connected via I2C for a sampling rate over 400 Hz

Because of the amount of noise that rendered the signal useless and the cut off frequency for the frequency domain graphs SPI was used for the trials testing the hypothesis and gathering the data to be analyzed,

In the tables below are the results of the 10 most relevant condition indicators calculated based on the vibration data of the trials presented on tables 5.2, 5.3 and 5.4 for each of the 3 axis of the sensor.

Table 5.2: Condition indicators for the vibration trials in respect to the X axis of the sensor

Index	Average	Skewness	Slope Sign Change	Zero Crossing	Histogram Upper Limit	Margin Index	Kurtosis	Latitude Factor	Clearance Factor	Avg. Standard Deviation Ratio
1	1.83E-02	0.332	712	9407	0.250	1.73	0.373	6.94	60.00	0.296
2	1.90E-02	0.160	1856	9367	0.555	6.63	1.044	11.95	81.93	0.237
3	1.86E-02	0.078	2063	9400	0.359	2.65	0.889	7.38	49.36	0.223
4	1.90E-02	0.184	1529	11238	0.328	2.52	0.434	7.69	60.05	0.266
5	1.82E-02	0.246	1655	11415	0.344	2.79	0.556	8.12	62.26	0.252
6	1.87E-02	0.206	1525	15087	0.352	3.12	0.422	8.87	73.70	0.282
7	1.77E-02	0.150	2265	15232	0.336	2.64	0.521	7.85	61.57	0.246
8	1.87E-02	0.284	2008	26525	0.312	2.72	0.455	8.69	75.09	0.302
9	1.88E-02	0.303	2061	26394	0.352	3.41	0.480	9.69	83.07	0.301
10	1.76E-02	0.323	3786	52593	0.312	2.78	0.529	8.88	76.91	0.287
11	1.88E-02	0.248	4598	53291	0.305	2.49	0.391	8.19	70.77	0.299
12	1.79E-02	0.093	1130	5747	0.352	2.69	0.976	7.64	53.18	0.225
13	1.77E-02	0.221	995	5803	0.437	4.29	0.862	9.82	70.79	0.231
14	1.82E-02	0.087	1052	5683	0.320	2.23	0.680	6.95	49.08	0.231
15	1.88E-02	0.056	1128	5819	0.414	3.73	0.983	9.00	61.40	0.235
16	1.81E-02	0.311	907	9502	0.367	3.57	0.532	9.71	80.86	0.279
17	1.86E-02	0.222	997	9532	0.305	2.38	0.513	7.81	64.49	0.281
18	1.86E-02	0.095	1318	9439	0.297	2.09	0.394	7.05	56.37	0.266
19	1.86E-02	0.227	1284	9616	0.305	2.28	0.498	7.48	59.49	0.269
20	1.90E-02	0.259	2473	28437	0.336	3.07	0.456	9.12	79.18	0.304
21	1.90E-02	0.281	2355	28506	0.305	2.52	0.474	8.26	70.88	0.303
22	1.89E-02	0.289	2146	28449	0.297	2.42	0.470	8.17	69.56	0.303
23	1.86E-02	0.261	2507	28727	0.297	2.40	0.431	8.09	70.07	0.299

Table 5.3: Condition indicators for the vibration trials in respect to the Y axis of the sensor

Index	Average	Skewness	Slope Sign Change	Zero Crossing	Histogram Upper Limit	Margin Index	Kurtosis	Latitude Factor	Clearance Factor	Avg. Standard Deviation Ratio
1	-4.10E-03	0.252	660	9770	0.266	2.00	0.407	7.51	69.59	-0.066
2	-4.88E-03	0.172	1408	9689	0.367	3.20	0.524	8.71	69.68	-0.067
3	-5.01E-03	0.128	1501	9789	0.367	3.07	0.592	8.37	63.70	-0.066
4	-5.22E-03	0.193	1224	12027	0.281	2.07	0.464	7.36	62.52	-0.078
5	-5.36E-03	0.217	1258	11476	0.289	2.06	0.461	7.12	59.72	-0.077
6	-5.07E-03	0.222	1416	15875	0.258	1.79	0.414	6.94	61.27	-0.078
7	-5.54E-03	0.249	1393	15501	0.297	2.34	0.445	7.90	69.40	-0.085
8	-4.62E-03	0.248	2035	28007	0.328	3.10	0.542	9.45	86.36	-0.075
9	-4.81E-03	0.265	1921	27497	0.258	1.89	0.483	7.31	66.52	-0.077
10	-5.83E-03	0.267	3641	54153	0.352	3.53	0.535	10.05	92.54	-0.095
11	-4.74E-03	0.222	4372	56244	0.281	2.24	0.476	7.96	73.08	-0.077
12	-4.55E-03	0.171	811	5916	0.352	2.97	0.686	8.45	67.49	-0.063
13	-5.51E-03	0.087	810	5922	0.312	2.42	0.635	7.73	62.53	-0.078
14	-4.79E-03	0.168	834	5868	0.305	2.20	0.733	7.23	55.64	-0.065
15	-3.88E-03	0.125	905	6073	0.320	2.44	0.762	7.62	59.28	-0.053
16	-5.63E-03	0.241	659	9691	0.250	1.74	0.322	6.97	64.47	-0.091
17	-3.93E-03	0.165	821	10022	0.305	2.58	0.479	8.47	76.30	-0.062
18	-4.45E-03	0.191	1072	10186	0.266	1.94	0.570	7.29	63.40	-0.069
19	-4.95E-03	0.209	897	9923	0.281	2.14	0.495	7.62	66.40	-0.076
20	-4.80E-03	0.190	2302	30313	0.266	2.03	0.533	7.64	69.25	-0.078
21	-4.45E-03	0.226	2445	30180	0.297	2.46	0.560	8.29	74.34	-0.071
22	-4.84E-03	0.244	2160	29604	0.297	2.49	0.446	8.38	76.15	-0.078
23	-4.25E-03	0.249	2478	30419	0.273	2.12	0.533	7.77	70.63	-0.069

Table 5.4: Condition indicators for the vibration trials in respect to the Z axis of the sensor

Index	Average	Skewness	Slope Sign Change	Zero Crossing	Histogram Upper Limit	Margin Index	Kurtosis	Latitude Factor	Clearance Factor	Avg. Standard Deviation Ratio
1	1.154	0.296	2447	6192	1.531	2.036	0.282	1.330	1.143	12.226
2	1.155	0.191	2472	6320	1.578	2.161	0.350	1.370	1.175	11.509
3	1.154	0.180	2562	6374	1.625	2.294	0.313	1.411	1.212	11.369
4	1.154	0.205	2983	7777	1.562	2.119	0.256	1.356	1.165	11.871
5	1.151	0.229	2992	7347	1.531	2.040	0.310	1.332	1.147	11.888
6	1.153	0.133	3893	10400	1.539	2.057	0.966	1.337	1.149	12.105
7	1.151	0.278	4167	9818	1.555	2.104	0.217	1.353	1.165	12.061
8	1.153	0.242	6506	17461	1.648	2.360	0.245	1.432	1.232	12.512
9	1.153	0.280	6381	17317	1.594	2.207	0.296	1.385	1.191	12.413
10	1.151	0.295	12985	33020	1.586	2.189	0.574	1.380	1.189	12.463
11	1.152	0.178	13955	36327	1.547	2.080	0.223	1.344	1.157	12.454
12	1.151	0.127	1619	3893	1.570	2.146	0.405	1.367	1.176	11.501
13	1.153	0.121	1552	3864	1.672	2.430	0.328	1.454	1.249	11.476
14	1.150	0.159	1426	3734	1.555	2.105	0.294	1.354	1.166	11.770
15	1.152	0.119	1653	3987	1.562	2.123	0.168	1.359	1.169	11.800
16	1.151	0.348	2362	5980	1.523	2.021	0.357	1.326	1.143	12.242
17	1.152	0.163	2535	6367	1.516	1.998	0.343	1.318	1.134	12.130
18	1.151	0.130	2656	6479	1.539	2.062	0.325	1.340	1.154	12.214
19	1.152	0.241	2342	6276	1.547	2.081	0.246	1.345	1.158	12.186
20	1.153	0.108	7616	19749	1.547	2.078	0.614	1.344	1.156	12.376
21	1.152	0.194	7232	19281	1.531	2.039	0.306	1.332	1.147	12.374
22	1.151	0.168	7111	18831	1.656	2.387	0.760	1.441	1.241	12.232
23	1.152	0.158	7562	19592	1.594	2.208	0.253	1.385	1.192	12.459

The first thing that stands out in this data is that across all axis the average value for the signal stays roughly the same. For the x and y axis this value is around 0 g, which is what is expected. For the z axis is around 1.15 g, which is slightly above the conventional value of 1 g, but is consistently close to the value of the control trial (index 1, no movement). The 0,15 g difference to what is expected may be due to the elevation of the factory or bad factory calibration of the sensor but does not bear any impact in the analysis done.

