GENERAL SIR JOHN KOTELAWALA DEFENCE UNIVERSITY

DEVELOP AND VALIDATE A PREDICTIVE MAINTENANCEMODEL USING ARTIFICIAL INTELLIGENCE

BY

O.D. RAMADASA (ENG/17/157)

C.O. ABEYSEKARA (ENG/17/156)

G.K. KALUTHANTRI (ENG/17/108)

SUPERVISED BY

MRS.H.M.S.M GASPE

MR. PRADEEP JAYATHILAKE

MR. D.H MARABEDDE

MR. SD AMARATHUNGA

THIS THESIS IS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF SCIENCE IN MECHATRONICS ENGINEERING

DEPARTMENT OF MECHANICAL ENGINEERING

FACULTY OF ENGINEERING

DECEMBER 2020

DECLARATION OF AUTHORSHIP

We, O.D. Ramadasa, C.O. Abeysekara and G.K. Kaluthantri declare that this thesis titled "Develop and validate a predictive maintenance model using artificial intelligence" and the content presented in it, is our own. We conform that,

- This work was done wholly or mainly in candidature for a degree at this University.
- Where we have consulted the published work of others, which has been always clearly attributed.
- Where we have quoted from the work of the others, the source is always given. Except for such quotations, this thesis is entirely our own work.
- We have acknowledged all main sources of help.

ENG / 17 / 157	O. D. Ramadasa	…….
ENG / 17 / 156	C. O. Abeysekara	……..…….….
ENG / 17 / 108	G.K. Kaluthantri	...…….…….

Date: …….

DECLARATION BY SUPERVISOR

I certify that the above statement made by the authors is true and that this thesis is suitable for submission to the university for the purpose of evaluation.

……..

MRS. H.M.S.M GASPE

Supervisor

Lecturer

Department of Mechanical Engineering,

Faculty of Engineering,

General Sir John Kotelawala Defense University

……..

MR. KDPR JAYATILAKA
Supervisor

Lecturer

Department of Mechanical Engineering,

Faculty of Engineering,

General Sir John Kotelawala Defense University

……..

MR. D.H MARABEDDE

Supervisor

Manager,

Automation Department,

Atlas Axillia Pvt. Ltd,

96, Parakrama Road, Peliyagoda, Sri Lanka

……..

MR. SD AMARATHUNGA

Supervisor

Engineer,

Automations Engineering Department,

Atlas Axillia Pvt. Ltd,

96, Parakrama Road, Peliyagoda, Sri Lanka

ACKNOWLEDGEMENT

Our sincere appreciation goes toward the Department of Mechanical Engineering of General Sir John Kotelawala Defence University and our supervisors, Mrs. H.M.S.M Gaspe and Mr. K.D.P.R. Jayathilake for their guidance and continuous support.

This research project was financially supported by Atlas Axillia (Pvt) Ltd, whom we are eternally grateful for. We extend our gratitude to the Department of Automations and the Department of Maintenance Engineering of Atlas Axillia (Pvt) Ltd, and special thanks goes to Mr. D.H. Marabedda, Manager – Department of Automations, Mr. S.D. Amarathunge - Department of Automations, Mr. A.N. Basnayake - Department of Automations, Mr. Shirantha Chularathna – Head of Maintenance.

I must appreciate the guidance given by the maintenance team as well as the machine operators specially in our data collection and machine level analysis.

Finally, our gratitude goes towards our families, friends, and other relatives, for accommodations, transport, funding, and morale support in this pandemic situation.

ABSTRACT

Industry 4.0, is the fourth and foremost Industrial Revolution or the approach in the automation of manufacturing industry, using technology integrated with IoT and real time condition monitoring. Introducing a cloud based Predictive Maintenance (PdM) model (For bearings in Phase I) will be a turning point for a production system that is switching to Industrial 4.0. By this project Atlas Axillia (Pvt) Ltd, the leading stationary manufacturing company in Sri Lanka can take an initiative to the more sophisticated PdM with the help of IoT. For our model initiation, a traditional exercise book manufacturing machine (Bielomatik P490) with complex mechanical components and noisy vibrations was used. 3 main critical points were identified and analyzed separately for the most crucial ones. The main reason the machine was selected is that the machine is facing some over maintenance conditions due to leftover life in bearings. By implementing a predictive maintenance system for Bielomatik P490, it is expected to minimize the annual maintenance cost of Rs.7 million. Machine learning based pattern recognition methodology is applied with vibration monitoring system to predict the potential failures of the production system. In addition, the precautions for the potential failure will be scheduled to the next maintenance program so the maintenance team can take necessary actions. Furthermore, a Maintenance Dashboard was introduced so the upper management can also interfere with the machine level Maintenance activities. Additionally, this maintenance model, which was focused on basically bearings, can be expand further for other locations where the PdM is essential.

Keywords: Predictive Maintenance, Machine Learning, Maintenance, Maintenance Dashboard, Patten Recognition, IoT

TABLE OF CONTENT

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 Background

1.1.1 Industrial 4.0

Industry 4.0, is the fourth and foremost Industrial Revolution or the approach in the automation of manufacturing industry, using technology integrated with IoT and real time condition monitoring. This concept continues to generate production companies' attention and have invested more on AI development and IoT solutions.

1.1.2 Our Scope

Introducing a cloud based PdM model will be a turning point for a production system that is switching to Industrial 4.0. By this project Atlas Axillia (Pvt) Ltd, the leading stationary manufacturing company in Sri Lanka can take an initiative to the more sophisticated PdM with the help of IoT.

For our model initiation, a traditional exercise book manufacturing machine with complex mechanical components and noisy vibrations is used. Critical Points were identified and analyzed separately for the most crucial ones. By implementing a predictive maintenance system for Bielomatik P490, it is expected to minimize the annual maintenance cost of Rs.7 million (Bearing cost – 2.8M, Workshop cost 2.1M, Parts and other 2.1M) and reduce monthly maintenance costs.

1.1.3 Objectives

1. Analyze the critical points of the machine with a higher possibility of failure.
2. Analyze the failing methods of critical points.
3. Establishing a methodology to acquire data from critical points.
4. Acquiring and Analyzing data sets and identify the deviation of those critical points when there is a failure.
5. Training and Validating the Model
6. Implementing a Maintenance Dashboard
6. Testing and troubleshooting.

CHAPTER 2: LITERATURE REVIEW

2.1 Industrial 4.0

The main goal to use modern technology and data for increase productivity and efficiency. By using sensors received data we have to extract useful information to reduce cost, optimize the capacity and most importantly keep downtime to a minimum. Industry 4.0 or the new era of Intelligent machining and intelligent manufacturing, including reliance on Cyber-Physical Systems (CPS), Computer integrated Manufacturing Systems, IoT integrated systems and implementation and operation of smart factories with smart workforce. Mass scale production or mass scale supply chain management procedures in the present global environment have a much complex anatomy. To deal with such complexity smart machines and smart labors are essential. Smart machines can be defined as the systems which can provide feedbacks so the machine itself can prepare for the changes or other stimulus in the system such as breakdowns and efficiency changes. Smart factory basically focuses on Increased production Volume and Efficiency if needed, Improve Product Quality and maintaining the quality, Reduce Manufacturing Costs including repair costs, Open Opportunities for Expansion in every scale. Based on the reviews and critics, it is mentioned that manufacturing companies; small or large scale need to focus more on sustainability and make use of technologies like 'Internet of Things' (IoT) and Predictive Maintenance to meet the demand changes and organizational goals. The main reason is that the resources are limited, and the demand is increasing.

2.2 Predictive Maintenance

2.2.1 Preventive maintenance (PM)

Preventive maintenance is an old-fashioned approach when dealing with production-based machine maintenance, which will be secluded to be performed machinery to reduce the potential of being a failure. It is mostly performed on the machinery while the machine is on the operating condition so the maintenance process will not interrupt the production process. Some occasions require the shutdown of the machine, so the time and the schedules are carefully selected.

When considering the outcome, it is expected to run the machine for its maximum life expectancy with the promising efficiency. So, the preventive maintenance provides its benefits, but with many limitations. The preventive maintenance models are more labor-intensive, the concurrent working of the labors without any interruption. And the repetition of the same work

plan is expected. The most crucial and the most considerable disadvantage is the potential of over-maintenance. Some parts and machinery may face a failure (i.e., Bearings) or wear off due to over-maintenance and the investment will not be promising. The benefits of predictive or simply condition based maintenance, such as continuous condition monitoring of equipment and machinery while predicting the date when maintenance or changeover should be planned, are extremely beneficial and strategic.

2.2.2 Predictive maintenance (PdM)

In general prediction is a more of a statement, declaration or indication of future events based on historical data, observation with limited scope, deduction and assumptions, instinct based decisions, or arrived by means of a mathematical approach. In predictive maintenance (PdM) scenarios, data is gathered with set conditions over time as a time series to monitor the state of machine. The aim is to identify any patterns that can help predict and ultimately prevent potential failures which may not be sensible to the traditional preventive maintenance. This can be optimized by introducing machine learning with lesser limitations. That the implementation of Machine learning (ML)-based prediction system for a machine or an equipment can led to a definite cost saving, better reliability, and the increased the availability of the production systems.

In summary, the PdM process can be simplified into two steps.

1. An observation an analysis of equipment failure characteristics in a time series.
2. A schedule to maximize the availability of machinery and instruments through the production process.

A PdM model will only require the relevant data to do a accurate prediction for an informal mathematical model on when will the machine is in a need of a repair or even replacement or a changeover, so that maintenance team can performed what is required, on time rather than performing a over-maintenance. Of course, some high-end production companies perform PdM, with SCADA systems, and the human-coded thresholds, human-coded limitations, alert rules, and other bounded rules. This semi-automated system or the approach cannot perform well on complex dynamic behaviors with many noises. It will not perform well under complex manufacturing process as large.

2.2.2.1 Advantages of predictive Maintenance

- Reduction in maintenance costs compared to Preventive Maintenance (PM)

When considering the maintenance cost, the actual labor cost, spare part cost, and other equipment cost can be reduced up to 50% according to past scenarios.

- Reduction in machine failures

Failures may sometimes give early warnings which may not be sensible to human senses. i.e., Anomalies in vibrations, thermal condition or change in power demand. Regular observation and prediction can prevent these failures including fatigue failures by PdM model.

- Reduced downtime for repairs

  In industry prevention is always better than a repair. During a repair, the idle time of machine and the labor can cost the company in much larger scale. Preventing a critical failure with prediction techniques can save the above-mentioned failures.

- Increased service life of parts

Other than the frequency of repairs or replacements, severity of machine damage and the actual machine condition after the replacement. It may not be the same as the original

2.3 Mechanical Component Analysis

2.3.1 Bearings

The main concept of using a bearing to a rotating or linearly moving object is to reduce the friction hence reduce the effort. It was introduced back in the time where some applications consumed more energy and effort for sliding and rolling.

The modern bearings serve the same function; to lessen the friction. Today, the bearings are used in many applications developing a toy to developing an aero plane. When it comes the scale, bearings provide the full flexibility regardless of the application.

2.3.1.1 The parts of a conventional bearing

To work properly, a bearing needs to work coherently with the following parts without any defect.

1. Outer race (also known as outer ring)
2. Inner race (also known as inner ring)

3. Rolling elements (also known as balls/ rollers)
4. Separator (also called as cage/ retainer)

The outer race is the bearing's exterior or the outmost ring. Its function is to protect the bearing's internal parts including ball elements, it must be very precise when machining. The inner race is the part where the bearing touches the shaft. The roller elements, commonly shaped as balls or rollers, provide the surface that reduces the moving friction of the shaft or any other supporter within its bearing housing. This combination of elements keeps the outer and inner races as two and enable them to move with lesser friction.

The shape or the type of the rolling elements most probably will depend on the type of the load, particular applications, and operating conditions. Rolling elements will distinguish the two basic bearing categories: ball bearings and roller bearings. There is a groove, or a space called the ball path on both the inner and outer parts of the bearing in which the balls roll have a clearance to move. For roller bearings, the surfaces are flat, so the guided rollers can move in between them. These surfaces are called roller paths. Finally, the separator is a metal holder for balls or rollers. Its Positioned between the inner and outer races, the separator keeps the rolling elements evenly spaced for ease movement.

2.3.2 Functions of a Bearing

A bearing is basically designed to:

1. Reduce friction.
2. Support a load from single or double end.
3. Guide moving parts – shafts, wheel, sprockets, and pivots (most common)

2.3.2.1 Reduce friction.

Whether they are used in naval, auto mobile, or industrial automation or machinery applications, bearings will perform the common function and have the same aim – to keep the moving part smoothly and consistently while reducing friction for low power consumption and wear.

2.3.2.2 Support a load.

A shaft, eventually loaded, will push the bearing in the same direction in which the load tries to move. The load is dependent on both weight and direction. It is crucial to select a bearing very careful because the wrong choice may face a potential failure.

There are three types of loads conditions,

1. When the direction of the weight being moved by creating a right angle to the exciting shaft, it is called as a radial load. The load pushes down the bearing.

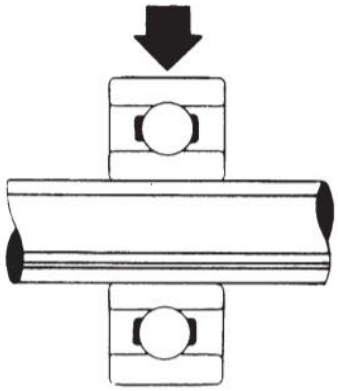


Figure 1 - Radial Load of a bearing

2. When the direction of the load is applied parallel to the existing shaft, it is called as a thrust. The load will push on the sideways of the bearing.

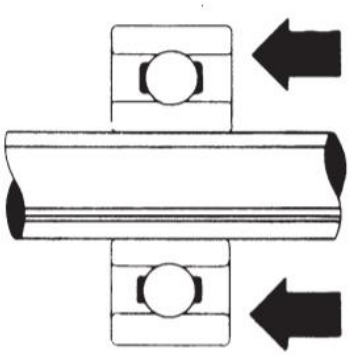


Figure 2 - Thrust load of a bearing.

3. When the direction of the load is a combination of radial load and a thrust load, the load pushes down sideways (combination of parallel and perpendicular) on the bearing. This combination is called as an angular load.

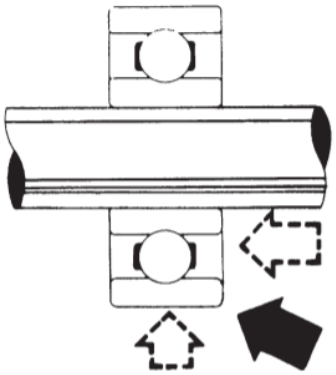


Figure 3 - Angular load of a bearing

2.3.2.3 Guide moving parts.

By supporting a loads and weights while reducing friction and wear, a bearing guides shaft during the operation. It assists the movement of crucial shafts, wheels, sprockets, and pivots. Without a bearing, the rotating part will just slip on the stationary part creating wear and high heats.

2.3.3 Operating conditions

2.3.3.1 Tolerance

When it comes to finish, size, loading conditions and diameter requirements, all bearings must meet certain standards. As an example, AFBMA (Anti-Friction Bearing Manufacturers Association) standards, regardless of the manufacturer's standards of certain bearings. In tapered type roller bearings, cups are interchangeable. No matter how refined or sophisticated the production method, there are variations in the manufacturing that will affect the operation conditions and bearing's dimensions.

Figure 4 - Tolerance

2.3.3.2 Internal bearing clearance

Clearance is essential so that the rollers have space to rotate without developing up extra friction and heat during its operation. Some components may expand due to the heat and reduce the tolerance further. The amount the inner race will rotate opposite to the outer race, under a given radial or thrust load, is called bearing clearance. This is a measurement of how much room there is between the internal parts, during the operation.

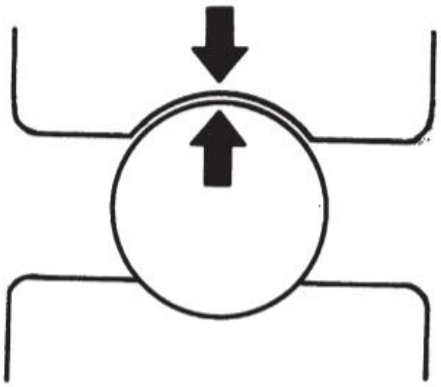


Figure 5 - Bearing Clearance.

2.3.3.3 Bearing seats.

Bearing races are mounted on machines or parts where the areas called “seats.” The cup seat will be the housing, while the cone seat will be the shaft. Within these two so called seats are upward extensions on which the inner and outer races rest. They are called as “shoulders.”

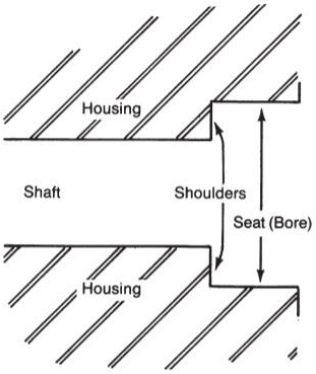


Figure 6 - Bearing Seats.

2.3.3.4. Alignment

The bearing cup, cone seat and most importantly the shaft and the housing must be properly aligned by any mean. A slight Misalignment will reduce the expected loading capacity and bearing life which will be measured in revolutions, proportionately to the amount of misalignment even the slightest.

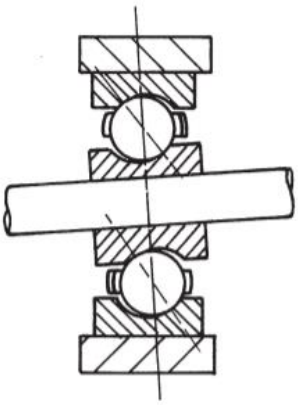


Figure 7 - Alignment of a bearing

2.3.3.5 Operating temperature

Type of the load, amount of friction and shaft speed, all contribute to one of the most crucial and most prominent conditions for its operation, the temperature. Each component of the sol called bearing must tolerate certain amounts of temperature fluctuations because the friction can be minimized by cannot prevent. So most importantly not all heat will be absorbed from the environment.

The bearings most frequently themselves may cause excessive heat. As examples,

1. Too heavy load conditions, resulting in misshapen racers and rollers.
2. Friction which can prevent or cannot, between the rolling elements, retainer, and races.
3. Excessive churning effect, from too much lubricant or wrong lubricant.
4. Surface friction, from too little lubricant or wrong lubricant.

2.3.3.6 Lubrication

Using the perfect type and quantity of lubricant for the application is another factor crucial for bearing performance and thermal conditions. Whenever bearing use causes excess friction proportionally heat will rise. Regular lubrication in preventive maintenance will help relieve the heating conditions that will results in bearing friction and wear.

2.3.3 Bearing types.

Our research is basically based on ball bearings because the test machine consists ball bearings in its critical points.

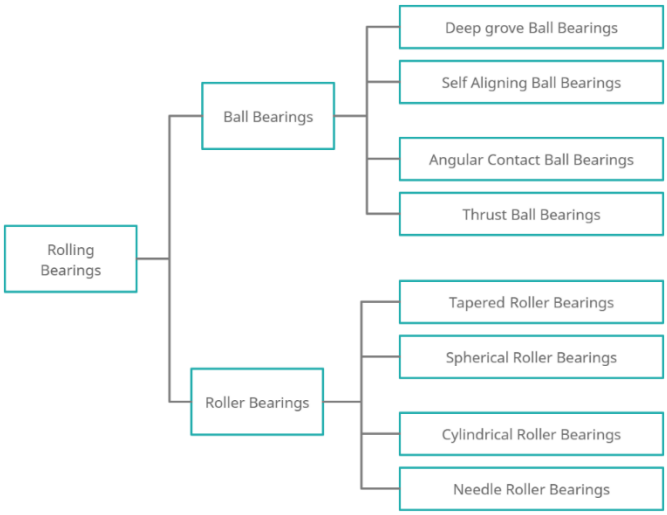


Figure 8 - Bearing Types.

2.3.5. Ball Bearings

Ball Bearings uses metal or ceramic balls as the rolling elements. They are characterized by point-to-point contact between the balls and the support raceways. As a rule, ball bearings operate in much higher speeds but not ideal for higher load conditions.

2.3.5.1. Deep-Groove Ball Bearings

The most common bearings are Deep-Groove Ball Bearings. Due to its simple design, they are easy to maintain and not as sensitive to operating conditions thus are used in a wide range of different applications. In addition to radial forces, they absorb axial forces in both directions.

Figure 9 - Deep Grove Ball Bearing

Their low torque also makes them suitable for high speeds.

2.3.5.2 Angular Contact Ball Bearings

Angular Contact Ball Bearings are commonly characterized by its contact angle. This means that forces are transferred through balls from one raceway to the other at a not perpendicular but at a particular angle. Angular-contact ball bearings are much suitable for combined loads with axial and radial loads.

Figure 10 - Angular Contact Ball Bearings

2.3.5.3 Self-Aligning Ball Bearings

Self-Aligning Ball Bearings have two rows of balls guided by a single cage and two row inner ring raceways but have the special feature of a continuous spherical outer ring raceway

allowing the inner ring / ball complement to swivel within the outer ring. This is what enables a degree of self-alignment in the application.

This type of bearing is recommended when alignment of the shaft and the housing (misalignment) are a problem, and the shaft could deflect. Self-aligning ball bearings are most suitable for absorbing radial forces.

Figure 11 - Self-Aligning Ball Bearings

2.3.5.4. Thrust Ball Bearings

Thrust Ball Bearings consist of two bearing discs with raceways for the balls. Thrust ball bearings were developed solely for absorbing axial forces in one direction, meaning they can locate the shaft axially in one direction.

Figure 12 - Thrust Ball Bearings

2.3.6. Roller Bearings

Roller Bearings are characterized by line contact. Line contact offers higher load rating than ball bearings of the same size; however, the speed ability is lower than a ball bearing due to the increased friction of a contact line.

2.3.6.1. Spherical Roller Bearings

Spherical Roller Bearings are very robust and work on the same principle as Self-aligning bearings with the exception that they use spherical rollers instead of ball rollers allowing higher loads to be supported. This can compensate for misalignments between the shaft and the housing.

Spherical roller bearings are suitable for absorbing high radial loads and moderate axial loads.

Figure 13 - Spherical Roller Bearings

2.3.6.2. Cylindrical Roller Bearings

Cylindrical Roller Bearings use line contact between the rolling elements and the raceways, which optimizes the distribution of stress factors at the point of contact. This arrangement means that cylindrical roller bearings have a very high radial load rating.

Depending on the design, they may also be able to transmit limited amounts of axial loads.

Figure 14 - Cylindrical Roller Bearings

2.3.6.3. Tapered Roller Bearings

Tapered Roller Bearings have tapered raceways in the inner and outer rings with conical rollers arranged between them.

Due to the contact angle, tapered roller bearings can absorb high radial and axial forces in one direction.

Tapered roller bearings are often combined in pairs to support axial forces in both directions.

Figure 15 - Tapered Roller Bearings

2.3.6.4. Needle Roller Bearings

Needle Roller Bearings are a special type of cylindrical roller bearing which contain long, thin rolling elements, known as needle rollers. The ratio of diameter to length is between 1:3 and 1:10. Needle roller bearings have a high load rating and are only suitable for radial forces.

If space is constricted, needle bearings can be a good solution.

Figure 16 - Needle Roller Bearings

2.3.7. Ball Bearings – Further Classification

The application we interested in basically include ball bearings. As a group, ball bearings have many uses in trucks, cars, and off-the-road vehicles. Some of the most common are in steering, assemblies, transmissions, and differentials. In other applications, such as heavy-duty wheel hubs, they have been replaced by roller bearings.

2.3.7.1 Operation

Though ball bearings and roller bearings share the same objective – to lessen friction – their strategies are quite different. The mechanical forces underlying ball bearing operation are simple to understand. When a ball bearing is inactive and still, the load applied will be distributed evenly through the races and balls on the contact area. Once the bearing is nudged

by a moving load, the ball starts to roll. Material in the race bulges out in front of the ball, then flattens out behind the ball. The ball flattens out in the lower front quadrant, then bulges in the lower rear quadrant. This process continues for each ball as long as the load is in motion. Continual metal-to-metal contact between the balls and races will eventually wear down the parts and result in bearing failure. So even in doing its job – to lessen friction between two surfaces – the bearing creates its own internal friction. This is one reason why lubrication within the bearing is critical in relieving friction.

2.3.7.2 Single row ball bearing

The single row is one of the most popular ball bearing designs. A crescent-shaped cut in both the inner and outer races forms a wide groove in which a single row of balls roll.

Though designed primarily for radial load capacity, this bearing can support substantial thrust loads in either direction, even at high operating speeds. Careful alignment between the shaft and housing is critical to its performance. The bearing is available with seals and shields for extra protection against contaminants, plus retention of lubricant. A variation of the single row bearing is the maximum capacity bearing. Additional balls can be assembled in the bearing for greater radial load capacity (fig. 7). However, the extra loading area limits the bearing's thrust load capacity.

2.3.7.3. Benefits/advantages

1. Good performance under radial loads
2. Deep groove permits thrust load capacity in either shaft direction.
3. Assures contaminant-free operation when seals are mounted on the bearing.

2.3.7.4. Applications

1. Transmission
2. Alternator
3. Differential
4. Steering gear
5. Air conditioner clutch

2.4 Piezoelectric Accelerometers

An Piezoelectric accelerometer is a sensor module that can generates an electrical signal (Analog) proportional to its applied acceleration or simple vibration.

Piezoelectric accelerometers are designed to measure low and high frequency vibration and shock (High amplitude vibration for a shorter period) for a wide variety of applications including Industrial applications. Predictive maintenance models, that are relying on vibration data, can take an Piezoelectric accelerometer as the main sensor module for reading vibration data. They are much simpler to use and much more accurate and reliable over a wide frequency ranges (low and high, which makes the Piezoelectric accelerometer the ideal choice for many testing and operating conditions.

The electronic circuitry in the embedded module will convert a high-impedance charge signal with some noises, generated by a piezoelectric sensing element (Piezoelectric accelerometer) into a usable low-impedance voltage signal (0-10V, in our application) using a signal conditioner that can be readily readable by an analog module, over ordinary two-wire (BNC connected) or coaxial cables to any data acquisition system or readout device.

Figure 17 - RVVP Cable

Figure 18 - Industrial Grade Accelerometer

2.4.1. Low Frequency Accelerometers

Piezoelectric accelerometers are generally AC coupled devices and they will not respond to uniform acceleration or linear acceleration (static or DC acceleration). Capacitance and resistance values internal to the accelerometer set the (DTC) Discharge Time Constant and low frequency response. If uniform acceleration is applied which will not constantly detect, the output signal will decay in the remaining time according to the DTC.

2.4.2. Construction of a Piezoelectric Accelerometers

A vast range of mechanical designs are considered to perform the transducer action required for a Piezoelectric accelerometer. Preload ring or stud applies a force to the sensing element assembly to make a rigid structure and insure linear behavior. Under acceleration, the seismic mass causes stress on the sensing crystals which results in a proportional electrical output. The

output is collected on electrodes and transmitted by 2 wires connected to the microelectronic circuitry in Piezoelectric accelerometers.

2.4.3 Piezoelectric Accelerometers – As a system

All Piezoelectric accelerometers which are an active transducer, require electrical power from a constant current DC voltage source. In general, applications the signal conditioner will provide the necessary power requirement. Piezoelectric offers different types of signal conditioners that provide 2 to 20 mA (4mA constant in our case) of current at a DC voltage level of +18 to +30 volts (24V in our model).

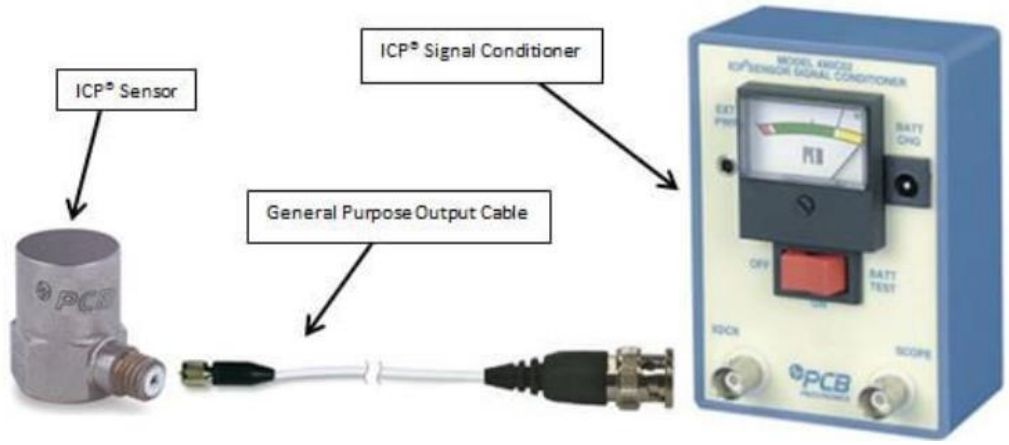


Figure 17 - Accelerometer with a signal conditioner

2.4 Condition Monitoring

In the machine condition monitoring, these could be a rotating machine-like pump, compressor, gas or steam turbine, motor etc. or non-rotating machine-like heat exchanger, distillation column, valves etc. In this any kind of machine type there need to detect points of poor health conditions. That detecting point might not be the failure point of the machine. But after that point machine would no longer run on the optimal condition. After detecting these points, the system needs maintenance activities to restore the condition.

Typically, in data driven condition monitoring systems worked on getting sensor readings and perform basic filtering to check weather received data reaching maximum or minimum limit points which previously defined. When reach any limits the system sends alarm signal and to inform the health conditions.

This kind of hard coded system with preassigned limits could send large number of false alarms when machine on the healthy condition. Also, there could occur alarm missing that would lead to increase the breakdown time of the machine and damage components.

Because of that there need to develop a health states deciding system based on the combination of various measurements to get a true indication of the situation in a complex production machine. Hongyu Yang, Joseph Mathew & Lin Ma (2002) concluded that using intelligent fault diagnosis of machinery could overcome the limitations alike efficiency, reliability, precision, and real time manner.

2.5.1 Fault prediction model development

During the literature review, discussed about many artificial intelligence and statistical analysis methods for bearing condition monitoring based on many research papers, books and articles. In conclusion we decided on perform analysis based on anomaly detection with theoretical analysis in this predictive maintenance system.

In this case mainly focused about Principal component analysis (PCA) with dimensional reduction and Artificial Neural Network (ANN) based on autoencoder networks. In theoretical analysis, referred about Ball pass frequencies in different defect types to label received data to predict fault type in future development.

2.5.1.1 Anomaly Detection

This can also refer as outlier detection. It is simply the mode of detecting and identifying anomalous data in any data-based event or observation that differs mainly from the rest of the data. This anomalous data can be critical in detecting a rare data pattern or potential problem in the form of financial frauds, medical analysis, in manufacturing process.

When we consider about how could detect anomalies in data, consider about an anomaly detection application which using two variables (X & Y) data.

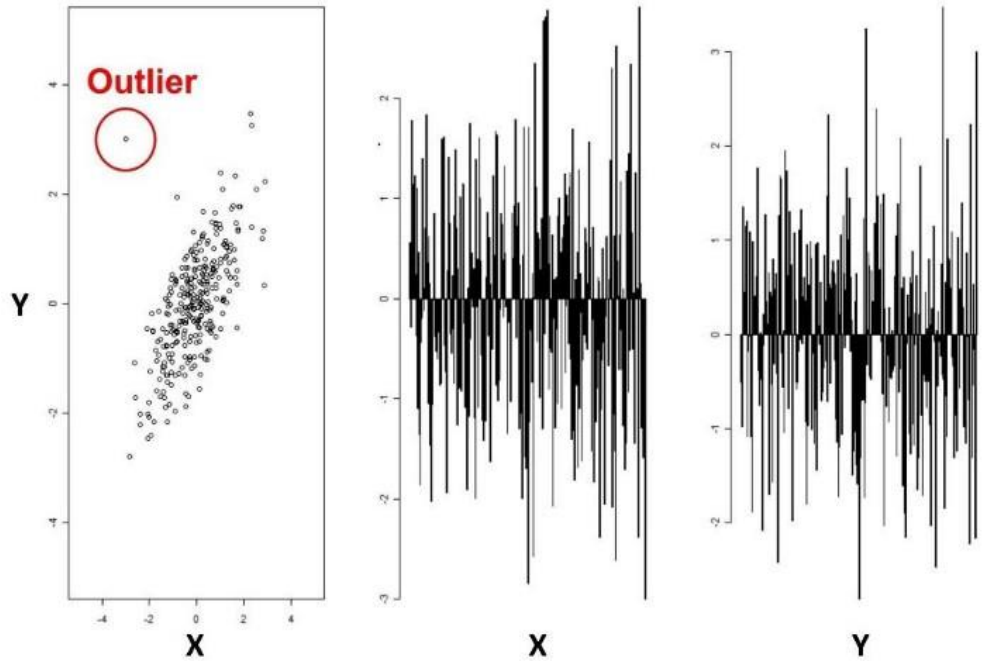


Figure 18 - Plot using two variables (X & Y) data

In this above plotted graph of two variables in the right side, it is difficult to detect any anomaly with that data. But when two data plotted against each other as right side in the figure it would help to identify anomaly data easily. This is the main reason of use artificial intelligence techniques on anomaly detection applications. Because detect outliers on big data could be challenging in real world applications with high complexity.