When analyzing the data considering attending to the velocity, it can be seen that across all axis the slope sign change, zero crossing and the average-standard deviation ratio lower in value with the increase in velocity. As for just the x and y axis, the skewness also tends to decrease with the increase in velocity. In the z axis no clear relationship is observed. Still regarding the x and y axis, the upper histogram limit, the margin factor and the kurtosis appear to increase with the velocity. There are some trials that do not follow this norm but this may be statistical outliers and not the trend. Ideally each trial would have been repeated at least three times in order to dilute the impact of these outliers, however, since the only time the machine was available was between it being finished and shipped to the client, there was not enough time.

When comparing the data from each section of the screw, there is no parameter, that across all three velocities that trials were conducted in sections, showed a consistent change. Despite the hypothesis, vibrations do not appear to be significant and consistently higher in the lower section of the ball screw across velocities. There does not seem to be any pattern across different condition indicators either. This seems to indicate that the results are due to normal variance.

### **5.1.5 Conclusions**

The first conclusion is that the system must not use I2C for connecting the ADXL345 sensor to the Raspberry Pi due to all its limitations.

As for the results obtained through SPI, all fall within the order of magnitude expected for this application, and are consistent within the normal variability, which can be interpreted as a preliminary validation of this protocol for this system. Therefore, when implementing the system this must be the communication protocol used to connect the sensor to the Raspberry Pi.

Overall, the condition indicators vary, withing the expectable limits, for trials with the same speed. However, this does not translate to a trend when correlated with the section of the ball screw was traveled. This points to none of the condition indicators being a good and reliable indicator of misalignment.

Due to the small number of trials, all the conclusions here presented should be confirmed with different machines and repetition of each trial in order to exclude outliers. It would also be impor-

tant to gather data from machines with different misalignments, in order to compare and see if any indicator of it emerges with the aggravation of the misalignment.

## 5.2 Torque

### 5.2.1 Theory and objectives

As mentioned before, changes in torque of the servomotors powering mechanical systems are considered to be indicative of failure. The hypothesis is that because wear of the parts manifests itself as a higher roughness of the surfaces with relative movement, and higher roughness leads to more friction between them, a higher torque will be needed to surpass the added friction for the same movement.

Since the information regarding torque and other parameters is available through datatraces, the main objective of this section of the thesis is to analyze this data for changes related to malfunction. There is also the aim to:

- Establish baselines of normal torque and the torque in cases of failure
- Establish the impact of misalignment of the ball screw in the torque exerted by the servomotors

These two last objectives are relevant not only in establishing PdM solutions in the long term but in the development of solutions that allow for correction of problems that impact the longevity of the machine before leaving the factory, augmenting therefore quality control and diminishing the need for assistance. More complete condition based maintenance solutions can also be implemented in the short term based on this data, bridging the transition from the current solutions to PdM solutions when these are developed.

### 5.2.2 Approach

To do this, the torque for a control and for the malfunctioning machine's were plotted against position and velocity. If any notable differences in these plots are detected, metrics that quantify these differences.

Firstly, from the e-MOB42 9611, datatraces were collected 3 days after the third replacement of the ball screw (i.e. 10<sup>th</sup> of March 2021) and before it's failure, in the 22<sup>nd</sup> of April 2021. About a month later this ball screw was replaced, at the 25<sup>th</sup> of May 2021.

Since the machine was at use by the client during this time, the data traces were from the work cycles of the parts it was manufacturing at the moment. These data traces were collected on days the same part was being produced, to facilitate comparison.

Data traces were also collected from an identical e-MOB 42 machine at AMOB, before going to its client. The machine is the same as the one from the vibration trials, with a 0.10 mm misalignment over a 200 mm path, less than half the misalignment of e-MOB 42 9611.

The data traces from the e-MOB 9611 allow for conclusions on the impact of wear over time. The reasoning being that since the misalignment provoked accelerated wear that the same changes in torque would be observed in normal wear, just more gradually and over a more prolonged window of time. This would mean that the time needed to model the wear of the machine could be shortened from around two years to around 2 months.

The data from the machine at AMOB's factory allows the assessment of how much of these differences is truly due to wear and how much is due to the misalignment. This also allows the quantification of the impact of misalignments on torque.

### **5.2.3 Results and discussion**

The results presented consist of two graphs, torque-position graphs and torque speed graphs. Because of the way the variable position is defined, as the distance from the top of the machine's body to the top of the machine's head, these graphs are not of immediate comprehension. In the picture below is an example of both these graphs, with key areas marked with different colors.



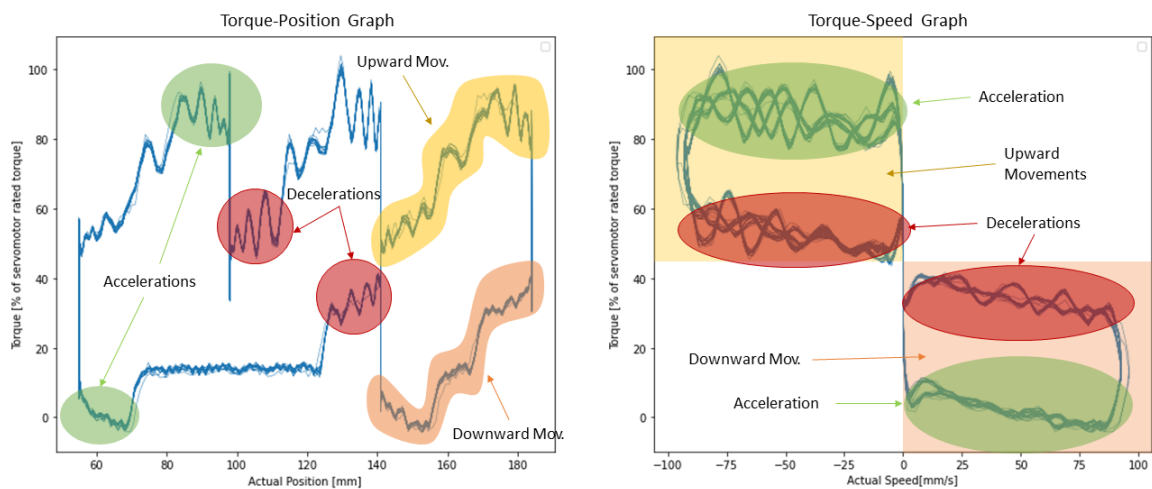


Figure 5.4: Examples of torque-position (on the left) and torque speed graphs (on the right) with key areas marked

The first thing that should be noticed is that in the torque position graphs the plot must be interpreted in an anti-clockwise direction. Each movement starts with an acceleration, marked green in figure 5.4, followed by an intermediary stage where it reached the speed set for that movement, and a final deceleration stage, marked red. In order to maintain its position, due to the lack of a mechanical brake, a certain torque is needed, around that 40% torque mark. Upward movements, such as the one marked in yellow, will fall above this line, while downward movements will fall below it, as marked in orange.

As for the torque speed graph, the movements fall within 2 quadrants of this graph, the 2<sup>nd</sup> and 4<sup>th</sup> quadrants, considering that line around 40% torque de quadrant division instead of the x axis. The 2<sup>nd</sup> quadrant, marked yellow, is where the upward movements of the head are plotted, while the 4<sup>th</sup>, marked in orange, is where the downward movements are plotted. Decelerations are located closer to the quadrant division (the 40% torque line), while accelerations are closer to the extremes of the graph. In this graph too, decelerations are marked red and accelerations marked green. The best way to interpret these torque-speed graphs is to read each quadrant independently in an anti-clockwise direction.

In figure 5.5 these graphs can be seen for the data from e-MOB 42 9611 after the replacement of the ball screw and before failure.

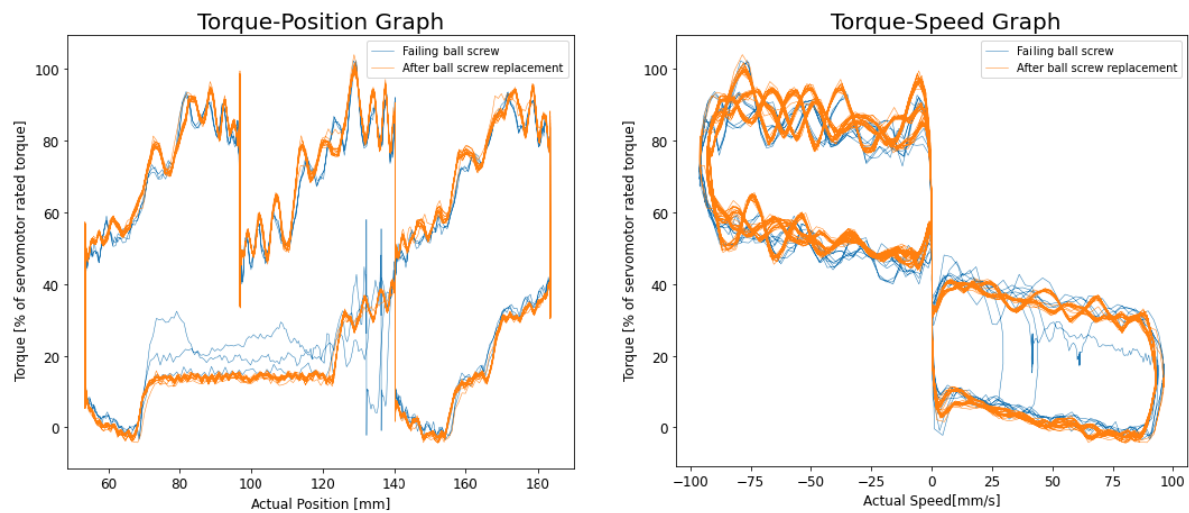


Figure 5.5: Torque-position (on the left) and torque speed graphs (on the right) for e-MOB 42 9611

Regarding the torque-position graph, no notable differences can be observed, with the notable exception of the downward movement from 55 mm to 141 mm. This difference consists of increased torque in the constant speed section and erratic behavior in the deceleration section. The increase in the constant speed area is not consistent across every time this movement was executed in this data trace. Therefore, each cycle of movements must be analyzed individually to try and understand what may be causing this behavior.

The torque velocity graph shows similar behavior, where two standout lines representing two downward movements clearly stand out on the failing ball screw. In these two movements, instead of the around 90 mm/s speed reach in all other movements, only around 25 mm/s and 40 mm/s speeds are reached. The erratic behavior in the deceleration phase of this movement is also notable here, with one of the movements showing noticeable dips around 40 mm/s and 60 mm/s.

In figure 5.6 these graphs can be seen for the data from e-MOB 42 9611 after the replacement of the screw is compared with data from the e-MOB 42 available at AMOB facilities with a less severe misalignment.

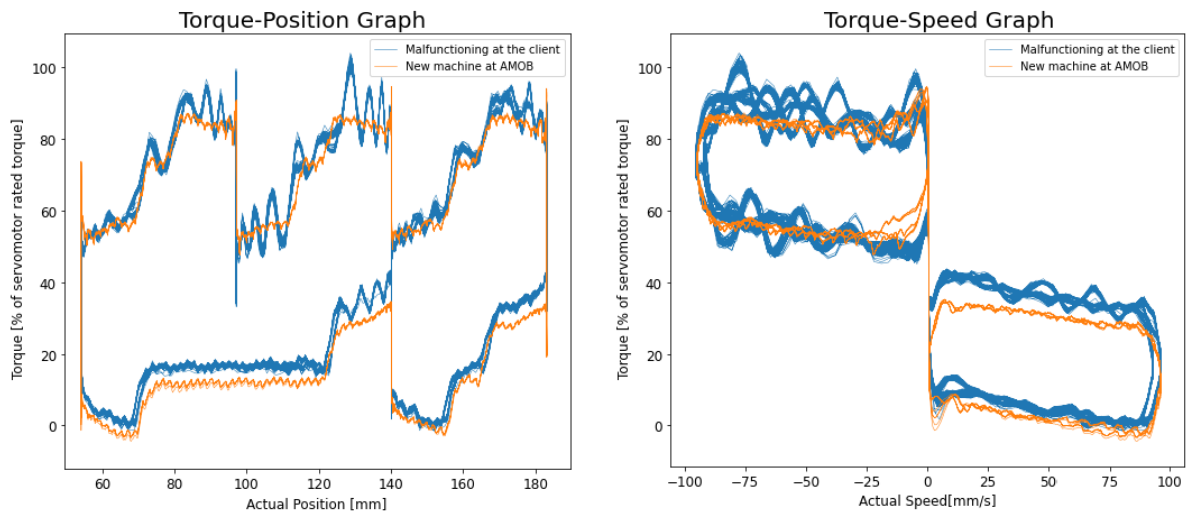


Figure 5.6: Torque-position (on the left) and torque speed graphs (on the right) for e-MOB 42 9611 (blue) and the e-MOB 42 available at AMOB (orange)

In these graphs differences between the torque in the two machines are very apparent, with higher and more tempestuous torques in the machine at the client than the one at AMOB.

In fact, these differences are most apparent in the decelerations for downward movements (green), the accelerations in upward movements (yellow), the constant speed section of the downward movement between 55 mm and 141 mm (orange) and the deceleration in the upward movement between 141mm and 98mm (blue). These areas are marked in the figure 5.7 bellow with the mentioned colors and the average torque and the torque range for each of these movements are presented in the table 5.5.

Table 5.5: Metrics of torque for parts movements where the malfunctioning machine diverges from the control machine at AMOB

Initial Position [mm]	Final Position [mm]	Average Malfunctioning Machine Torque [% of servomotor rated torque]	Average New Machine at AMOB Torque [% of servomotor rated torque]	Malfunctioning Machine Torque Range [% of servomotor rated torque]	Machine at AMOB Torque Range [% of servomotor rated torque]
96	82	86.49	83.92	17.6	7.7
140	124	86.68	83.42	29.1	8.1
183	165	86.85	83.13	19.3	11.3
116	102	59.83	55.68	36.1	19.3
128	140	33.28	28.31	14.0	7.4
173	183	33.11	28.12	8.2	6.5
80	120	14.28	9.42	6.1	4.6

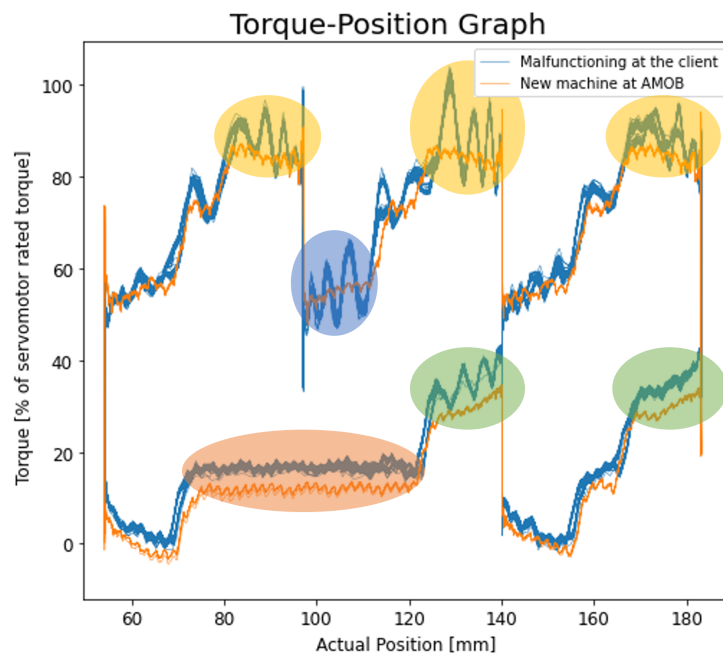


Figure 5.7: Torque-position for e-MOB 42 9611 (blue line) and the e-MOB 42 available at AMOB (orange line) with areas of interest colored

The data presented in the table above shows that for regions where the plots of the two machines are significantly different the average torque is between 2.5 to 5 percentage points of the rated torque of the servomotor higher for the malfunctioning machine. This means the machine requires more torque to lift the same load. Based on this data, this difference decreases with the increase in servomotor torque, which consequently means the upward movements show smaller differences of average torque than downwards movements. As a matter of fact, the average difference for

upwards movements is of 3.425 percentage points of servomotor rated torque while for downward movements is 4.94.

Concerning the range of torque, the values vary widely. Because these values are numerically small, the ratio of the range for the malfunctioning machine over the range of the machine at AMOB is a better metric to assess this data. The range for the malfunctioning machine is between 1.26 and 3.59 times higher than the machine at AMOB. Here, even though the average range for upwards movements is still higher than for downward movements, there is overlap in the two, which does not point to any significant correlation between torque range and direction of movement. There is also no indication of a clear correlation between range and the the position of the screw (i.e. whether range is higher or lower depending on if the nut is towards the top or bottom of the screw).

Even though when analyzing for in which part of the movement (acceleration, constant speed or deceleration) this data seems to indicate significantly higher ranges, averaging 2.53 times higher for the malfunctioning machine, for acceleration, since no acceleration area of downward movements is represented in this data, as it is not significantly different than the control (i. e. machine at AMOB), this apparent correlation is due to the selection of the sections of the graph that were analyzed. This does not have the same impact on the remaining correlations as there was at least one sample from each category of the parameter (e.g. there was data from almost all positions represented in the data in table 5.5).

#### 5.2.4 Conclusions

In the two experiments conducted in this section, there are two main takeaways: there is no significant changes in torque for the machine with the severe misalignment at the client before and after the ball screw being replaced and that that machine has significantly different torque patterns than a more well aligned machine.