2.5.1.2 Principal component analysis (PCA)

Principle component analysis is an established technique in machine learning. PCA frequently use in exploratory data analysis because it reveals the inner structure of the data and supports to explain the variance of data. Nagdev Amruthnath, Tarun Gupta (2018) implemented an unsupervised ML model for early fault detection and they found that PCA model featuring promising results in anomaly detection. In this kind of application, we must work with data in multiple dimensions and it would be challenging. To reduce the complexity of that there need to perform some techniques to reduce the number of variables. Dimensionality reduction is a main way to fulfill that requirement. PCA is a dimensionality reduction method that is often use in large data sets.

It performs a linear mapping of the data to a lower dimensional space in such a way that the variance of the data in the low dimensional representation is maximized. When performing, the covariance matrix of the data is constructed, and the eigenvectors of this matrix will be calculated. The eigenvectors that correspond to the largest eigenvalues which known as principal components be used to reconstruct a large fraction of the variance of the first original data. The original feature space would be reduced with some data loss, but it would retain the most important variance to the space spanned by a few eigenvectors.

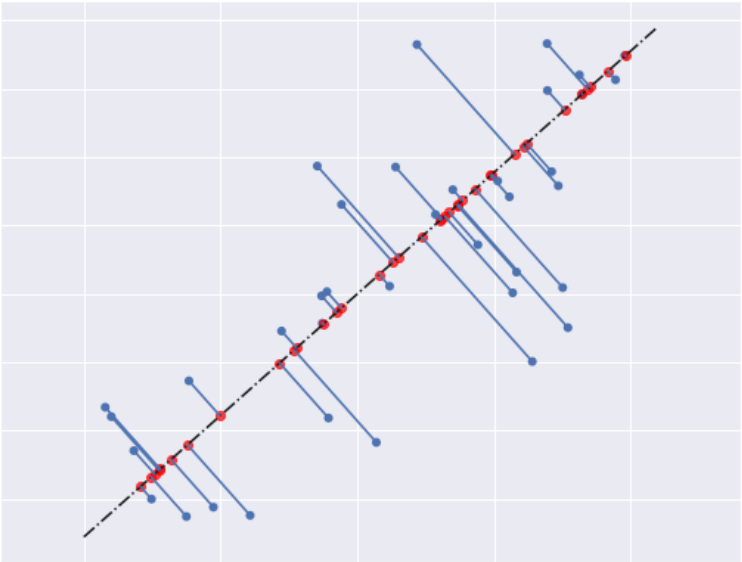


Figure 19 - Projection of 2-dimensional space to a 1-dimensional space

Above figure illustrated a projection of 2-dimensional space to a 1-dimensional space with blue dots represents original data vectors, the red dots represent their projections, Blue lines shows the reconstruction error, and the dashed black line is the optimal projection that received by solving the optimization.

2.5.1.2.1 Standardization

When use PCA technique on a dataset, it must be scaled. The results are sensitive to the relative scaling. Because if there are large difference between the ranges of initial variables, those variables will dominate over those with small ranges. Transforming the data to comparable scale can prevent from those problems.

2.5.1.2.2 Covariance Matrix Computation

In this process, it will help to understand how the variables of the input dataset are varying from the mean with respect to each other. Because occasionally variables are highly corelated

in such a way that they contain redundant information. Then by computing covariance matrix will be helpful to identify these correlations.

We can compute the covariance of two variables X, Y by using the following formula,

$$\mathrm{Cov}(X, Y) = \frac{1}{n-1}\sum_{i=1}^{n}(Xi - \bar{x})(Yi - \bar{y})$$

Equation 1 - Covariance of two variables X, Y

The covariance matrix is a $p \times p$ symmetric matrix which p indicate the number of dimensions. That has as entries the covariance associate with all possible pairs of the initial variables.

When consider about 3-dimensional dataset with 3 variables X, Y, Z the covariance matrix is,

$$\begin{bmatrix} \mathrm{Cov}(X,X) & \mathrm{Cov}(X,Y) & \mathrm{Cov}(X,Z) \\ \mathrm{Cov}(Y,X) & \mathrm{Cov}(Y,Y) & \mathrm{Cov}(Y,Z) \\ \mathrm{Cov}(Z,X) & \mathrm{Cov}(Z,Y) & \mathrm{Cov}(Z,Z) \end{bmatrix}$$

Equation 2 - 3-dimensional dataset with 3 variables X, Y, Z the covariance matrix

When the sign of the covariance gets positive, the two variables increase or decrease together which means correlated. If the sign of covariance gets negative that indicates one variable increase when the other decreases. Which states inversely correlated.

Compute the eigenvectors and eigenvalues of the covariance matrix.

Eigenvectors and eigenvalues are the linear algebra concepts which essential to calculate from the covariance matrix to determine the principal components of the data. The principal components represent the direction of data that explain a maximal amount of variance to capture most information of the data.

To calculate eigenvalue and eigenvectors from the covariance matrix, Let assume A be a square matrix, v is a vector and λ is a scalar that satisfies $Av = \lambda v$, then λ is called eigenvalue associated with eigen vector v of A.

The eigenvalues of A are roots of the characteristic equation.

$$\det(A - \lambda I) = 0$$

Equation 3 - Characteristic Equation

2.5.1.2.3 Feature Vector

As the previous steps, computing the eigenvectors and ordering them with their eigenvalues in descending order allows to find the principal components in order of significance. During this process, select these components to preserve by discarding those with less significance those have low eigenvalues and form a matrix with remaining components. This matrix states as feature vector.

2.5.1.2.4 Multivariate anomaly detection

When identifying anomalies with high dimensional data, data visualization would be difficult. Because of that multivariate statistics helps to reduce the complexity in this operation. Then consider abut collected datasets, they typically have a certain distribution like gaussian distribution. Then to identify anomalies in a more quantitively way, there need to calculate the probability distribution $\mathrm{p}(x)$ from the data points. Afterward receive new value x and compare $\mathrm{p}(x)$ with a threshold r. If $\mathrm{p}(\mathrm{x}) < \mathrm{r}$, it considered as an anomaly because normal values tend to have a large $\mathrm{p}(x)$ and anomalous examples tend to have small $\mathrm{p}(x)$ values.

In the condition monitoring application, this discussed anomaly detection approach would be ideal because it could give feedback about machine health condition by monitoring parameters in the equipment. Data would generate when then machine occurred sub optimal operation or failure.

2.5.1.3 The Mahalanobis distance

The Mahalanobis distance (MD) is the distance between two points in multivariate space. When consider about a situation of estimating the probability that a data point belongs to a distribution. The first step would be to find the centroid of the sample points. Spontaneously, the closer point in situation would be the center of mass. Also, there required to distinguish if the set is distributed over a large range or small range to decide whether a given distance from the center is significant or not. The simplistic approach to estimate the standard deviation of the distances of the sample points from the center of mass is using by plugging this into the normal distribution and that would be supportive to derive the probability of the data point belonging to the similar distribution.

The disadvantage of the above discussed approach was that the sample points are distributed about the center of mass in a spherical manner. The distribution to be non-spherical and for instance it would be ellipsoidal. Then expectation of the probability of the test points rely on the distance from the center of mass and the direction. In those directions where the ellipsoid

has a short axis the test point must be closer, while in those where the axis is long the test point can be further away from the center. When place this on a mathematical base, the ellipsoid would be the best representation of the set's probability distribution that can be estimated by calculating the covariance matrix of the samples. As previously said that Mahalanobis distance (MD) is the distance of the test point from the center of mass divided by the width of the ellipsoid in the direction of the test point.

When get the test point x from the center of mass y divided by the width of the ellipsoid in the direction of the test point is give the Mahalanobis distance $\mathrm{D}(\mathrm{x},\mathrm{y})$. The equation for this is shown in below. The covariance matrix of vector x stated as C and C^{-1} is its inverse matrix.

$$\mathrm{D}(\mathrm{x},\mathrm{y}) = \sqrt{(\mathrm{x}-\mathrm{y})^{\mathrm{T}}\mathrm{C}^{-1}(\mathrm{x}-\mathrm{y})}$$

In MD use to categorize test point in to N classes. In this bearing condition monitoring application, mainly classified "Normal" and "Anomaly" classes. In this case use training data, which based on operating condition in order to calculate the covariance matrix. In the test datasets, it would be supportive to compute MD to "Normal" class and classify the test point as an "Anomaly" if the distance is greater than the certain threshold value. A study based on ball bearing defect diagnostics Shuen-De Wu, et al., (2013) pointed that in ball bearing based system MD can use to measure the distinction between the characteristics and determines the effectiveness of identifying the component looseness in several points in a machine.

2.5.1.4 Artificial Neural Network

An artificial neural network (ANN) is a specific computing system designed to simulate the way the human brain analyzes and processes information. It is the foundation of artificial intelligence (AI) and solves problems that would prove impossible or difficult by human or statistical standards. ANNs have self-learning capabilities that enable them to produce better results as more data becomes available. Yasir Hassan Ali, (2018) in his publication suggests many approaches for machine fault diagnostics and in ANN discussion he states about a study of estimating oil film thickness through acoustic emission signal with ANN and that resulted 99.9% success rate in prediction of classification. By this example he tried to show the importance and capability of using ANN in real time applications.

2.5.1.4.1 Autoencoder networks

The discussed second approach is based on autoencoder neural networks. This constructed on same principles with that previously discussed statistical analysis. But there are many differences compared with PCA analysis.

Autoencoder is an unsupervised artificial neural network that learns how to efficiently compress and encode data then learns how to reconstruct the data back from the reduced encoded representation to a representation that is as close to the original input as possible. Autoencoders has capability with based on design to reduces data dimensions by learning how to ignore the noise in the data. Muhammad Sohaib and Jong-Myon Kim, (2018) suggested about a reliable fault diagnosis approach on autoencoder based deep NN to classify faults in varying speeds by considering complex envelope signals by demonstrating the ability of AE apply on the rotatory machine early fault prediction applications.

Autoencoder consist of four main components.

1) Encoder:
   Which is the model learning how to reduce the input dimensions and compress the input data into an encoded representation.
2) Bottleneck:
   Which is the layer that contains the compressed representation of the input data. This is the lowest possible dimensions of the input data.
3) Decoder:
   Which is the model learning how to reconstruct the data from the encoded representation to be as close to the original input as possible.
4) Reconstruction Loss:
   This is the method that measures measure how well the decoder is performing and how close the output is to the original input.

The training in this ANN involves back propagation to minimize the network's reconstruction loss.

Then architecturally auto encoders can vary between a simple feedforward network, LSTM network or convolutional neural network which depending on the application.

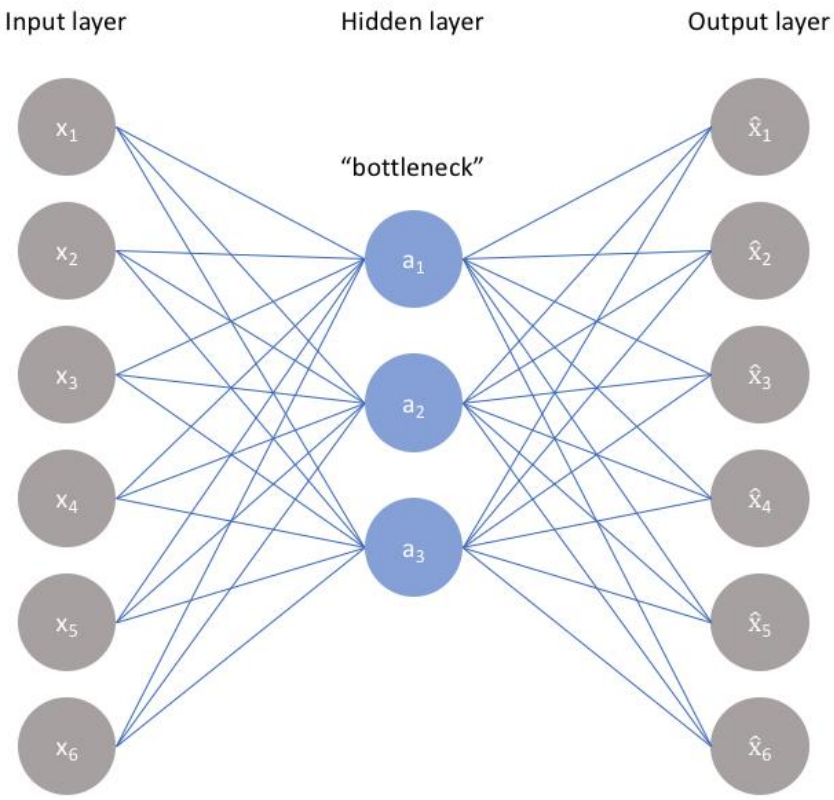


Figure 20 - Structure of a ANN

As visualized above, the NN could take an unlabeled dataset and frame it as a supervised learning problem tasked with outputting $\hat{x}$, a reconstruction of the original input x . This network can be trained by minimizing the reconstruction error, $\mathcal{L}(x, \hat{x})$, which measures the differences between our original input and the consequent reconstruction. The bottleneck is a key feature of the network design.

In this application there use an autoencoder with a feedforward, non-recurrent neural network which very similar to the numerous single layer perceptron which makes a multilayer perceptron (MLP) having an input layer, an output layer and one or more hidden layers connecting them but with the output layer having the same number of nodes as the input layer, and with the persistence of reconstructing its own inputs.

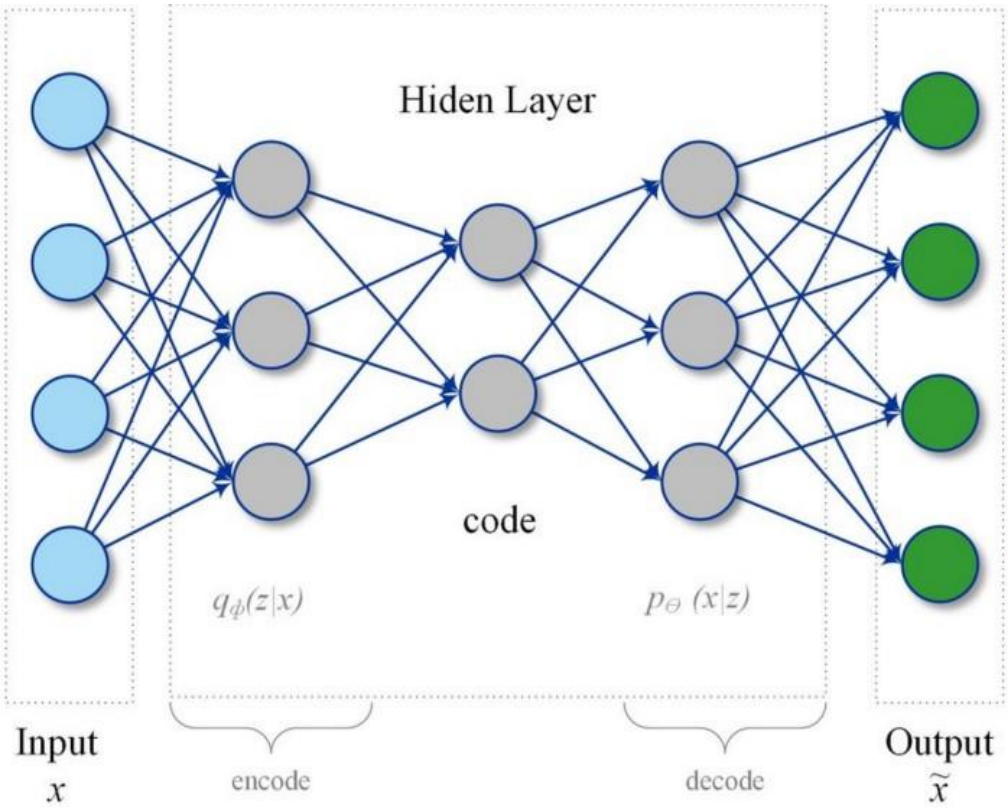


Figure 21 - Function of Hidden Layers

Anomaly detection in condition monitoring system based on auto encoder ANN system, first compress sensor readings to a lower dimensional representation that capture correlations and intersections between the various variables. This is a similar process compared with the PCA model but additionally autoencoder allow for non – linear interactions between the variables.

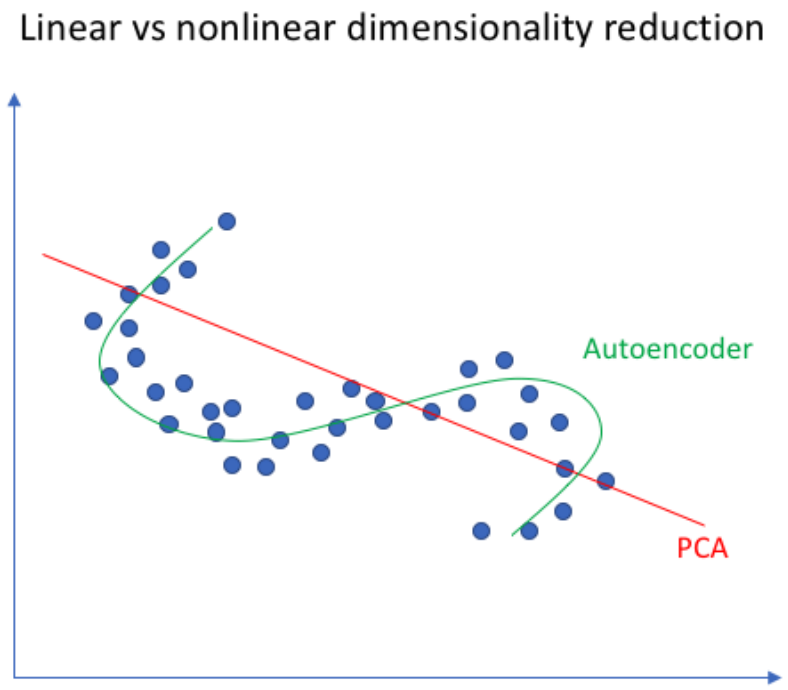


Figure 22 - Linear vs Non-Linear Dimension Reduction

In the developing condition monitoring system, autoencoder network will train data on "Normal" state with compressing and reconstructing the input variables. The during the dimensionality reduction the network learns the connections between the various variables, and it would be able to re-construct them back to the original variables at the output. The major objective is that as the monitored equipment's' health degrades, this should affect the interaction between the variables like pressure, vibration, temperature etc. As this occurs one will start to increased error in the network's re-construction of the input variables. The re-construction error, then there would be capable of getting an indication of the health by monitoring the equipment. If this error will increase that means the equipment degrades. This is similar the first approach of using the Mahalanobis distance but here uses the probability distribution of the reconstruction error to identify whether a data point is normal or anomalous.

2.5.1.5 Rolling element bearing components and failing frequencies.

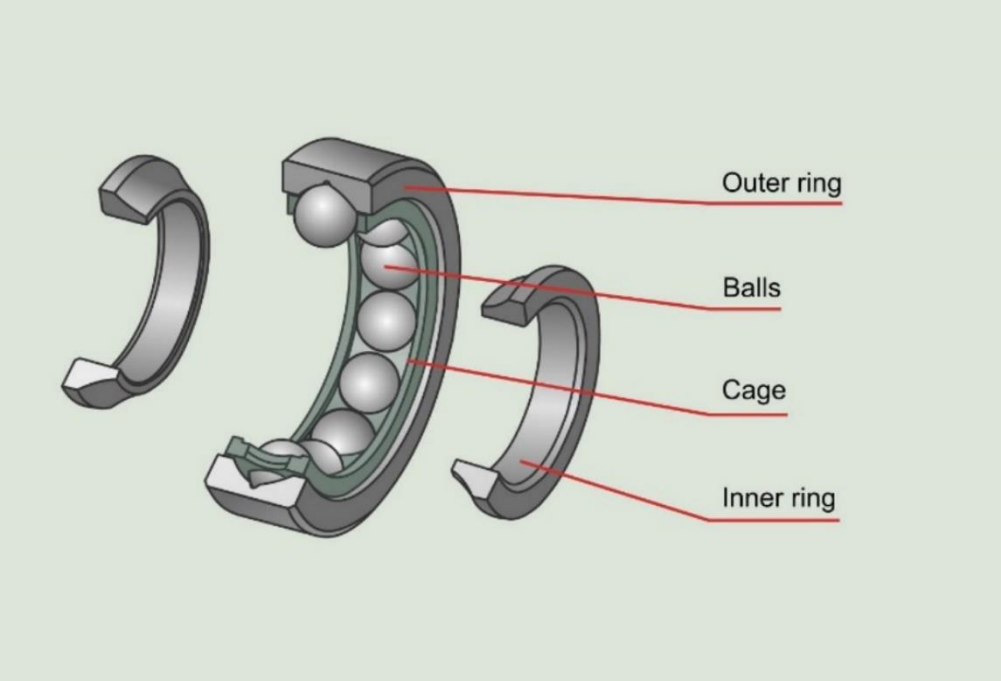


Figure 23 - Components of a Bearing

Rolling element bearings consist of several clearly differentiated components. They are,

- Outer race balls or rollers
- Inner race balls or rollers
- Cage

The deterioration of each of these elements will produce one or more characteristic failing frequencies in the frequency spectra and that will allow to identify easily and quickly the failure. That four possible bearing failing frequencies are,

(In the below equations, d is the ball diameter, D is the pitch diameter, f_r is the shaft speed, n is the number of rolling elements and $\emptyset$ is the bearing contact angle.)

- Ball pass frequency in outer race (BPFO)

$$\mathrm{BPFO} = \frac{\mathrm{n f_r}}{2}\left(1 - \frac{\mathrm{d}}{\mathrm{D}}\cos\emptyset\right)$$

- Ball pass frequency in inner race (BPFI)

$$\mathrm{BPFI} = \frac{\mathrm{n f_r}}{2}\left(1 + \frac{\mathrm{d}}{\mathrm{D}}\cos\emptyset\right)$$

- Fundamental train frequency (FTF)

$$\mathrm{FTF} = \frac{f_r}{2}\left(1 - \frac{d}{D}\cos\emptyset\right)$$

- Ball (roller) spin frequency (BSF)

$$\mathrm{BSF} = \frac{D}{2d}\left[1 - \left(\frac{d}{D}\cos\emptyset\right)^2\right]$$

As previously shown each bearing has its own geometric characteristics from which allows to determine its failing frequencies. These frequencies will appear in the spectral signatures when the bearing is deteriorated. Practically, for most bearings these failing frequencies will not be integer numbers, so that the dominant vibration, when there is a defect in any of the bearing components will be non-synchronous.

When one of the bearing components is defected, the frequency spectrum the fundamental frequency corresponding to the damaged element, always accompanied by harmonics. In those cases where the bearing physical parameters are not known, there are some empirical formulas that will allow to determine the failing frequencies of the bearing races and the cage in purpose of the number of rolling elements and the rotating speed.

Bearing failing frequencies that expressed above mainly depend on the contact angle, hence any slight variation of this will result in a variation of the ideal bearing failing frequencies, making it difficult to recognize these frequencies in the spectrum. The possible causes that can produce a variation of the contact angle can be very diverse. Some of them are misalignment, thermal growth, excessive bolt tightening, pitting, or peeling in the bearing races, etc. All this will affect these precalculated frequencies (ideal frequencies) that they do not precisely coincide with the frequencies that appear in the spectrum (actual frequencies), and therefore in some cases a certain level of margin must be allowed in the identification of the failing frequencies.

2.5.1.5 Dataset used.

The dataset that used in this project received from Kaggle datasets. The data was generated by the Center for Intelligent Maintenance Systems (IMS), University of Cincinnati with support from Rexnord Corporation in Milwaukee, Wisconsin.

When consider about the developed test rig for data acquisition, there were four bearings were installed on a shaft and the rotation speed was kept constant at 2000 RPM by an AC motor coupled to the shaft via rub belts. A radial load of 6000 lbs. is applied onto the shaft and bearing by a spring mechanism. As well all bearings are force lubricated. Rexnord ZA-2115 double row bearings were installed on the shaft. To get reading there were installed high sensitivity quartz ICP accelerometers on the bearing housing, in this case there were two accelerometers for each bearing in x- and y-axes for data set "1st_test", one accelerometer for each bearing for data sets "2nd_test" and "3rd_test". In this experiment All failures occurred after exceeding designed lifetime of the bearing which is more than 100 million revolutions.

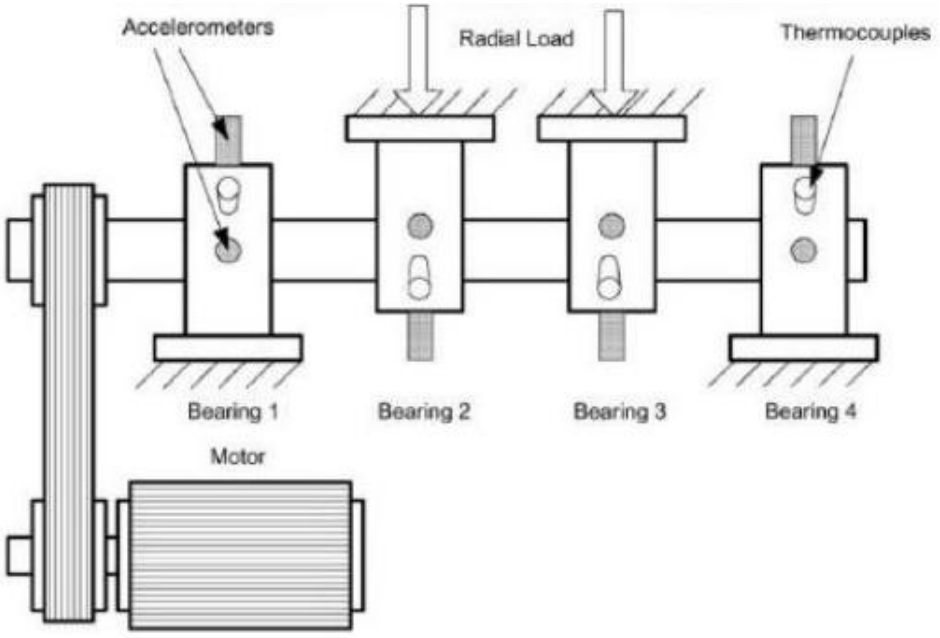


Figure 24 - Test Setup for Bearing

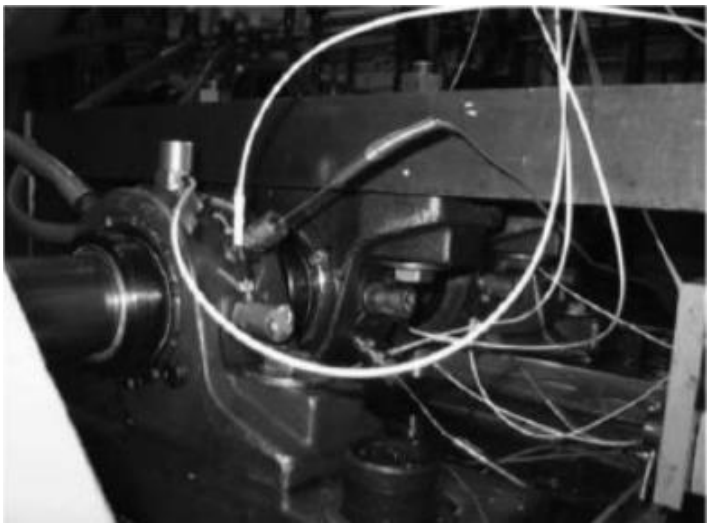


Figure 25 - Test Setup for bearings

Then consider about the structure of the received data, there are three datasets in this resource. Each data set describes a test to failure experiment. Each data set contains of individual files that are 1-second vibration signal snapshots recorded at specific intervals. Each file consists of 20,480 points with the sampling rate set at 20 kHz. The file name indicates date and time when the data was collected. Each record (row) in the data file is a data point. Data collection was facilitated by NI DAQ Card 6062E.The larger intervals of time stamps indicate resumption of the experiment in the next working day.

These are detailed information about the resource datasets.

1st_test:

Recording Duration:	October 22, 2003 12:06:24 to November 25, 2003 23:39:56
No. of Files:	2156
No. of Channels:	8
Channel Arrangement:	Bearing 1 – Ch 1&2, Bearing 2 – Ch 3&4, Bearing 3 – Ch 5&6, Bearing 4 – Ch 7&8
File Recording Interval:	Every 10 minutes (except the first 43 files were taken every 5 minutes)
File Format:	ASCII
Description:	At the end of the test-to-failure experiment, inner race defect occurred in bearing 3 and roller element defect in bearing 4.

2nd_test:

Recording Duration:	February 12, 2004 10:32:39 to February 19, 2004 06:22:39
No. of Files:	984
No. of Channels:	4
Channel Arrangement:	Bearing 1 – Ch 1, Bearing2 – Ch 2, Bearing3 – Ch3, Bearing 4 – Ch 4
File Recording Interval:	Every 10 minutes
File Format:	ASCII
Description:	At the end of the test-to-failure experiment, outer race failure occurred in bearing 1.

3rd_test:

Recording Duration:	March 4, 2004 09:27:46 to April 4, 2004 19:01:57
No. of Files:	4448
No. of Channels:	4
Channel Arrangement:	Bearing1 – Ch 1, Bearing2 – Ch 2, Bearing3 – Ch3, Bearing4 – Ch4
File Recording Interval:	Every 10 minutes
File Format:	ASCII
Description:	At the end of the test-to-failure experiment, outer race failure occurred in bearing 3.

The reason why these datasets are most suitable for this project are as stated about test rig, the proposed Bielomatik book making machine also worked on constant load and constant speed. There is also used ball bearings in selected monitoring points and most importantly the used accelerometers in the received datasets are almost identical in parameters to the purposed accelerometers in this condition monitoring project.

Accelerometer model used in datasets – PCB 353B33

Accelerometer model selected for project - CA-YD-187T02

Following are comparison of some parameters in both accelerometers.

Performance	**PCB 353B33**	**CA-YD-187T02**
Sensitivity	100 mV/g	100 mV/g
Measurement Range	±50 g pk	±50 g pk
Frequency Range (±10%)	0.7 to 6500 Hz	1 to 10,000 Hz
Frequency Range (±3dB)	0.35 to 12000 Hz	0.4 to 12,000 Hz
Resonant Frequency	≥22 kHz	≥30 kHz
Amplitude linearity	≤1 %	≤1%
Temperature Range (Operating)	-54 to +121 °C	-40 to +120 °C
Shock limit	±10000 g pk	3000 g pk
Operating voltage	18 to 30 VDC	18 to 28 VDC
Operating current	2 to 20 mA	2 to 10 mA
DC bias voltage	7.5 to 11.5 VDC	12±2 VDC
Sensing element	Quartz	Piezoelectric Ceramics
Case material	Titanium	304 Stainless steel
Weight	27 g	52 g
Output	10-32 Coaxial Jack	Top 5/8-24 2-pin Socket

2.5.1.6 Autoencoders and PCA comparison and selection

As previously disused, dimensionality reduction will help to decrease amount of training parameters, reduce overfitting and poor generalization and in reducing computational time. Autoencoders and PCA are common ways of reducing the dimensionality of the feature space. When doing comparison between these two concepts these are some points which find out during the process,

1. PCA is essentially a linear transformation while Autoencoders (AE) are capable of modelling complex nonlinear functions.
2. PCA is a simple linear transformation on the input space to directions of maximum variation and AE is a more sophisticated and complex technique that can model relatively complex relationships and non-linearities.
3. PCA is significantly fast as there exist algorithms that can fast calculate it while AE trains through Gradient descent and it is comparatively slower.