Therefore, it can be concluded that misalignment trumps any effects of wear and aging of the parts when it comes to torque. This means that alignment is an important factor to consider when making the PdM system. The proposed solution is to take advantage of the architecture devised for the PdM system in this thesis and, based on the data from the data presented in this section and from additional data, create a system that would serve to validate the alignment based on torque. For example, using the data from the control machine as a baseline and the data from the machine with the misalignment to establish thresholds. Before leaving the AMOB facilities for the client the machine would run a cycle similar to the ones from these experiments, if its torque values are over the threshold the machine would need to be realigned. This is faster and easier than checking the alignment as presented in section 3.4.6.4 for every machine. This would allow for the mitigation of the influence of alignment on the models for PdM, increasing their reliability and

decreasing their complexity. An additional advantage is that machines correctly aligned require less maintenance procedures.

An interesting experiment to validate the conclusions drawn in this section would be to continue to record data traces of the machine that was used as control in the second experiment over time once at the customer. The changes between the data from different points in time would allow the modeling of the wear of the machine in terms of torque. If not, then this level of misalignment is still excessive in terms of implementing PdM based on torque.

# Chapter 6

## Conclusion

### 6.1 Conclusion

Developing predictive maintenance systems is a onerous and time-consuming endeavor. However, productivity gains and cost savings are a big payoff.

In the context of AMOB, a traditional family own company, the implementation of these kinds of solutions present a big departure from the currently implemented strategies.

The fact that this thesis was developed in an industrial setting also presented its whole set of challenges. This is due to the objectives of a company, maximizing production and reducing costs, such as stock, are not always aligned with academic and research goals, dependent on constant settings and comparable and controllable conditions. An example of this was the window of time for implementation and testing of the solutions on an actual machine, which is defined by the time between finishing its assembly and shipping it to the client. Because the machines available for testing were meant to go to clients, it also meant any destructive or damaging tests were off the table, since the machine needed to be kept in mint condition. Another hurdle was the lack of track records for the machines, which meant that, instead of devising maintenance algorithms and testing them, this thesis had to focus more on creating that underlining infrastructure for that data to be collected.

Despite these difficulties, the architecture for a system of predictive maintenance was outlined, based on the most common problems in the e-MOB machines. This meant finding appropriate metrics to detect this problems and ways to measure them (sensors). A solution to process this data, the Raspberry Pi, and to store it, the Azure cloud, were also established, having in mind AMOB's needs and the solutions they were already implementing. How the communication between all these parts of the system, as well as standards for data keeping such as file types were chosen.

The programming responsible for the gathering and treatment of the data from the sensor was also done and successfully tested. Based on these tests the SPI protocol was chosen for communication between the Raspberry Pi and the machine. The tests done also failed to confirm the proposed hypotheses that ball screw misalignment manifested as vibration. More testing, under a more diverse set of conditions (different degrees of misalignment) is needed to evaluate if this system can also be used to detect misalignments or not, and if so, what is the threshold for these misalignments to be detected by it.

As for the torque data from the tests, the main takeaway is that the effects of misalignment on it overshadow those of wear. This means that any conclusions on the effects of wear on torque cannot be established since both machines used in testing suffered, to different degrees, of misalignment.

Overall, there was success in the main objective of this thesis, which was to study how predictive maintenance could be applied to draw bending machines and devise a solution to implement it in AMOB's machines. However, this was only done to the extent of planing. This means the project is left in the developing and testing phase, with some of that developing and testing being also done in this thesis, namely of the sensors.

## **6.2 Future Work**

The next steps in this project are to collect data on several machines over the lifetime of the parts and testing different ML algorithms, such as the ones presented in this thesis, using this data to establish which to implement in this system. The extensive work of professor René Vinivio Sánchez in this field can be used as a basis for this work.

Another path that should be pursued is to expand the system presented in this thesis to the other axis, namely those whose load varies depending on the tube being used and study how the different materials, diameters and geometries being produced affect the longevity of those axis. This means solving challenges such as how to deal with the movement of this axis varying widely depending on the part being manufactured by the client.

It is also proposed that tests are done with varying lubrication and that lubrication solutions that can be electronically controlled are studied, with the objective of creating a lubrication solution based on prescriptive maintenance.



# Bibliography

AMOB. e-mob52 — user manual, a.

AMOB. Amob - company presentation, January 2020b.

an emma, mamccrea, DCtheGeek, CHEEKATLAPRADEEP-MSFT-zz, v hearnya, JasonWHowell asd JSed225, MSshujia, changeworld, nschonni, neumannandaniel, and SnehaGunda and-carolinacmoravia. Azure stream analytics on iot edge, Dec 2020. URL <https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-edge>.

Microsoft Azure. Machine learning algorithms, 2021. URL <https://azure.microsoft.com/en-us/overview/machine-learning-algorithms/#uses>.

Nikita Bansal and S. Dubey. Image compression using hybrid transform technique. *Journal of Global Research in Computer Sciences*, 4:13–17, 2013.

Encyclopedia Britannica. October 30 2007. URL <https://www.britannica.com/technology/calendering>.

Gabriel H. Cassel Barbosa, Marcus Varanis, Kelven M. S. Delgado, and Clivaldo de Oliveira. An acquisition system framework for mechanical measurements with python, raspberry-pi and mems sensors. *Revista Brasileira de Ensino de Fisica*, 42, 00 2020. ISSN 1806-1117. URL [http://www.scielo.br/scielo.php?script=sci\\_arttext&pid=S1806-11172020000100500&nrm=iso](http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1806-11172020000100500&nrm=iso).

Haoming Chen and Bing Zeng. New transforms tightly bounded by dct and klt. *IEEE Signal Processing Letters - IEEE SIGNAL PROCESS LETT*, 19:344–347, 2012. doi: 10.1109/LSP.2012.2195172.

Analog Devices. Adxl345 datasheet and product info. 2013. URL <https://www.analog.com/media/en/technical-documentation/data-sheets/ADXL345-EP.pdf>.

HIWIN. Linear guideway technical information, 1998a. URL [https://www.hiwin.tw/download/tech\\_doc/bs/Ballscrew-\(E\).pdf](https://www.hiwin.tw/download/tech_doc/bs/Ballscrew-(E).pdf).

HIWIN. Linear guideway technical information, 1998b. URL [https://www.hiwin.com/pdf/linear\\_guideways\\_1.pdf](https://www.hiwin.com/pdf/linear_guideways_1.pdf).

- Hiwin, 2016. URL [https://www.hiwin.tw/support/bs\\_lifecalculate.aspx](https://www.hiwin.tw/support/bs_lifecalculate.aspx). Accessed: 2021-5-10.
- Hiwin. Sizing tool, 2021. URL <https://www.hiwin.de/en/service/auslegungstool>. Accessed: 2021-5-10.
- M. Iwaniec, A. Holovatyy, V. Teslyuk, M. Lobur, K. Kolesnyk, and M. Mashevskya. Development of vibration spectrum analyzer using the raspberry pi microcomputer and 3-axis digital mems accelerometer adxl345. In *2017 XIIIth International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH)*, pages 25–29. doi: 10.1109/MEMSTECH.2017.7937525.
- D. Jung, Z. Zhang, and M. Winslett. Vibration analysis for iot enabled predictive maintenance. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 1271–1282. ISBN 2375-026X. doi: 10.1109/ICDE.2017.170.
- Anil Katharotiya, Swati Patel, and Goyani Mahesh. Comparative analysis between dct & dwt techniques of image compression. *Journal of Information Engineering and Applications*, 1: 9–17, 2011.
- Seda Köseoğlu and Hasan Parlak. *Capacity Calculator for Rotary Draw Tube Bending*. Thesis, 2012. URL <http://www.lnu.se>.
- Wo Jae Lee, Haiyue Wu, Huitaek Yun, Hanjun Kim, Martin B. G. Jun, and John W. Sutherland. Predictive maintenance of machine tool systems using artificial intelligence techniques applied to machine condition data. *Procedia CIRP*, 80:506–511, 2019. ISSN 2212-8271. doi: <https://doi.org/10.1016/j.procir.2018.12.019>. URL <https://www.sciencedirect.com/science/article/pii/S2212827118312988>.
- Won Gi Lee, Jin Woo Lee, Min Sung Hong, Sung-Ho Nam, YongHo Jeon, and Moon G. Lee. Failure diagnosis system for a ball-screw by using vibration signals. *Shock and Vibration*, 2015: 435870, 2015. ISSN 1070-9622. doi: 10.1155/2015/435870. URL <https://doi.org/10.1155/2015/435870>.
- Lasse Lensu. *Discrete Cosine Transform*. 1998. URL <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.27.4601&rep=repl&type=pdf>.
- nagimov. `nagimov/adxl345spi`, Nov 2020. URL <https://github.com/nagimov/adxl345spi#spi-bus-and-sampling-rates>.
- Margot Note. *2 - Digital image basics*, pages 39–59. Chandos Publishing, 2011. ISBN 978-1-84334-599-2. doi: <https://doi.org/10.1016/B978-1-84334-599-2.50002-7>. URL <https://www.sciencedirect.com/science/article/pii/B9781843345992500027>.
- OMNI-X. *OMNI-X GUIDE TO ROTARY DRAW BENDING*. 2016. URL <https://www.omni-x.com/storage/file/42-catalog-omni-x-2016-en.pdf>.