4. PCA projects data into dimensions that are orthogonal to each other resulting in very low or close to zero correlation in the projected data. And PCA gets the orthogonal subspace because eigenvectors come from eigen decomposition applied to a covariance matrix and in that condition, eigen decomposition results in orthogonal eigenvectors. AE transformed data doesn't assurance that because the way it's trained is merely to minimize the reconstruction loss.
5. PCA is faster and computationally cheaper than autoencoders.
6. PCA hyperparameter is 'k' that is number of orthogonal dimensions to project data into while for AE it is the architecture of the neural network.
7. AE with a single layer and linear activation has similar performance as PCA. AE with multiple layers and non-activation function known as Deep Autoencoder is disposed to overfitting and can be controlled by Regularization and careful designing.

The selection of technique depends on the properties of feature space itself. If the features have non-linear relationship with each other than autoencoder will be able to compress the information better into low dimensional latent space leveraging its capability to model complex non-linear functions.

2.5.1.7 Choosing Python for development

AI based projects are differ from most traditional software projects. The main difference in in the technology stack, and the requirement of deep research. To implement an AI based application there need to use a programming language that stable, flexible and has required tools available for mathematical and statistical computations. Python is an interpreted language. The Python interpreter runs a program by executing one statement at a time. Python is more favored for AI based projects because of its syntax structure simplicity and virtuous capability in data handling. According to Jean Francois Puget from IBM's machine learning department expressed his view that python is the most popular language for AL and ML based on the search data results of indeed.com.

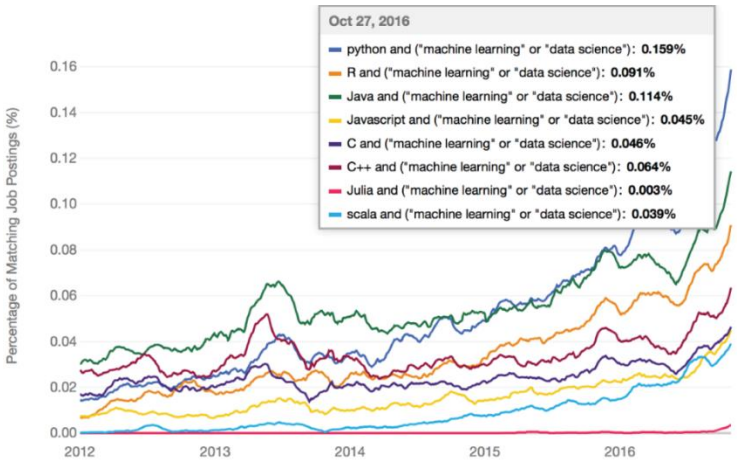


Figure 26 - Popularity of each programming language

Then consider from the view of the development side, python assistances developers to be productive and confident about developing programs and models. Also, there are trend of increasing python developing community than compare with other programming platforms and result of that having a great quantity of frameworks and they help to make programming easier and reduce the developing time.

According to conducted research on python language, there are many strong features that shows importance to opt python for AI and ML based projects.

1) It has an enormous library eco system.

   An enormous number of libraries existence is one of the main motives Python is the most popular programming language used for AI based projects. A library means a module, or a group of modules published by different sources like PyPi which include a pre-written part of code that allows users to reach some functionality or perform different activities. Python libraries provide base level objects, because of that developers do not have to develop them from the very beginning of every time when needed to use them.

   ML requires continuous data processing, and Python's libraries allow to access, handle, and transform relevant data. Followings are some of the most widespread libraries that can use for AI based model development,

   i) Scikit-learn –

      This open-source library widely used for conducting basic ML algorithms like clustering, linear and logistic regressions, regression, classification etc.

ii) Pandas –

This library helps for high-level data structures and analysis. It permits merging and filtering of data, also gathering it from other external sources like Excel, CSV for instance.

iii) Keras –

This used for deep learning applications. Specially this permits fast calculations ant prototyping and it consume the GPU in addition to the CPU for the computation process.

iv) TensorFlow –

One of the most well-known AI based comprehensive and flexible library with capability of developing up to deep learning models with using complex computation models.

v) Matplotlib –

A comprehensive library for creating static, animated, and interactive visualizations in Python. It is a cross-platform library for making 2D plots from data in arrays. It offers an object-oriented API that supports in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. Also, it can be used in Python and IPython shells, Jupyter notebook and web application servers.

2) Has a low entry barrier.

Developing an AI based application means dealing with a lot of data that required to process in the most convenient and effective way. This low entry barrier tolerates more data scientists to quickly pick-up Python and start using it for AI development without wasting too much energy to learning the language.

Python programming language based on the day-to-day English language, and that results the learning process become easier. Its simple syntax allows to easily work with complex systems, with confirming clear relations between the system elements. And there are many documentations related with libraries and programming that can access freely. Also, the python's community always be active to help with programming difficulties.

3) Flexibility

Python is a high-level programming language which has many flexible features,

- It gives an option to select either to use Object-oriented programming (OOP) or scripting.
- There is not essential to recompile the source code, developers can implement any changes and quickly see the results.
- Programmers allows easily combine Python and other languages to reach goals with their applications.

Furthermore, this flexibility permits developers to select the programming styles which they are fully comfortable with or combine these styles to resolve different types of problems in the most efficient way.

i) The imperative style contains of instructions that describe how a computer should perform these commands. With this style, it can express the order of computations which happen as a variation of the program state.

ii) The functional style is also known as declarative because it states that what operations should be performed. It does not consider about the program state, then related to the imperative style, it declares statements in the form of mathematical equations.

iii) The object-oriented style is mainly based on two concepts. They are class and object. This style is not fully supported by Python language, by way of it can't fully perform encapsulation, nevertheless developers can still use this style to a finite degree.

iv) The procedural style is the most common among beginners, because it proceeds tasks in a step-by-step arrangement. It's frequently used for sequencing, iteration, modularization, and selection.

This flexibility feature decreases the possibility of errors, because of that programmers have a chance to take the condition under control and work in a comfortable environment during program development.

4) Platform independence

Python is not only comfortable to use and easy to study but also very adaptable. Its mean is that Python for ML development can execute on any platform including

Windows, MacOS, Linux, Unix, and twenty-one further. To transfer the process from one platform to another, developers need to implement several small-scale variations and change some lines of code to create an executable form of code for the preferred platform. Developers can use packages like PyInstaller to prepare their code for running on different platforms. This helps to save time and cost for test on many platforms and make whole process simpler and user oriented.

5) Readability

   Python is easy to read and result of hat each Python developer can understand the program of their peers and modify, copy, or share it. There is no misunderstanding, errors, or conflicting paradigms, and this leads to more a more effective exchange of algorithms, ideas, and tools between AI developers. There are exist tools alike IPython which is a command shell for interactive computing that offers many features including testing, debugging, tab-comparison etc. and help to the process.

6) Good visualization options

   Python offers a variety of libraries, and some of them are good visualization tools. Nevertheless, for AI based applications developers, it's important to use that visualizations in their projects. Because it's vital to be able to represent data in a human-readable format.

   Libraries like Matplotlib allow developers to build charts, histograms, and plots for better data comprehension, effective presentation, and visualization. In different application programming interfaces correspondingly simplify the visualization process and helps that it easier to create clear documentations.

7) Community support

   It's always very helpful when there's strong community support built around the programming language. Python is an open-source language which means that there's a significant number of resources open for programmers starting from beginner to professional level.

   There are lot of Python documentation is available online as well as in Python communities and forums, where programmers and ML developers discuss their errors, solved problems, and help each other to overcome their difficulties.

8) Increasing popularity

Python language is becoming more popular among data scientists and developers. According to Stack Overflow, the popularity of Python is predicted to grow until 2020, This indicates it is easier to search for developers and replace team players when there needed. Also, the cost of their work mostly not going to be high compared with using a less popular programming language.

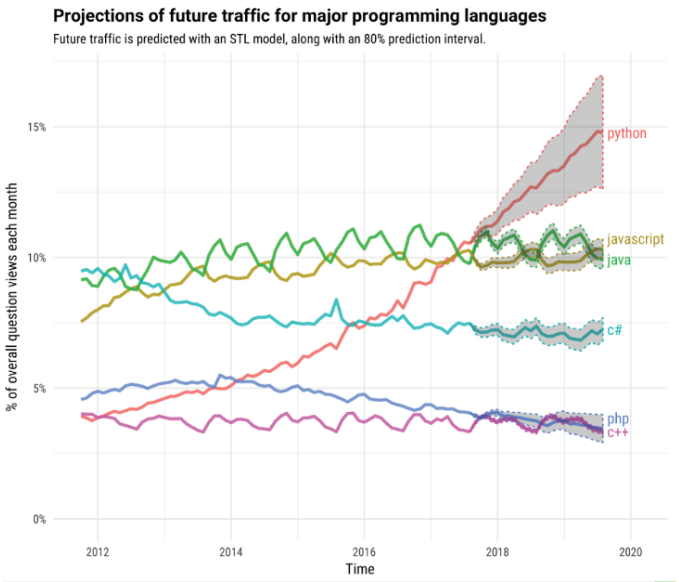


Figure 27 - Projection of future traffic for major programming languages

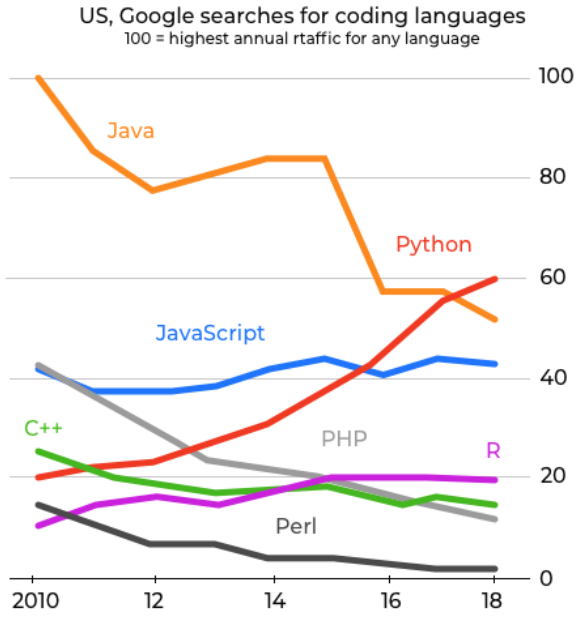


Figure 28 - US google search records of programming languages.

2.5 Selecting a PLC

A Programmable Logic Controller, or PLC is an industrial digital computer ruggedized and adapted for industrial control, manufacturing processing and automation applications.

Industrial applications are different from domestic engineering applications. Due to high current and voltages used in industrial manufacturing processes, manufacturing plants are always subjected to electromagnetic fields which can damage delicate electrical components over time if they were not properly secured. Hence, normal micro controllers like Arduino cannot be used in industrial applications. Electronics components of a PLC are well secured using circuit protection methods, so PLCs are used in almost every manufacturing facilities.

Leading PLC manufacturers:

- Allen Bradley
- ABB
- Siemens
- Omron PLC
- Mitsubishi PLC
- Hitachi PLC
- Delta PLC
- General Electric (GE) PLC
- Honeywell PLC

In Atlas Axillia, generally Haiwell PLCs are used for industrial automation applications. Since it is a new PLC brand, it is needed to thoroughly consider the features and specifications of the PLC to check whether Haiwell PLCs would suffice the requirements of the predictive maintenance system.

Most important considerations in selecting a PLC are:

- System requirements
- Environment requirements
- Inputs and Outputs (I/O s)
- Speed
- Type of communication protocol
- Programming

2.6.1 System Requirements

The tasks that the system is required to carry out will determine the kind of PLC needed. Another deciding factor is whether the system is to be built from scratch or if existing products are already installed. These factors are important because it is needed to match a PLC's functionality with the tasks at hand and with any existing products that are installed.

2.6.1.1 Environmental Requirement

Although most PLCs are ruggedized for tough physical environments, issues such as dust, vibration, temperature, or specific facility codes may be affected. Since a dust proof aluminum enclosure will be used to protect the electrical components of the control panel, environmental requirements can be negligible.

If the temperature was considered, the typical operating range for PLCs is 0-55 degrees Celsius, but if ambient temperature of the manufacturing facility falls outside of that range or has other atypical environmental requirements, it is needed to do research on PLC models which can operate in such ambient temperatures or find a solution for a proper cooling system for the predictive maintenance system.

2.6.1.2 Inputs and Outputs

It is important to consider the number of the inputs and outputs of a PLC and whether it is compatible with external expansion modules. Also, the type of the inputs and outputs should be considered.

There are several types of Haiwell PLCs according to the input and outputs modules, some contains fixed number of inputs and outputs and does not compatible with external expansion modules (Haiwell C-series – Economic PLC). Some Haiwell PLCs are compatible with several analog and digital input and output expansion modules (Haiwell T series and H series PLCs).

2.6.1.3 Speed

The processing speed of the PLC must be fast enough to handle multiple I/O's and the types of data collection required for the application. Another consideration is the scan time, which is the amount of time it takes for the CPU to perform one cycle of gathering inputs, running the PLC program, and updating the outputs.

The Haiwell PLCs claim to have an instruction execution speed of a 0.05μs per basic instruction which is fast enough for this system.

2.6.1.4 Types of Communication protocols

The type of communication protocols your system will use is another factor when selecting a PLC. Sometimes PLCs are equipped with onboard communication ports, but others require additional communication modules. Other options are to communicate remotely via Ethernet or to build upon multiple types of communications as your system requires.

It is built-in Modbus RTU / ASCII protocol, freedom communication protocol and Haiwell bus high-speed communication protocol of Haiwell company (each port supports the above protocols). One communication port can take different baud rates, different data formats, different communication protocols at the same time.

2.6.1.5 Programming

The Haiwell Plc can be programmed as a general PLC. It can program by the following program languages,

- LD (ladder)
- FBD (Function block)
- IL (instruction list)

Considering all the parameters, it was decided to use a Haiwell PLC since it is possible to take the almost the same output level as a high-end PLC for this application.

CHAPTER 3: METHODOLOGY

3.1 Identifying a background to implement the predictive maintenance system.

Among the other machines the in the printing section Bielomatik machine is one of the most important and critical machines. So, we decided to implement our program to the machine considering its age and working condition. (manufactured in 1972)

3.2 Machine Condition Analysis

The machine we intended to analyze is one of the oldest machines in the company. It was manufactured in Germany in year 1972, so the drawings and the component details were not much clear. We had to develop a drawing with details obtained from the technician and the maintenance management.

We identified the components but all most all the bearing components were changed. In the initial stages our machine learning model was based on the bearing details and dimensions. Since the bearing details are much different than the drawings, we altered our model as a universal analyzer.

Bielomatik 2, the exercise book printing machine in Atlas Axillia (Pvt) Ltd produced exercise books in wider variety to satisfy the demand of the whole nation. Being the leader in stationary manufacturing industry, the company much rely on the Bielomatik machine, so the operational condition of the machine is crucial to the operation schedule. Maintenance team working very thoroughly for keeping the machine online according to the production schedule.

3.3 Bielomatik 2

The fully automatic P 35 is a highly sophisticated solution back in the days for the manufacturing and printing of double side printed and double wired (stapler) exercise books. It is optimally used for large runs, from manual feeding paper reel imported from India to the finished exercise book. The printing range is much broader which includes single ruled, double ruled, square rules while the paper size varies from A5-A4.

Basic machine stations consist of reel stand where the holding and aligning mechanism of the paper role takes place, 2 printing towers with changeable printing polymer roles, cross cutting section to cut the desired length, flexible cover feeding, punching and wire binding. Other than that, the company has modified the machine, so the cover mixing process is also automated.

The tension regulating mechanism is much more mechanism than a fully automated one but serves the purpose.

We divided the machine into many specific parts for critical analysis of the potential failure points.