- José Bessa Pacheco, José Manuel Oliveira, Marta Oliveira, and Ana Reis. Conceção ecológica de máquinas de curvar tubo. *Tecnometal*, 2019.
- Scipy, 2021. URL <https://www.scipy.org/>.
- ANNA MARIA SORIANI. *Analysis of process signals from rotary draw bending operations*. Thesis, 2018/2019. URL <http://hdl.handle.net/10589/151483>.
- STAM. Rotary draw bending vs compression bending, 2014. URL <https://staminc.com/rotary-draw-bending/rotary-draw-bending-vs-compression-bending/>.
- G. P. Sullivan, R. Pugh, A. P. Melendez, and W. D. Hunt. *Operations & Maintenance - Best Practices*, volume Release 3.0, page Chapter 5. August 2010.
- René Vinicio Sánchez. Diagnóstico de fallos en maquinaria rotativa mediante señales de vibración, corriente y emisión acústica empleando ia. In *Ciclo CV-2021. Ingeniería Mecánica en el siglo XXI. Nuevos retos y oportunidades*. Federação Iberoamericana de Engenharia Mecânica.
- René-Vinicio Sánchez, Pablo Lucero, Jean-Carlo Macancela, Mariela Cerrada, Rafael E Vásquez, and Fannia Pacheco. Multi-fault diagnosis of rotating machinery by using feature ranking methods and svm-based classifiers. In *2017 International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC)*, pages 105–110, 2017. doi: 10.1109/SDPC.2017.29.
- Peter Wayner, editor. *Compression*, book section Chapter 5, pages 73–101. Morgan Kaufmann, Boston, 2009. ISBN 978-0-12-374479-1. doi: <https://doi.org/10.1016/B978-012374479-1.50002-2>. URL <https://www.sciencedirect.com/science/article/pii/B9780123744791500022>.
- Yasdin Yasir, Alimin Alimin, and Siti Ramasih. Identification of induced draft fan (idf) damage in boiler waste gas system. *VANOS Journal of Mechanical Engineering Education*, 4, 07 2019. doi: 10.30870/vanos.v4i1.6111.



# Appendix A

```
1 import smbus as smbus
2 #import smbus
3 from datetime import datetime, timedelta
4 import time
5 from numpy import*
6 bus = smbus.SMBus(1) # Defines Bus as object of class Smbus
7 address = 0x53 # Accelerometer hexadecimal address
8 address_interference_accel=0x1D # Address of the accelerometer
   → mounted on the machine's body to subtract interferences from the
   → signal from the ball screw
9 # PARAMETERS AND RECORDS IN HEXADECIMAL ADXL345
10 ACCEL_2G = 0x00 #RANGE
11 ACCEL_4G = 0x01
12 ACCEL_8G = 0x02
13 ACCEL_16G = 0x03
14 SCALE_MULTIPLIER_2G = 4/1024 # MULTIPLICATORS FOR G
15 SCALE_MULTIPLIER_4G = 8/1024
16 SCALE_MULTIPLIER_8G = 16/1024
17 SCALE_MULTIPLIER_16G = 32/1024
18 BW_RATE_1600HZ = 0x0F
19 BW_RATE_800HZ = 0x0E
20 BW_RATE_400HZ = 0x0D
21 BW_RATE_200HZ = 0x0C
22 BW_RATE_100HZ = 0x0B
23 BW_RATE_50HZ = 0x0A
24 BW_RATE_25HZ = 0x09
25 BW_RATE_12_5HZ = 0x08
26 BW_RATE_6_25HZ = 0x07
27 BW_RATE_3_13HZ = 0x06
```

```

28 BW_RATE_1_56HZ = 0x05
29 BW_RATE_0_78HZ = 0x04
30 BW_RATE_0_39HZ = 0x03
31 BW_RATE_0_20HZ = 0x02
32 BW_RATE_0_10HZ = 0x01
33 BW_RATE_0_05HZ = 0x00
34 SCALE_REGISTER = 0x31 #RANGE CONTROL
35 BW_REGISTER = 0x2C # CONTROL BANDWIDTH
36 MODE_REGISTER = 0x2D # CONTROL THE MEASUREMENT MODE
37 MEASURE_MODE = 0x08 # MEASURE MODE
38 # Functions
39 def convert_LSBandMSB(raw_val): # CONVERTS DATA TO 10 BITS
40     convert_val = (raw_val [1] & 0x03) * 256 + raw_val [0]
41     if convert_val > 511:
42         convert_val -= 1024
43     return convert_val
44 def option_scale(SCALE):
45     if scale == int(2):
46         return [ACCEL_2G, SCALE_MULTIPLIER_2G]
47     elif scale == int(4):
48         return [ACCEL_4G, SCALE_MULTIPLIER_4G]
49     elif scale == int(8):
50         return [ACCEL_8G, SCALE_MULTIPLIER_8G]
51     elif scale == int(16):
52         return [ACCEL_16G, SCALE_MULTIPLIER_16G]
53 def option_banda(banda):
54     if banda == float(0.05):
55         return BW_RATE_0_05HZ
56     elif banda == float(0.10):
57         return BW_RATE_0_10HZ
58     elif banda == float(0.20):
59         return BW_RATE_0_20HZ
60     elif banda == float(0.39):
61         return BW_RATE_0_39HZ
62     elif banda == float(0.78):
63         return BW_RATE_0_78HZ
64     elif banda == float(1.56):
65         return BW_RATE_1_56HZ
66     elif banda == float(3.13):
67         return BW_RATE_3_13HZ

```

```

68     elif banda == float(6.25):
69         return BW_RATE_6_25HZ
70     elif banda == float(12.5):
71         return BW_RATE_12_5HZ
72     elif banda == float(25):
73         return BW_RATE_25HZ
74     elif banda == float(50):
75         return BW_RATE_50HZ
76     elif banda == float(100):
77         return BW_RATE_100HZ
78     elif banda == float(200):
79         return BW_RATE_200HZ
80     elif banda == float(400):
81         return BW_RATE_400HZ
82     elif banda == float(800):
83         return BW_RATE_800HZ
84     elif banda == float(1600):
85         return BW_RATE_1600HZ
86 # USER INPUTS
87 RANGE_XG=int()
88 scale = int(input("Scale [2,4,8,16]:"))
89 RANGE_XG = option_scale(scale)
90 BANDW_XHZ=float()
91 banda = float(input("Bandwidth [0.05, 0.1, 0.2, 0.39, 0.78, 1.56,
    ↪ 3.13, 6.25, 12.5, 25, 50, 100, 200, 400, 800, 1600]:"))
92 BANDW_XHZ = option_banda(banda)
93 freqInput = input("Measuring frequency[Hz]:")
94 #ene=input("'2^(:)'") #nsample
95 #nsamp=2int(ene) #nsample
96 minutes = input("Measurement Duration[m]:")
97 seconds = input("Measurement Duration[s]:")
98 duration = timedelta(0,int(seconds),0,0,int(minutes))
99 time.sleep(1)
100 run = input("Test number:")
101 # MEASUREMENT CONFIGURATION
102 bus.write_byte_data(address, MODE_REGISTER, MEASURE_MODE)
103 alem=bus.read_byte_data(address, MODE_REGISTER)
104 if alem == MEASURE_MODE:
105     print("CORRECTLY CONFIGURED MODE")
106 time.sleep(0.2)

```

```

107 bus.write_byte_data(address, BW_REGISTER, BANDW_XHZ)
108 allm=bus.read_byte_data(address, BW_REGISTER)
109 if allm == BANDW_XHZ:
110     print("BANDWIDTH CORRECTLY CONFIGURED")
111 time.sleep(0.2)
112 bus.write_byte_data(address, SCALE_REGISTER, RANGE_XG[0])
113 alum=bus.read_byte_data(address, SCALE_REGISTER)
114 if alum == RANGE_XG[0]:
115     print("SCALE CORRECTLY CONFIGURED")
116 print(str(RANGE_XG[1]))
117 time.sleep(1)
118
119
120 bus.write_byte_data(address_interference_accel, MODE_REGISTER,
    → MEASURE_MODE)
121 alem=bus.read_byte_data(address_interference_accel, MODE_REGISTER)
122 if alem == MEASURE_MODE:
123     print("CORRECTLY CONFIGURED MODE FOR SECOND SENSOR")
124 time.sleep(0.2)
125 bus.write_byte_data(address_interference_accel, BW_REGISTER,
    → BANDW_XHZ)
126 allm=bus.read_byte_data(address_interference_accel, BW_REGISTER)
127 if allm == BANDW_XHZ:
128     print("BANDWIDTH CORRECTLY CONFIGURED FOR SECOND SENSOR")
129 time.sleep(0.2)
130 bus.write_byte_data(address_interference_accel, SCALE_REGISTER,
    → RANGE_XG[0])
131 alum=bus.read_byte_data(address_interference_accel, SCALE_REGISTER)
132 if alum == RANGE_XG[0]:
133     print("SCALE CORRECTLY CONFIGURED FOR SECOND SENSOR")
134 print(str(RANGE_XG[1]))
135 time.sleep(1)
136
137
138 # TRANSFORM THE STRING FREQUENCY INPUT TO DATETIME
139 milsec = float(1000.0/float(freqInput))
140 if milsec==1000:
141     period=timedelta(0,1)
142 elif milsec%1==0:
143     period=timedelta(0,0,0,int(milsec))