- Unwinding Unit
- Brake Unit
- Printing Tower
- Cross Cutter
- Over lapping
- Sheet Counting Unit
- Cover feeder.
- Stitching Unit
- Folding unit
- Spine pressing unit.
- Long knife section
- Separating knives section
- Delivery conveyor

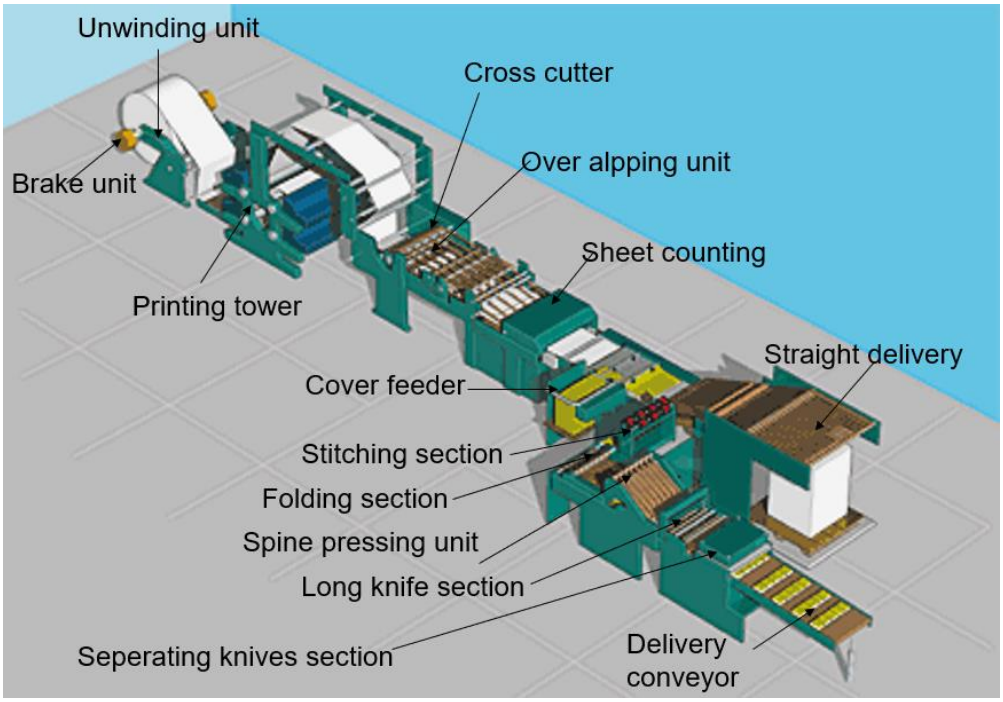


Figure 29 -Bielomatik 2 Printing System

3.3.1 Critical Area identification

According to the past maintenance data and component replacement history, we were able to conclude that the Printing tower and cross cutter breakdowns costed the most for the production schedule. Since the crosscutter bearing is specially made for the machine itself. The machine contains many mechanisms with,

- Mechanical
- Electrical
- Electronic
- Hydraulic
- Pneumatic actuators.

Mechanical components are the oldest and most critical components that should face the attention.

Figure 30 - Printing Tower

Mechanical Components include,

- Gear Wheels
- Chain & Sprockets
- Bearings
- Pulleys & Belts
- Levers Shafts
- Springs, Nut & Bolts, Washers etc...

Bearing took our attention because the past data convinced that the bearing conditions will result in the most critical failures with much higher down time.

3.3.1.1 Printing Tower Analysis

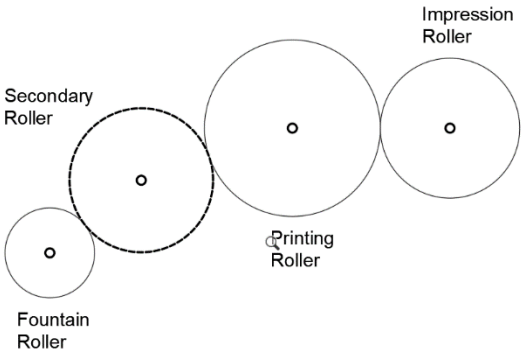


Figure 31 - Printing Mechanism

This printing tower consists of 2 printing stations with 2 Printing polymer roles. These rollers contain the required printing die.

Figure 32 - Printing Polymer

3.3.1.2 Cross Cutter Section Analysis

Figure 33 - Cross Cutter Section

Cross cutter station will cut the papers into the desired lengths such that the model will be CR or A5. The bearing shown in the above picture is the most crucial component of the whole mechanism because the bearing is custom made for the cutter itself in India. So, failing in such components could stop the operation for days.

In summary our project scope will describe the analysis and 24/7 monitoring and prediction of the condition of the following two modules.

I. Printing Tower – 2 Bearings
II. Cross Cutter - Main Bearing

3.4 PCA based model development.

In this implementation we selected "2nd_test" data set from the data resource for the computation because of it has one reading from each bearing and in the monitored system and it contains 984 data files with recordings over 20480 datapoints in each file. According to datasets documentation, at the end of the experiment, outer race failure occurred in bearing 1.

In the model development there assumed that gear degradation occurs gradually over time, there were used "2nd_test" dataset and its used one datapoint of every 10 minutes in the following analysis. Each 10-minute datapoint is combined by using the mean absolute value of the vibration recordings over the 20480 datapoints in each file. Then merged together that data in a single data frame. After that transformed the index to datetime format and stored data in .csv format.

	Bearing 1	Bearing 2	Bearing 3	Bearing 4
2004-02-12 10:32:39	0.058333	0.071832	0.083242	0.043067
2004-02-12 10:42:39	0.058995	0.074006	0.084435	0.044541
2004-02-12 10:52:39	0.060236	0.074227	0.083926	0.044443
2004-02-12 11:02:39	0.061455	0.073844	0.084457	0.045081
2004-02-12 11:12:39	0.061361	0.075609	0.082837	0.045118

Figure 34 - Vibration data collection

Then visualize the processed data to inspection and get a graphical feedback of the selected data.

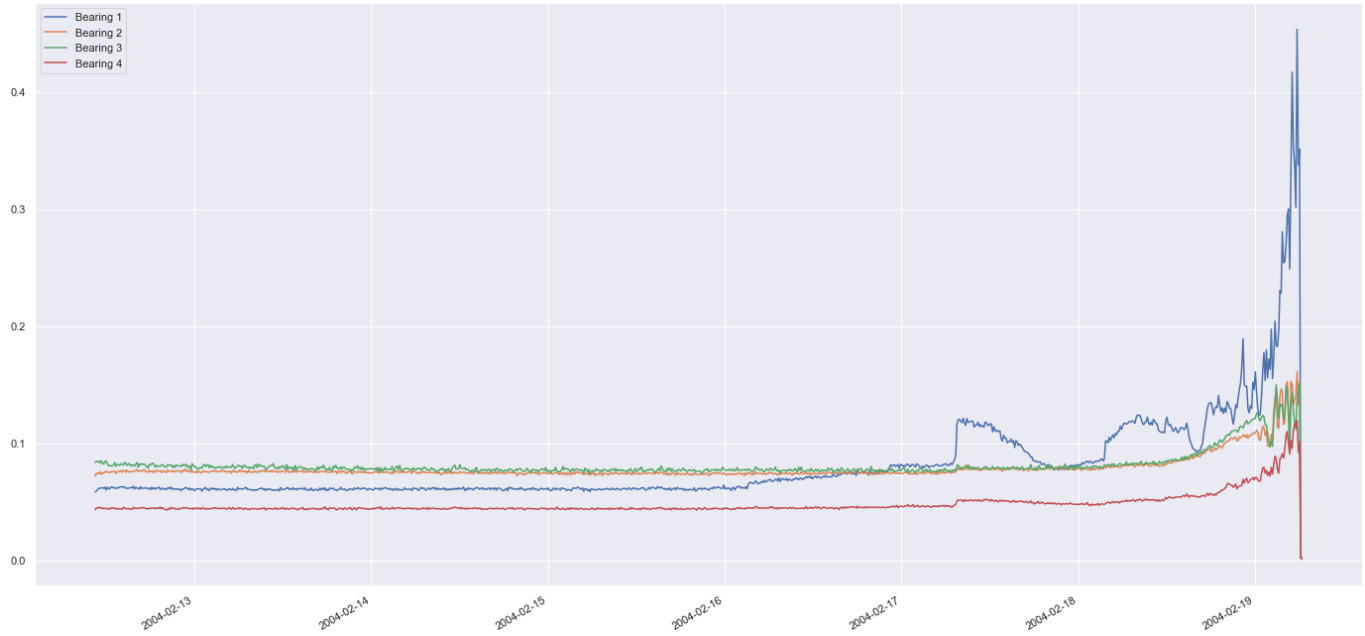


Figure 35 - 2nd test merged time domain data visualization

Then defined train and test data before setting up the model. In this step, separate train and test data based on datetime index and perform a simple split where the train on the first part of selected dataset which represents normal operation condition (healthy) and test the remaining part of the dataset for occurring failure on system.

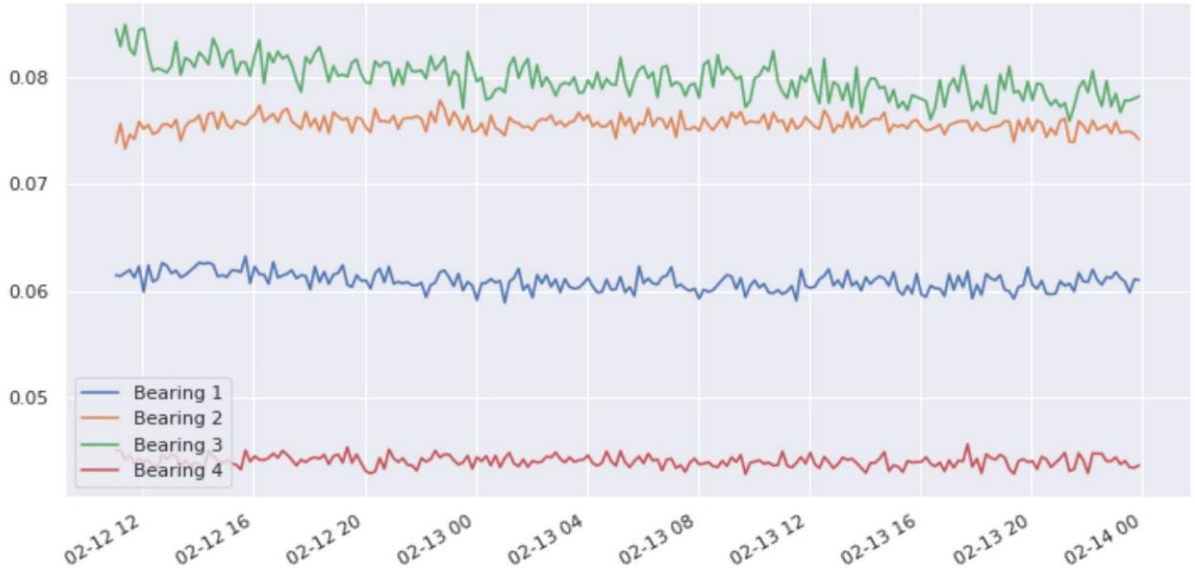


Figure 36 - Training dataset visualization

Then used preprocessing tools from Scikit-learn to scale the input variables of the model using by “MinMaxScaler” to perform simple rescales in the data to be in range of 0 to 1.

After that process, developed a PCA type model for anomaly detection using by “sklearn. decomposition module in PCA” in order to achieve dimensionality reduction with high dimensional sensor readings. After the PCA parameter configuration, developed python script along with sci-kit learn library for define functions related with PCA. In this stage performed calculating the covariance matrix, calculating the MD distance, detect MD outliers and calculation of the threshold value. Then develop python function for check if matrix was positive definite for future development. Then insert pre-defined train and test data in to the developed PCA model. After that calculate the covariance matrix and its inverse based on data in the training dataset. Also calculated the mean value for the input variables in the training set, as this is used later to calculate the MD to datapoints in the test set. Then using the covariance matrix and its inverse, calculated the MD for the training data defining “normal conditions”, and find the threshold value to flag datapoints as an anomaly. Then calculate the MD for the datapoints in the test set and compare that with the anomaly threshold.

The square of the MD to the centroid of the distribution followed a x^2distribution if the assumption of normal distributed input variables is satisfied. This is also the assumption behind the above calculation of the “threshold value” for flagging an anomaly. As this assumption is not essentially satisfied in our case, it is beneficial to visualize the distribution of the MD to set a good threshold value for flagging anomalies. Then, visualizing the square of the MD, which should then ideally follow a x^2 distribution and MD.

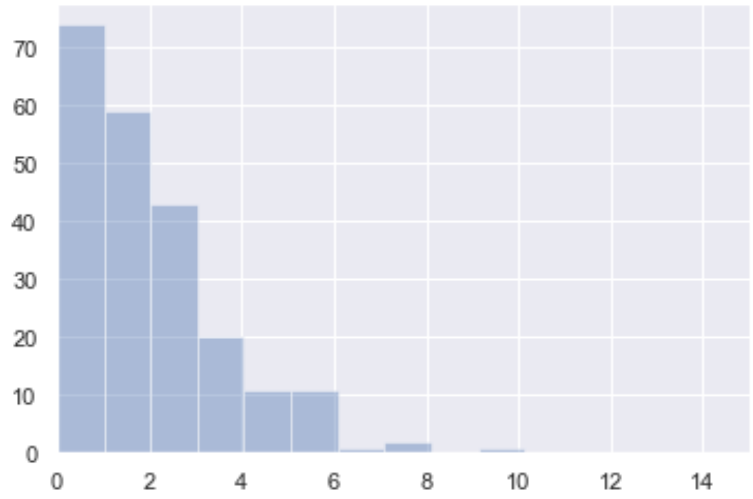


Figure 37 - Square of the MD visualization

By using above graphically plotted distributions, calculated threshold value for flag anomaly. After that use this computed MD and threshold value for flag anomaly in Boolean format. In this case Anomaly value get “TRUE” when MD get above the threshold value. After that merge data and save it in .csv format.

Lastly, plot graph with calculated anomaly in MD and check when it crosses the calculated anomaly threshold. In this following graph the y-axis noted in logarithmic scale.

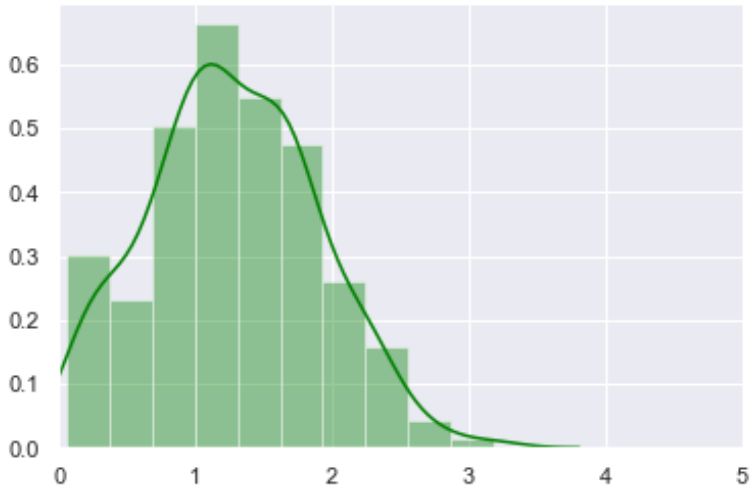


Figure 38 - Visualization of MD

Power BI and anything open in desktop

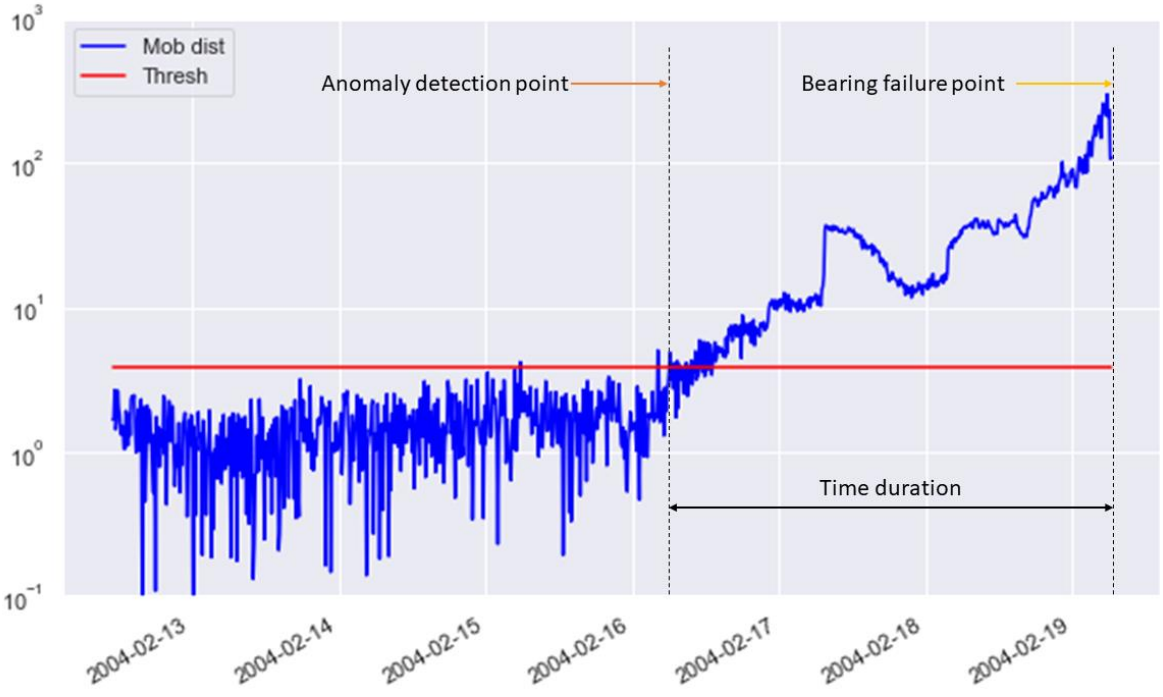


Figure 39 - Verify the test data on PCA model.

From the above generated figure, at this condition the model able to detect anomaly approximately 3 days before the actual bearing failure.

3.4.1. Autoencoder based model development.

In this model simply use autoencoder NN to compress the received sensor readings data to low dimensional representation to detect correlations and interactions between the various variables. This approach basically like PCA model but allows for non-linearities among the input variables.

First defined the AE NN using a 3-layer NN which contain 10 nodes in first layer, 2 nodes in middle layer and third layer with 10 nodes. For loss function, used mean square error and trained the model using the “Adam” optimizer.

To fit the defined model with data. In this case used 5% of the training data for validation after every epoch.

```
Epoch 1/100
 1/21 [>.............................] - ETA: 0s - loss: 0.1732WARNING:tensorflow:Callbacks method `on_train_batch_end` is sl
ow compared to the batch time (batch time: 0.0000s vs `on_train_batch_end` time: 0.0010s). Check your callbacks.
21/21 [==============================] - 0s 9ms/step - loss: 0.1606 - val_loss: 0.0879
Epoch 2/100
21/21 [==============================] - 0s 2ms/step - loss: 0.1116 - val_loss: 0.0590
Epoch 3/100
21/21 [==============================] - 0s 2ms/step - loss: 0.0750 - val_loss: 0.0365
Epoch 4/100
21/21 [==============================] - 0s 2ms/step - loss: 0.0467 - val_loss: 0.0248
Epoch 5/100
21/21 [==============================] - 0s 2ms/step - loss: 0.0289 - val_loss: 0.0212
Epoch 6/100
21/21 [==============================] - 0s 2ms/step - loss: 0.0209 - val_loss: 0.0221
Epoch 7/100
21/21 [==============================] - 0s 2ms/step - loss: 0.0186 - val_loss: 0.0233
Epoch 8/100
21/21 [==============================] - 0s 2ms/step - loss: 0.0179 - val_loss: 0.0222
Epoch 9/100
```

Figure 40 - NN Model Training

Then developed a python sketch for visualize training loss and validation loss.

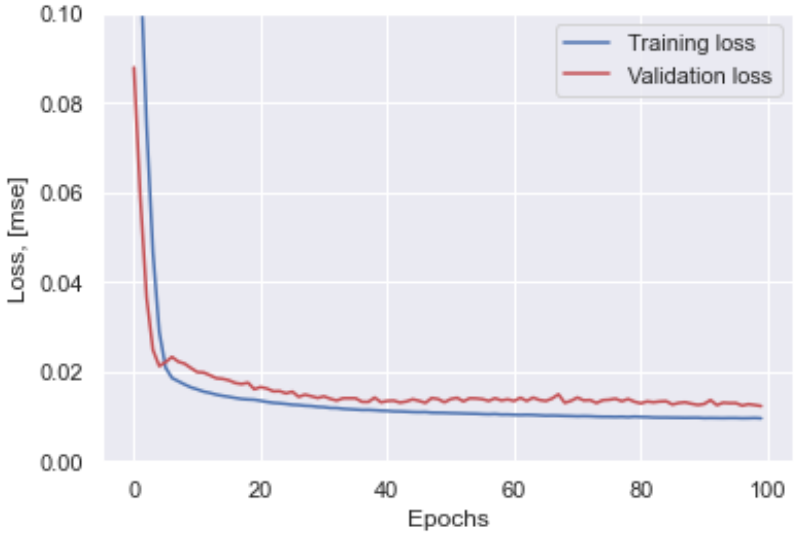


Figure 41 - Train and validation Loss Visualization

After that plotted the distribution of the calculated loss in the training dataset in order to identify a suitable threshold value for in this process, considered about setting threshold value above the noise level to flag any anomaly statistically significantly above the noise backgrounds in the system.

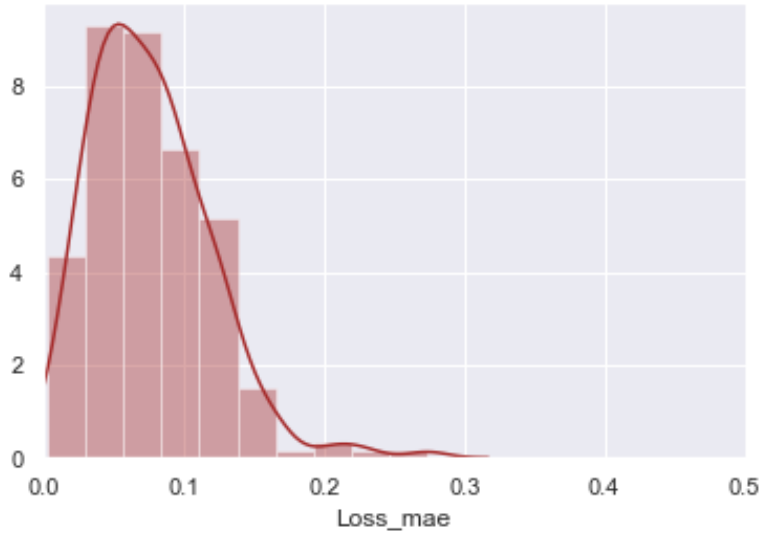


Figure 42 - Loss distribution of Training data

According to the above loss distribution figure, selected threshold value as 0.3 an implemented the guideline in the program for calculate loss in test data and logically select the Boolean value of anomaly flag compared with assigned threshold value.

	Loss_mae	Threshold	Anomaly
2004-02-13 23:52:39	0.135230	0.3	False
2004-02-14 00:02:39	0.103883	0.3	False
2004-02-14 00:12:39	0.031544	0.3	False
2004-02-14 00:22:39	0.072124	0.3	False
2004-02-14 00:32:39	0.120766	0.3	False

Figure 43 - Anomaly flagging process.

Finally merged all train and test data related anomaly detection information and visualize the model output graphically.

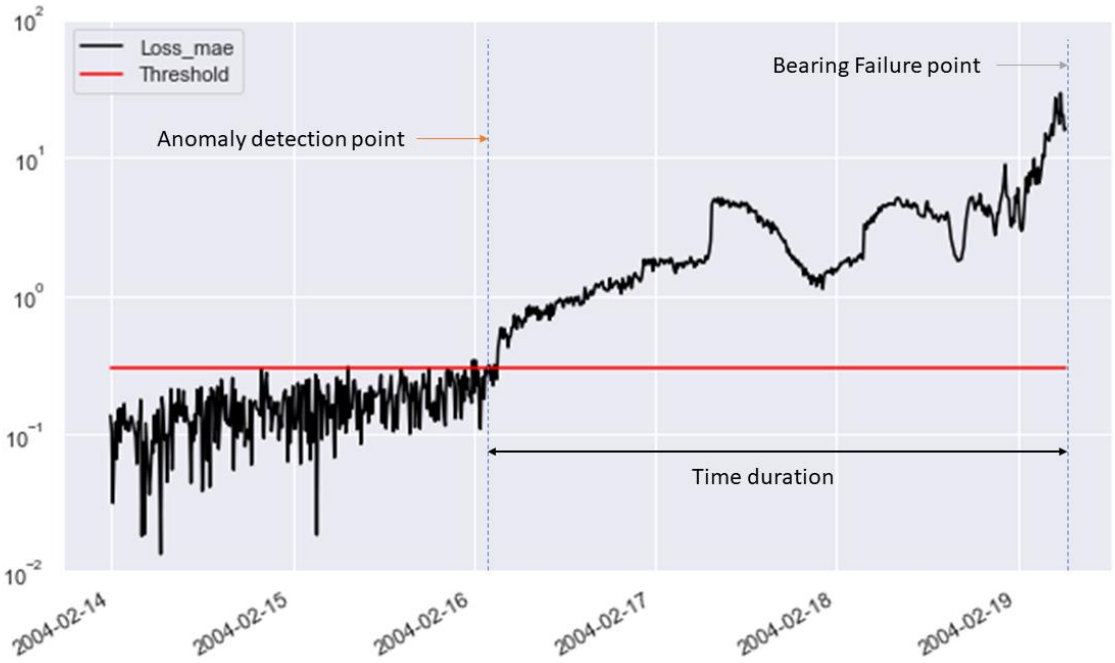


Figure 44 - AE based model outcome visualization

As shown in above outcome of the autoencoder based model there also flagged anomaly approximately 3 days before failure. Also, in this application the results of AE model and PCA model provided almost similar outcomes and consider with previously discussed advantages and disadvantages of each technique, the PCA based model seems to be the most efficient, cost friendly approach for the Bielomatik machine condition monitoring system.

3.4.2. Flowcharts of PCA

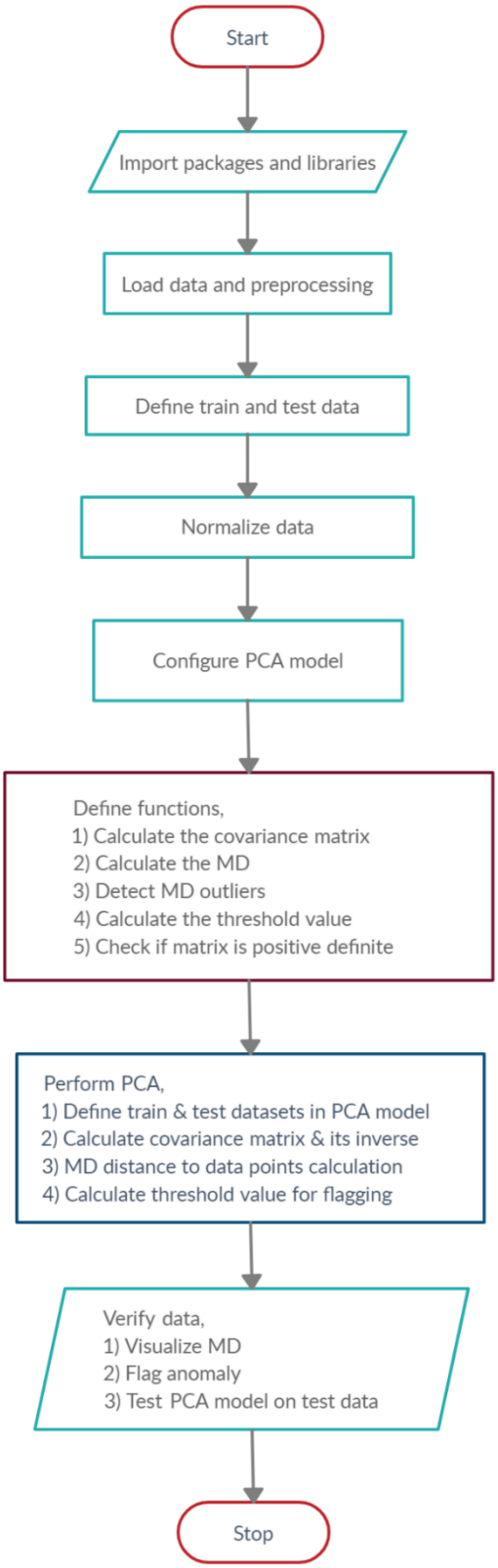


Figure 45 - Flowcharts of PCA

3.4.3. Flowcharts of AE

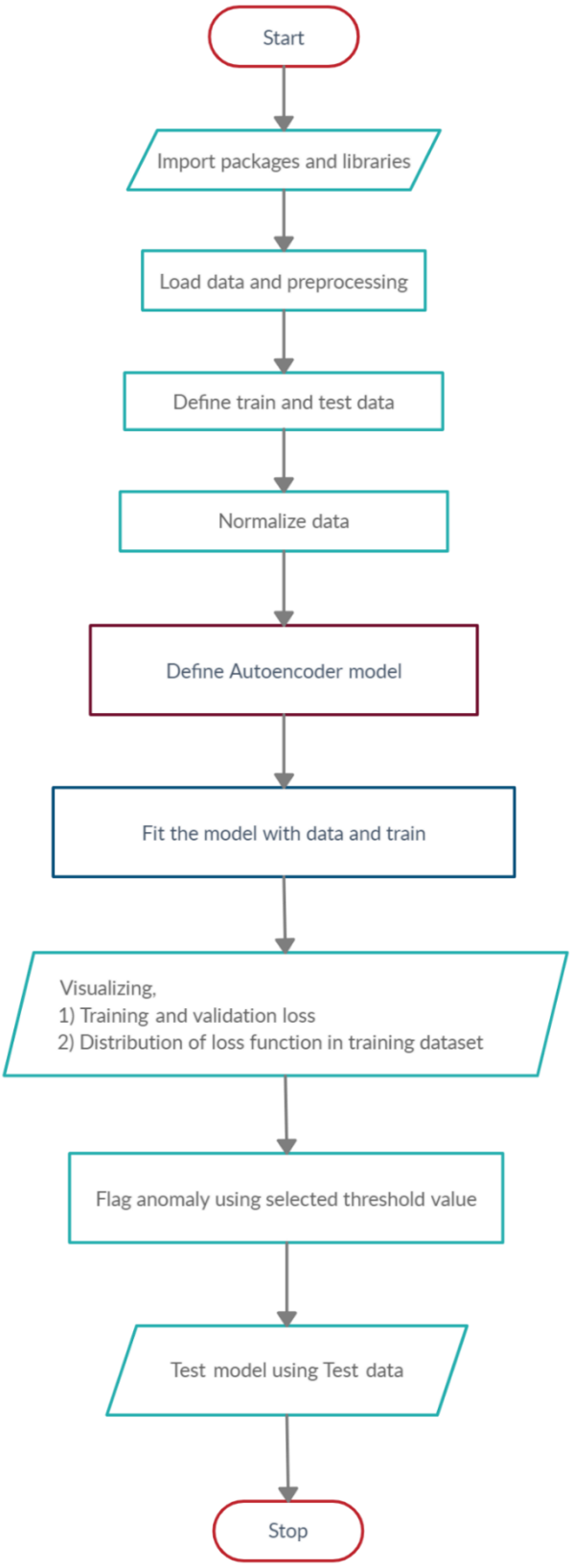


Figure 46 - Flowcharts of AE

3.5 Signal Conditioner

Signal conditioning is a process of data acquisition, to perform this process the signal conditioner is used as the instrument. The process of signal conditioning is converting one type of electrical signal into another type of electrical signal. The purpose of signal conditioning is to amplify a faint electrical signal and convert this signal into another signal form which is an easy to read and compatible form for data acquisition. This process helps to provide a precise measurement of this faint electrical signal which are essential for accurate data acquisition.

3.5.1. Functions of a Signal Conditioner

1. Signal Conversion

   This is the main function of a signal conditioner. Signal conversion means detect the faint signal and convert it into a higher level of readable electrical signal.

2. Linearization

   When the signal produced by the relevant sensor does not have a straight-line relationship with the physical measurement, this method is used by interpreting the signal from the software. This function allows to reach higher accuracy.

3. Amplifying

   Amplification is the process of increasing the signal for processing or digitization. This is done in two ways.

   i. Increase the resolution of the input signal.
   ii. Increase the signal-to-nose ratio.

4. Filtering

   Filtering is another important function. It filters the signal frequency function and include only the valid data and block noises if there were any. By the digital algorithm, this filter can be made from either passive or active components.