```



```

144 else:
145     period=timedelta(0,0,(milsec-int(milsec))*1000, int(milsec))
146 # CREATING LIST WHERE DATA WILL BE RECORDED WITH APPEND
147 tempo=[]
148 accdata=[]
149 second_accdata=[]
150 t=0
151 time.sleep(1)
152 # start control
153 while True:
154     init = input("Start Data Acquisition (y/n):")
155     if init == "y":
156         break
157     else:
158         pass
159 # LOOP DATA ACQUISITION
160 print("Start of Data Acquisition")
161 previousMillis = datetime.now()
162 start = datetime.now()
163 endloop=datetime.now()+duration
164 while datetime.now().time() <endloop.time():
165     currentMillis = datetime.now()
166     if (currentMillis - previousMillis >= period):
167         previousMillis = datetime.now()
168         accdata.append([[bus.read_byte_data(address,
169             ↳ 0x32),bus.read_byte_data(address,
170             ↳ 0x33)], [bus.read_byte_data(address,0x34),bus.read_byte_data(address,
171             ↳ 0x35)], [bus.read_byte_data(address,0x36),bus.read_byte_data(address,
172             ↳ 0x37)])
173         second_accdata.append([[bus.read_byte_data(address_interference_accel,
174             ↳ 0x32),bus.read_byte_data(address_interference_accel,
175             ↳ 0x33)], [bus.read_byte_data(address_interference_accel,0x34),bus.read
176             ↳ 0x35)], [bus.read_byte_data(address_interference_accel,0x36),bus.read
177             ↳ 0x37)])
178         #accdata.append([[bus.read byte data(address, 0x36),bus.read byte
179             ↳ data(address, 0x37)])
180         tempo.append(datetime.now())
181     end=datetime.now()
182 acqt = end-start
183 print('| End of Data Acquisition |')
184 # CONVERSION OF INPUTS IN LSB AND MSB FOR GRAVITY

```

```

176 for i in range(len(tempo)):
177     for j in range(0,3):
178         accdata[i][j] = convert_LSBandMSB(accdata[i][j]) *
            ↳ RANGE_XG[1]
179         second_accdata[i][j] =
            ↳ convert_LSBandMSB(second_accdata[i][j]) * RANGE_XG[1]
180 # FILE OPENING
181 ttime=[]
182 opentime = datetime.now()
183 nome = 'ADXL_Main_Data_Unfiltered_run_'+ str(run)
184 saida = open(nome+".csv", "w")
185 for i in range(len(tempo)):
186     saida.write(str(i+1)+",")
187         # MEASUREMENT NUMBER
188     saida.write(str(tempo[i].time()+",") # TIME
189     saida.write(str((tempo[i]-tempo[0]).total_seconds()+",")
190         # Time from start to measurement
191     saida.write(str(accdata[i][0])+",")
192         # Accelerometer X
193     saida.write(str(accdata[i][1])+",")
194         # Accelerometer y
195     saida.write(str(accdata[i][2])+",")
196         # Accelerometer z
197     saida.write(str(acqt)+"\n")
198     ttime.append((tempo[i]-tempo[0]).total_seconds())
199 saida.close()
200
201 #WRITING DATA BY THE SECOND ACCELEROMETER ON A SEPARATE FILE
202 opentime = datetime.now()
203 nome_no_noise = 'ADXL_Dummy_Data'+str(run)
204 saida_no_noise = open(nome_no_noise+".csv", "w")
205 for i in range(len(tempo)):
206     saida_no_noise.write(str(i+1)+",")
207         # MEASUREMENT NUMBER
208     saida_no_noise.write(str(tempo[i].time()+",") # TIME
209
            ↳ saida_no_noise.write(str((tempo[i]-tempo[0]).total_seconds()+",")
210         # Time from start to measurement
211     saida_no_noise.write(str(second_accdata[i][0])+",")
212         # Accelerometer X

```

```

213     saida_no_noise.write(str(second_accdata[i][1])+",")
214         # Accelerometer y
215     saida_no_noise.write(str(second_accdata[i][2])+",")
216         # Accelerometer z
217     saida_no_noise.write(str(acqt)+"\n")
218     ttime.append((tempo[i]-tempo[0]).total_seconds())
219 saida.close()
220
221
222
223 ''' Important definitions '''
224 acc=transpose(accdata)
225 acc_dummy=transpose(second_accdata)
226 sigZ=acc[2]
227 sigZ_dummy=acc_dummy[2]
228 sigY=acc[1]
229 sigY_dummy=acc_dummy[1]
230 sigX=acc[0]
231 sigX_dummy=acc_dummy[0]
232 time_step=acqt.total_seconds()/len(sig)
233 Fs=1/time_step
234 ''' Starting to calculate FFT '''
235 from scipy import fftpack
236 sample_freqZ = fftpack.fftfreq(sigZ.size, d=time_step)
237 sample_freqY = fftpack.fftfreq(sigY.size, d=time_step)
238 sample_freqX = fftpack.fftfreq(sigX.size, d=time_step)
239     # generates sampling frequencies
240 sig_fft_Z = fftpack.fft(sigZ) # calculates the fast Fourier
    → transform for main sensor signal for the Z axis
241 sig_dummy_fft_Z = fftpack.fft(sigZ_dummy) # calculates the fast
    → Fourier transform for dummy sensor signal for the Z axis
242 sig_fft_Y = fftpack.fft(sigY) # calculates the fast Fourier
    → transform for main sensor signal for the Y axis
243 sig_dummy_fft_Y = fftpack.fft(sigY_dummy) # calculates the fast
    → Fourier transform for dummy sensor signal for the Y axis
244 sig_fft_X = fftpack.fft(sigX) # calculates the fast Fourier
    → transform for main sensor signal for the X axis
245 sig_dummy_fft_X = fftpack.fft(sigX_dummy) # calculates the fast
    → Fourier transform for dummy sensor signal for the X axis
246

```

```

247 pidxs = where(sample_freq> 0) # Only the positive part of the
    → spectrum will be used
248 freqsZ = sample_freqZ[pidxs]
249 freqsY = sample_freqY[pidxs]
250 freqsX = sample_freqX[pidxs]
251 powerZ = abs(sig_fft_Z)[pidxs]
252 powerY = abs(sig_fft_Y)[pidxs]
253 powerX = abs(sig_fft_X)[pidxs]
254 print ("| Finish FFT")
255 ''' Find the frequency according to the peak '''
256 N_freq=int(input('==== Enter number of frequencies to find:'))
257 xcoords=zeros((N_freq))
258 DT=int(len(freqs)/N_freq)
259 for i in range(N_freq):
260     fr=freqs[(i)*DT: DT*(i+1)]
261     xcoords[i]=fr[argmax(power[(i)*DT: DT*(i+1)])]
262 ''' Name to be identified '''
263 dataname=nome+".csv"
264 figname = 'ADXL_Time_Domanain_Run'+str(run)+'.png'
265 freqname='ADXL_Frequency_Domain_Run '+str(run)+'.png'
266 timefreqname='ADXL_Time_Frequency_Domain_Run '+str(run)+'.png'
267 ''' Gráfico '''
268 import matplotlib.pyplot as plt
269 plt.figure(1,figsize=[12,9])
270 plt.style.use('classic')
271 plt.subplot(3, 1, 1)
272 plt.plot(ttime,sig_fft_X,color='#2f4875',linewidth=0.8)
273 plt.xlabel('Time [s]')
274 plt.ylabel('Acceleration [g]')
275 plt.title('X Axis')
276 plt.subplot(3, 1, 2)
277 plt.plot(ttime,sig_fft_Y,color='#2f4875',linewidth=0.8)
278 plt.xlabel('Time [s]')
279 plt.ylabel('Acceleration [g]')
280 plt.title('Y Axis')
281 plt.subplot(3, 1, 3)
282 plt.plot(ttime,sig_fft_Z,color='#2f4875',linewidth=0.8)
283 plt.xlabel('Time [s]')
284 plt.ylabel('Acceleration [g]')
285 plt.title('Z Axis')

```

```

286 plt.legend()
287 plt.grid(True)
288 plt.savefig(filename)
289 plt.show(block=False)
290 plt.pause(5)
291
292 plt.figure(2,,figsize=[12,9])
293 plt.subplot(3, 1, 1)
294 plt.plot(freqsX, powerX,color='#2f4875',linewidth=0.8)
295 plt.xlabel('Frequency [HZ]')
296 plt.ylabel('Power [dB]')
297 plt.title('X Axis')
298 for xc in xcoords:
299
    → plt.axvline(x=xc,linestyle='--',linewidth=1,color='#FF4040',label=str(ro
300 plt.subplot(3, 1, 2)
301 plt.plot(freqsY, powerY,color='#2f4875',linewidth=0.8)
302 plt.xlabel('Frequency [HZ]')
303 plt.ylabel('Power [dB]')
304 plt.title('Y Axis')
305 for xc in xcoords:
306
    → plt.axvline(x=xc,linestyle='--',linewidth=1,color='#FF4040',label=str(ro
307 plt.subplot(3, 1, 3)
308 plt.plot(freqsZ, powerZ,color='#2f4875',linewidth=0.8)
309 plt.xlabel('Frequency [HZ]')
310 plt.ylabel('Power [dB]')
311 plt.title('Z Axis')
312 for xc in xcoords:
313
    → plt.axvline(x=xc,linestyle='--',linewidth=1,color='#FF4040',label=str(ro
314 plt.legend()
315 plt.grid(True)
316 plt.savefig(freqname)
317 plt.show(block=False)
318 plt.pause(5)
319
320
321 plt.figure(3,figsize=[12,9])
322 plt.style.use('classic')

```

```
323 plt.subplot(3,1,1)
324 powerSpectrum,frequenciesFound,time,imageAxis=plt.specgram(sigX,Fs=Fs,noverlap=255,cr
325 plt.xlabel('Time [s]')
326 plt.ylabel('Frequency [Hz]')
327 plt.title('X Axis')
328 plt.subplot(3,1,2)
329 powerSpectrum,frequenciesFound,time,imageAxis=plt.specgram(sigY,Fs=Fs,noverlap=255,cr
330 plt.xlabel('Time [s]')
331 plt.ylabel('Frequency [Hz]')
332 plt.title('Y Axis')
333 plt.subplot(3,1,3)
334 powerSpectrum,frequenciesFound,time,imageAxis=plt.specgram(sigZ,Fs=Fs,noverlap=255,cr
335 plt.xlabel('Time [s]')
336 plt.ylabel('Frequency [Hz]')
337 plt.title('Z Axis')
338 plt.savefig(timefreqname)
339 plt.show(block=False)
340 plt.pause(5)
341 print ("| Figure Saved |")
342 print ("| Completely Finish |")
```

## Appendix B

```
1 import os
2 import numpy as np
3 from matplotlib import mlab
4 import matplotlib.pyplot as plt
5 sample_rate_Hz = 3200
6 length_s = input('Duração?')
7 length_s = int(length_s)
8 ensaio = input('Ensaio?')
9 while True:
10     init = input("Start Data Acquisition (y/n):")
11     if init == "y":
12         break
13     else:
14         pass
15 print("Start of Data Acquisition")
16 os.system(f'sudo adxl345spi -t {length_s} -f {sample_rate_Hz} -s
   ↪ spi_ens_{ensaio}.csv')
17 print("Ended Data Acquisition")
18 print("Writing Acquired Data")
19 acc_data = np.genfromtxt(f'spi_ens_{ensaio}.csv', delimiter=',',
   ↪ names=True)
20 acc_x, freq_x, _ = mlab.specgram(acc_data['x'], Fs=sample_rate_Hz,
   ↪ NFFT=sample_rate_Hz * length_s)
21 acc_y, freq_y, _ = mlab.specgram(acc_data['y'], Fs=sample_rate_Hz,
   ↪ NFFT=sample_rate_Hz * length_s)
22 acc_z, freq_z, _ = mlab.specgram(acc_data['z'], Fs=sample_rate_Hz,
   ↪ NFFT=sample_rate_Hz * length_s)
23
24 plt.figure(1,figsize=[12,6])
```

```
25 plt.style.use('classic')
26 plt.subplot(3, 1, 1)
27 plt.plot(acc_data['time'], acc_data['x'], linewidth=0.5)
28 plt.xlabel('Time [s]')
29 plt.ylabel('Acceleration [g]')
30 plt.title('X Axis')
31 plt.subplot(3, 1, 2)
32 plt.plot(acc_data['time'], acc_data['y'], linewidth=0.5)
33 plt.xlabel('Time [s]')
34 plt.ylabel('Acceleration [g]')
35 plt.title('Y Axis')
36 plt.subplot(3, 1, 3)
37 plt.plot(acc_data['time'], acc_data['z'], linewidth=0.5)
38 plt.xlabel('Time [s]')
39 plt.ylabel('Acceleration [g]')
40 plt.title('Z Axis')
41 plt.legend()
42 plt.grid(True)
43 plt.savefig(f'Time_domain_spi_ens_{ensaio}.png')
44 plt.show(block=False)
45 plt.pause(5)
46 plt.close
47
48 plt.figure(2, figsize=[12, 6])
49 plt.style.use('classic')
50 plt.subplot(3, 1, 1)
51 plt.plot(freq_x[10:], acc_x[10:], linewidth=0.5)
52 plt.xlabel('Frequency [Hz]')
53 plt.ylabel('Power [dB]')
54 plt.title('X Axis')
55 plt.xlim((0, 1600))
56 plt.subplot(3, 1, 2)
57 plt.plot(freq_y[10:], acc_y[10:], linewidth=0.5)
58 plt.xlabel('Frequency [Hz]')
59 plt.ylabel('Power [dB]')
60 plt.title('Y Axis')
61 plt.xlim((0, 1600))
62 plt.subplot(3, 1, 3)
63 plt.plot(freq_z[10:], acc_z[10:], linewidth=0.5)
64 plt.xlabel('Frequency [Hz]')
```



```
65 plt.ylabel('Power [dB]')
66 plt.title('Z Axis')
67 plt.xlim((0,1600))
68 plt.legend()
69 plt.grid(True)
70 plt.savefig(f'Frequency_domain_spi_ens_{ensaio}.png')
71 plt.show(block=False)
72 plt.pause(5)
73 plt.close
74
75
76 plt.figure(3,figsize=[12,6])
77 plt.style.use('classic')
78 plt.subplot(3,1,1)
79 plt.specgram(acc_x[10:],NFFT=length_s*3200,
    ↪ Fs=3200,noverlap=255,cmap='Spectral')
80 plt.xlabel('Time [s]')
81 plt.ylabel('Frequency [Hz]')
82 plt.title('X Axis')
83 plt.subplot(3,1,2)
84 plt.specgram(acc_y[10:],NFFT=length_s*3200,
    ↪ Fs=3200,noverlap=255,cmap='Spectral')
85 plt.xlabel('Time [s]')
86 plt.ylabel('Frequency [Hz]')
87 plt.title('Y Axis')
88 plt.subplot(3,1,3)
89 plt.specgram(acc_z[10:],NFFT=length_s*3200,
    ↪ Fs=3200,noverlap=255,cmap='Spectral')
90 plt.xlabel('Time [s]')
91 plt.ylabel('Frequency [Hz]')
92 plt.title('Z Axis')
93 plt.savefig(f'Time_Frequency_domain_spi_ens_{ensaio}.png')
94 plt.show(block=False)
95 plt.pause(5)
96 plt.close
```



## Appendix C

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 from scipy.stats import kurtosis
5 from scipy.stats import skew
6
7 ens1 = pd.read_csv("spi_ens_1.csv", sep=',')
8
9 avgx=np.mean(ens1[' x'], axis=0)
10 print('The average x acceleration is:',avgx)
11
12 avgy=np.mean(ens1[' y'], axis=0)
13 print('The average y acceleration is:',avgy)
14
15 avgz=np.mean(ens1[' z'], axis=0)
16 print('The average z acceleration is:',avgz)
17
18 skx=skew(ens1[' x'], axis=0)
19 print('The skewness of x axis is:', skx)
20
21 sky=skew(ens1[' y'], axis=0)
22 print('The skewness of y axis is:', sky)
23
24 skz=skew(ens1[' z'], axis=0)
25 print('The skewness of z axis is:', skz)
26
27 enslarray=ens1.to_numpy()
28 aux1=0;
29 sscx=0;
```

```

30 for i in range(2, (len(ens1)-1)):
31     ↪ aux1=(enslarray[i][1]-enslarray[i-1][1])*(enslarray[i][1]-enslarray[i+1][1]);
32     if aux1>=0.02:
33         sscx=sscx+1
34     print('The slope sign change for a threshold of 0.02 for axis x is:
35         ↪ ', sscx)
36     aux1=0;
37     sscy=0;
38     for i in range(2, (len(ens1)-1)):
39         ↪ aux1=(enslarray[i][2]-enslarray[i-1][2])*(enslarray[i][2]-enslarray[i+1][2]);
40         if aux1>=0.02:
41             sscy=sscy+1
42     print('The slope sign change for a threshold of 0.02 for axis y is:
43         ↪ ', sscy)
44     aux1=0;
45     sscz=0;
46     for i in range(2, (len(ens1)-1)):
47         ↪ aux1=(enslarray[i][3]-enslarray[i-1][3])*(enslarray[i][3]-enslarray[i+1][3]);
48         if aux1>=0.02:
49             sscz=sscz+1
50     print('The slope sign change for a threshold of 0.02 for axis z is:
51         ↪ ', sscz)
52     aux1=0;
53     aux2=0;
54     aux3=0;
55     zcx=0;
56     for i in range(1, len(ens1)-1):
57         aux1=-enslarray[i][1]*enslarray[i+1][1]
58         if aux1<0:
59             aux2=0;
60         elif aux1==0:
61             aux2=0.5;
62         elif aux1>0:
63             aux2=1;

```

```
64  if aux2>0:
65      aux3=1
66  elif aux2==0:
67      aux3=0;
68  else:
69      aux3=-1;
70  zcx=zcx+aux3;
71  print('The zero crossing value for the x axis is: ', zcx)
72
73  aux1=0;
74  aux2=0;
75  aux3=0;
76  zcy=0;
77  for i in range(1, len(ens1)-1):
78      aux1=-ens1array[i][2]*ens1array[i+1][2]
79      if aux1<0:
80          aux2=0;
81      elif aux1==0:
82          aux2=0.5;
83      elif aux1>0:
84          aux2=1;
85      if aux2>0:
86          aux3=1
87      elif aux2==0:
88          aux3=0;
89      else:
90          aux3=-1;
91      zcy=zcy+aux3;
92  print('The zero crossing value for the y axis is: ', zcy)
93
94  aux1=0;
95  aux2=0;
96  aux3=0;
97  zcz=0;
98  for i in range(1, len(ens1)-1):
99      aux1=-(ens1array[i][3]-1.2)*(ens1array[i+1][3]-1.2)
100     if aux1<0:
101         aux2=0;
102     elif aux1==0:
103         aux2=0.5;
```

```
104     elif aux1>0:
105         aux2=1;
106     if aux2>0:
107         aux3=1
108     elif aux2==0:
109         aux3=0;
110     else:
111         aux3=-1;
112     zcz=zcz+aux3;
113     print('The zero crossing value (adjusted to offset the acceleration
114           ↪ reading of gravity by the sensor) for the z axis is: ', zcz)
115
115     max=np.amax(enslarray, axis=0)
116     min=np.amin(enslarray, axis=0)
117     hulx=max[1]-0.5*((max[1]-min[1])/(len(ens1)-1))
118     print('The histogram upper limit for the x axis is: ', hulx)
119
120     huly=max[2]-0.5*((max[2]-min[2])/(len(ens1)-1))
121     print('The histogram upper limit for the y axis is: ', huly)
122
123     hulz=max[3]-0.5*((max[3]-min[3])/(len(ens1)-1))
124     print('The histogram upper limit for the z axis is: ', hulz)
125
126     aux1=0;
127     for i in range(1,len(ens1)):
128         aux1=aux1+(abs(enslarray[i][1]))**0.5
129     aux1=aux1/len(ens1)
130     mfx=(max[1]/(aux1))**2
131     print('The margin factor for the x axis is: ', mfx)
132
133     aux1=0;
134     for i in range(1,len(ens1)):
135         aux1=aux1+(abs(enslarray[i][2]))**0.5
136     aux1=aux1/len(ens1)
137     mfy=(max[2]/(aux1))**2
138     print('The margin factor for the y axis is: ', mfy)
139
140     aux1=0;
141     for i in range(1,len(ens1)):
142         aux1=aux1+(abs(enslarray[i][3]))**0.5
```

```

143 aux1=aux1/len(ens1)
144 mfz=(max[3]/(aux1))**2
145 print('The margin factor for the z axis is: ', mfz)
146
147
148 kurt=kurtosis(enslarray, axis=0)
149 kurtx=kurt[1]
150 print('The kurtosis for the x axis is: ', kurtx)
151 kurty=kurt[2]
152 print('The kurtosis for the y axis is: ', kurty)
153 kurtz=kurt[3]
154 print('The kurtosis for the z axis is: ', kurtz)
155
156 aux1=0;
157 for i in range(1,len(ens1)):
158     aux1=aux1+(abs(enslarray[i][1]))**0.5
159 aux1=aux1/len(ens1)
160 lfx=max[1]/(aux1**2)
161 print('The latitude factor for the x axis is: ', lfx)
162
163 aux1=0;
164 for i in range(1,len(ens1)):
165     aux1=aux1+(abs(enslarray[i][2]))**0.5
166 aux1=aux1/len(ens1)
167 lfy=max[2]/(aux1**2)
168 print('The latitude factor for the y axis is: ', lfy)
169
170 aux1=0;
171 for i in range(1,len(ens1)):
172     aux1=aux1+(abs(enslarray[i][3]))**0.5
173 aux1=aux1/len(ens1)
174 l fz=max[3]/(aux1**2)
175 print('The latitude factor for the z axis is: ', l fz)
176
177 aux1=0;
178 for i in range(1,len(ens1)):
179     aux1=aux1+(enslarray[i][1])*(enslarray[i][1])
180 clx=(max[1])/((1/len(ens1)*aux1))
181 print('The clearence factor for the x axis is: ', clx)
182

```

```

183 aux1=0;
184 for i in range(1,len(ens1)):
185     aux1=aux1+(enslarray[i][2])*(enslarray[i][2])
186     cly=(max[2])/((1/len(ens1)*aux1))
187     print('The clearence factor for the y axis is: ', cly)
188
189 aux1=0;
190 for i in range(1,len(ens1)):
191     aux1=aux1+(enslarray[i][3])*(enslarray[i][3])
192     clz=(max[3])/((1/len(ens1)*aux1))
193     print('The clearence factor for the z axis is: ', clz)
194
195 standard_dev=np.std(enslarray, axis=0)
196 AvgStdRatioX=avgx/standard_dev[1]
197     print('The average-standard deviation ratio for the x axis is: ',
198           → AvgStdRatioX)
199 AvgStdRatioY=avgy/standard_dev[2]
200     print('The average-standard deviation ratio for the y axis is: ',
201           → AvgStdRatioY)
202 AvgStdRatioZ=avgz/standard_dev[3]
203     print('The average-standard deviation ratio for the z axis is: ',
204           → AvgStdRatioZ)
205
206 print(f"{avgx},{skx},{sscX},{zcx},{hulx},{mfx},{kurtx},{lfx},{clx},{AvgStdRatioX}")
207 print(f"{avgy},{sky},{sscy},{zcy},{huly},{mfy},{kurty},{lfy},{cly},{AvgStdRatioY}")
208 print(f"{avgz},{skz},{sscZ},{zCz},{hulz},{mfz},{kurtz},{lfz},{clz},{AvgStdRatioZ}")

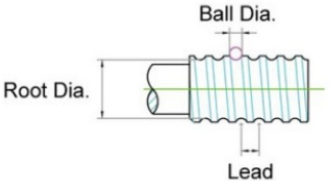
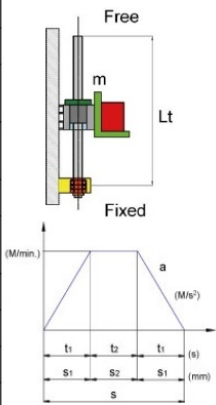
```





# Appendix D

## Life Calculation for HIWIN Ballscrew

Input request of BS specifications			
Accuracy	Rolled BS-without preload		
Model	38-10K4-FSC(DIN)		
Nominal Diameter	38(mm)		
Lead	10(mm)		
Type of nut			
Basic dynamic axial load rating C	5660(kgf)		
Basic static axial load rating Co	12410(kgf)		
Ball Diameter.	6.350(mm)		
Root Diameter Of BS	32.810		
Operating conditions			
Configuration	Vertical		
Mounting types	Fixed-Free		
Preload	0		
Distance between supported bearings Lt	440(mm)		
Table weight + Work piece weight m	1500(kgf)		
External Axial force Fa	25(kgf)		
Coefficient of friction of slider μ	0.05		
Velocity v	5.4(M/min)		
Acceleration a	0.6(M/s <sup>2</sup> )		
Acceleration time t1	0.15		
Total Stroke (S)	45(mm)		
Safety factor	1.2		
Calculation results			
Rotating Speed	540(rpm)	Expected rating Fatigue life	29.49x10 <sup>6</sup> (rev.)
Mean rotating speed	415(rpm)	Expected Service life in hours	1183(hours)
Average load	1832(kgf)	Expected Service life in distance	295(km)
	Customer operating conditions	HIWIN standard	Note
DN value	20520	-	-
Critical speed(rpm)	5768	-	-
Buckling load(kgf)	7617	-	-