3.5.2 IEPE Signal Conditioner YE3826A

Figure 47 - IEPE Signal Conditioner YE3826A

YE3826A is a IEPE Signal Conditioner. It is used here to amplify the signals from 6 vibration sensors. It can be powered by either AC or DC. There are 12 channels, meaning 12 sensors could be connected at the same time. Input and output connections are done through BNC connectors. This is a heavy rugged industrial grade signal conditioner.

Features

- 12 channel signal conditioner
- Selectable supply current
- Ten times of signal amplification
- Frequency bandwidth is wide
- Indicates the operating status
- BNC output or 25-way D-connector output

Performance

- Output voltage is to -10V to 10V peak
- Output current is 5mA
- Accuracy is $\leq \pm 1.0\%$
- Upper cut-off frequency is 100kHz
- Lower cut-off frequency is 0.3Hz

Environmental Characteristics

- Operating temperature is 0^0C to 40^0C
- Storage temperature is from -55^0C to 85^0C
- Maximum humidity is 95%RH

3.6. Viber X2 Pro

3.6.1. Introduction

Viber X2 Pro is a handheld vibration monitoring instrument which is specially designed for maintenance and repair personnel as an instrument for basic condition monitoring checks. It is battery powered and easy to use and reliable for such condition analyses. The reliability of the device is obtained through taking accurate measurements for four selectable frequency ranges. This device only consumes less power so the maintenance and repair personnel can use this device for their entire maintenance program from a single charge. Batteries are rechargeable and a battery charger is provided with the device.

Figure 48 - Viber X2 Pro

A high-performance accelerometer comes with the device and it is used to take real-time measurement of the total vibration. This measurement is processed withing the device and it outputs a Bearing Condition measurement and both the vibration level, and the bearing condition are displayed simultaneously in the inbuilt LCD display. The bearing condition is given in the frequency range of 0,5 to 16 kHz and there is a large dynamic range for the vibration signal of the device (up to 50g).

Stability of the reading is indicated from a bar indicator. And vibration danger will be alarmed by red and yellow LED's.

3.6.2. Vibration Analysis

Recommended vibration levels

- **0 – 3 mm/s**
    - None or very small bearing wear.

    - Low noise level

- **3 – 7 mm/s**
    - Noticeable bearing wear
    - Increased noise level

- **7 – 11 mm/s**
    - Large vibrations
    - High noise level

- **≥ 11 mm/s**
    - Very large vibrations
    - High noise level

Necessary Actions

In the 0 – 3 mm/s vibration level, it is only considered as very small vibration level, hence it is not needed to take any actions at this level. In the level of 3 – 7 mm/s vibration level, it is beginning to notice some vibration so maintenance crew should keep track of the particular machine. They should try to investigate the reason and should plan an action during nest regular stop of the machine. They should keep the machine under observation and using the handheld vibration device crew should measure at shorter time intervals than before to detect any further vibration increments. Those new readings should be compared with the older readings to identify the vibration increment if there any. The level of 7 – 11 mm/s and above 11 mm/s is detrimental to the safe operation of the machine. If the condition of the machine has fallen into this level, the machine has to be stopped if technically and economically possible. Not every machine can withstand this level without causing any internal or external damage to the machine.

Recommendation bearing condition level

The bearing condition value is the total RMS value of the acceleration of all high frequency vibrations within the range from 50 Hz up to 16000 Hz. This is given by a “gBC-value”.

Based on the gBC value and the speed of the machine, it is possible to plot a graph and get following information about the bearing: GOOD, ACCEPTABLE, USEFUL and WORN.

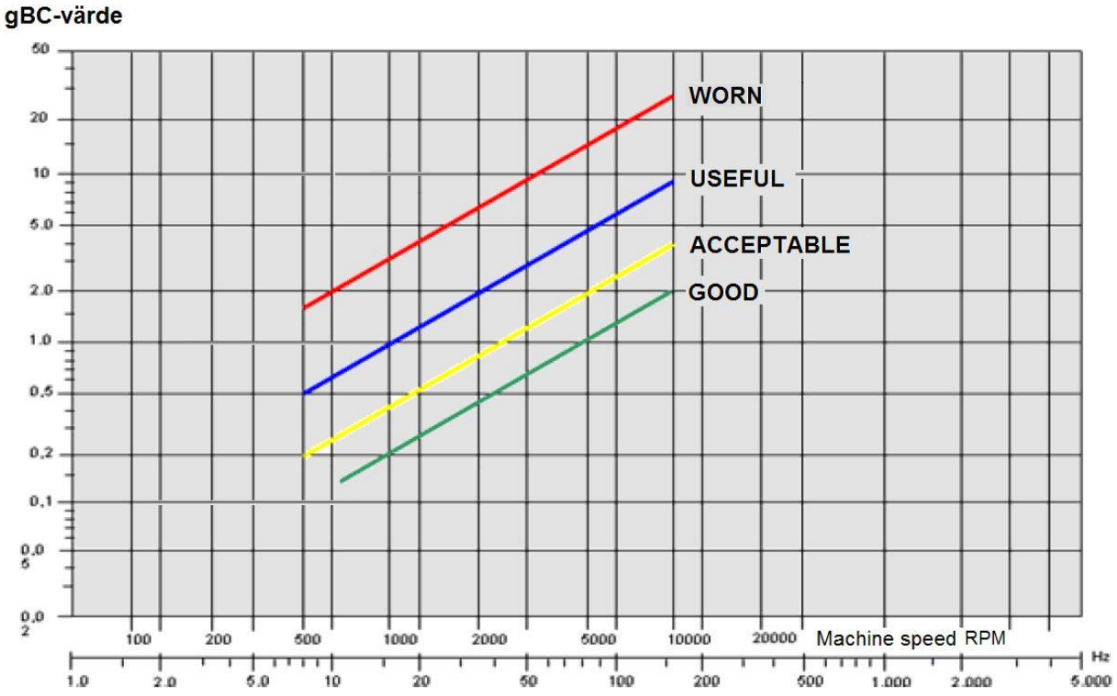


Figure 49 - Viber, Bearing Conditions

3.7. Development of finalized ML model

In the final model implementation, used "1st test" data set from the selected data resource. It contains vibration data of 4 bearings with containing x and y axis readings from each bearing. Then include relevant column names and create data frames for the future implementations.

Then implement commands and functions for data visualize loaded dataset as previously stated in collective approach and in bearing channel wise. Also saved that channel based separated bearing vibration data in csv format for further maintenance model development process.

At that time separate all data into training and testing manner for future data feeding steps. In this instance, used the ratio in typical method by electing approximately 66% of total number of collected data points which means the amount of data in a column for training purpose and

rest of the data for the testing purposes. In this implementation, there were 2156 points of data and there were selected 1437 data points for the training purpose.

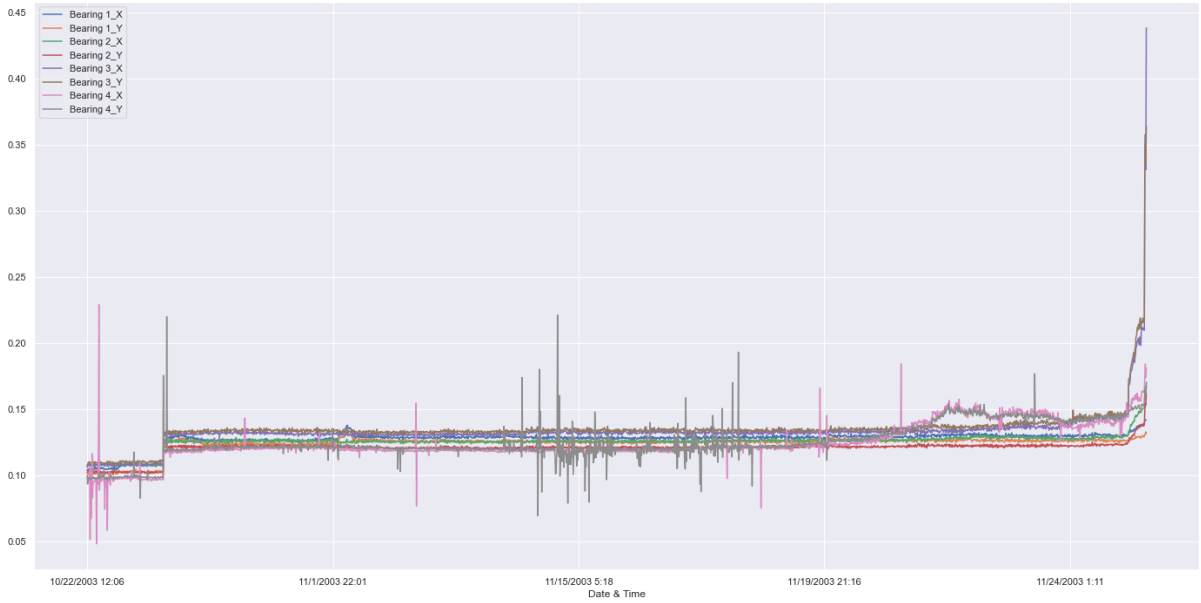


Figure 50 - Bearing Vibration Timer Series

	Bearing 1_X	Bearing 1_Y	Bearing 2_X	Bearing 2_Y	Bearing 3_X	Bearing 3_Y	Bearing 4_X	Bearing 4_Y
0	0.104148	0.100253	0.107147	0.102004	0.106149	0.108150	0.094803	0.099513
1	0.103651	0.099854	0.108189	0.102920	0.106661	0.108458	0.095070	0.093587
2	0.105039	0.101543	0.108543	0.104042	0.108740	0.109875	0.096158	0.098299
3	0.104900	0.101573	0.108152	0.103378	0.108068	0.110010	0.096814	0.098602
4	0.104779	0.102181	0.107943	0.102629	0.108454	0.109350	0.096358	0.098471

Figure 52 - Dataset Sample

	Bearing 1_X	Bearing 1_Y	Bearing 2_X	Bearing 2_Y	Bearing 3_X	Bearing 3_Y	Bearing 4_X	Bearing 4_Y
1437	0.128434	0.125578	0.126885	0.121640	0.132284	0.135108	0.123061	0.122793
1438	0.128577	0.125696	0.126330	0.120965	0.133077	0.134697	0.121705	0.122566
1439	0.127497	0.125105	0.126129	0.122020	0.132355	0.135089	0.122385	0.122108
1440	0.128858	0.125602	0.126435	0.120981	0.131664	0.134150	0.123454	0.124053
1441	0.128380	0.125245	0.126084	0.121655	0.131317	0.132599	0.122143	0.123188

Figure 51 - Dataset Sample 2

Then, preprocessed the data by using “MinMaxScaler” as above discussed and did a random shuffle in training data to reduce the uniformity in the feeding data.

After that configured the PCA model by initiating training and testing data in to the developing PCA model and defining required data frames. Then, develop functions to define required operations with the PCA model. In this step develop functions for operations,

- Calculate the covariance matrix
- Calculate Mahalanobis distance
- Detect Mahalanobis distance outliers
- Threshold value calculation
- Check the matrix is positive definite

Afterward used these developed functions for calculate the covariance matrix and its inverse, Mahalanobis distance to datapoints, calculate threshold value for flagging an anomaly and visualize the Mahalanobis distance.

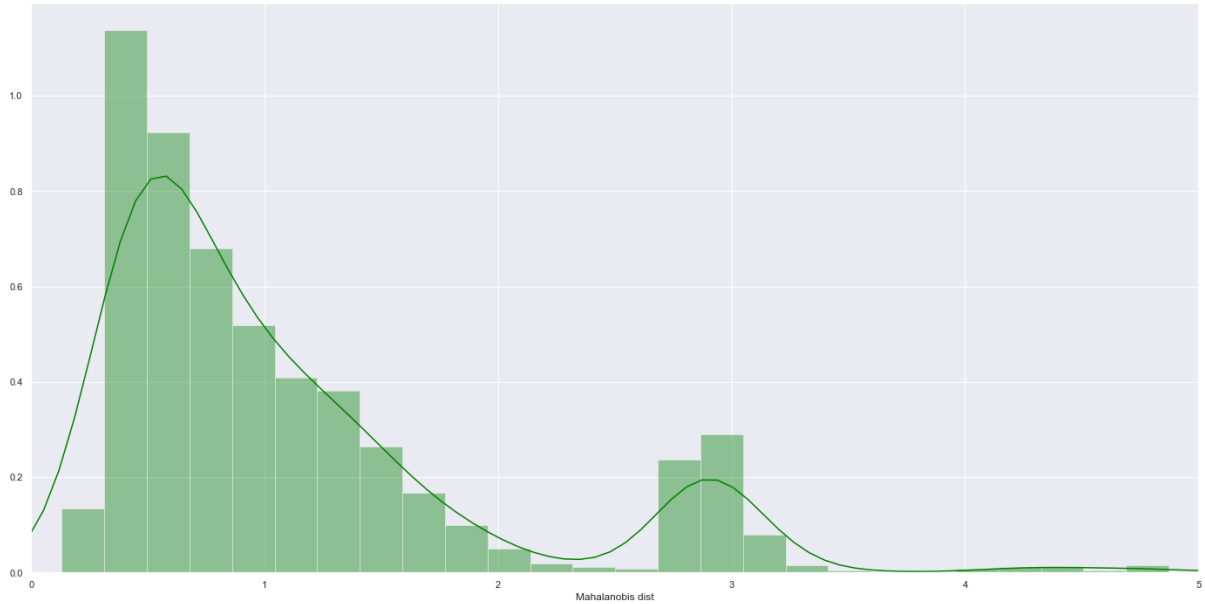


Figure 53 - Flagging an anomaly and visualize the Mahalanobis distance.

Subsequent implemented the program for flag the anomaly by considering calculated threshold value and Mahalanobis distance values in each data points to get that anomaly information in Boolean manner.

	Mob dist	Thresh	Anomaly
1437	3.884466	3.366935	True
1438	4.139639	3.366935	True
1439	3.301553	3.366935	False
1440	4.194975	3.366935	True
1441	3.754678	3.366935	True

Figure 54 - Anomaly information in Boolean manner.

Afterwards, merged relevant Date and Time column into the Anomaly flagged data to visualization purposes. And use predefined testing data for the validation process and verify the developed PCA model with visualizing the resulted data.

	Date & Time	Mob dist	Thresh	Anomaly
0	2003-10-22 12:06:00	3.138008	3.366935	False
1	2003-10-22 12:09:00	3.283228	3.366935	False
2	2003-10-22 12:14:00	3.056239	3.366935	False
3	2003-10-22 12:19:00	3.025401	3.366935	False
4	2003-10-22 12:24:00	2.960883	3.366935	False

	Date & Time	Mob dist	Thresh	Anomaly
1437	2003-11-19 09:22:00	3.884466	3.366935	True
1438	2003-11-19 11:04:00	4.139639	3.366935	True
1439	2003-11-19 11:06:00	3.301553	3.366935	False
1440	2003-11-19 11:16:00	4.194975	3.366935	True
1441	2003-11-19 11:26:00	3.754678	3.366935	True

Figure 55 - Anomaly Detection Biggening to End

As above shown, there are considerable fluctuations in calculated Mahalanobis distance values in Date and time domain. There used a smoothing technique, exponentially weighted window functions and developed code for calculate exponentially weight average and reduce the fluctuations and increase the smoothness in the curve.

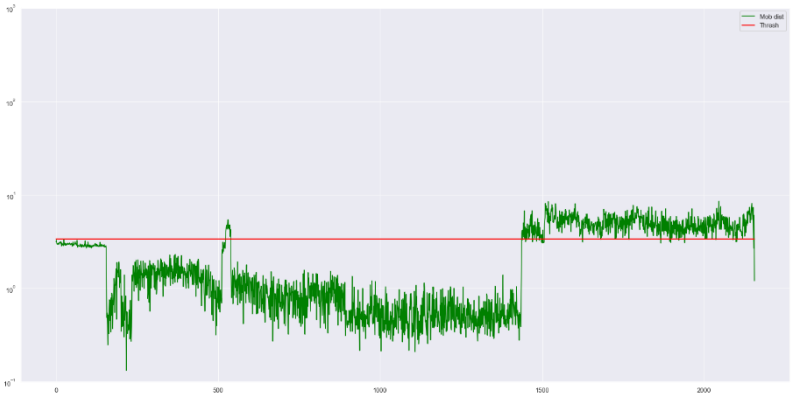


Figure 56 - Resulted MD values on finalized model.

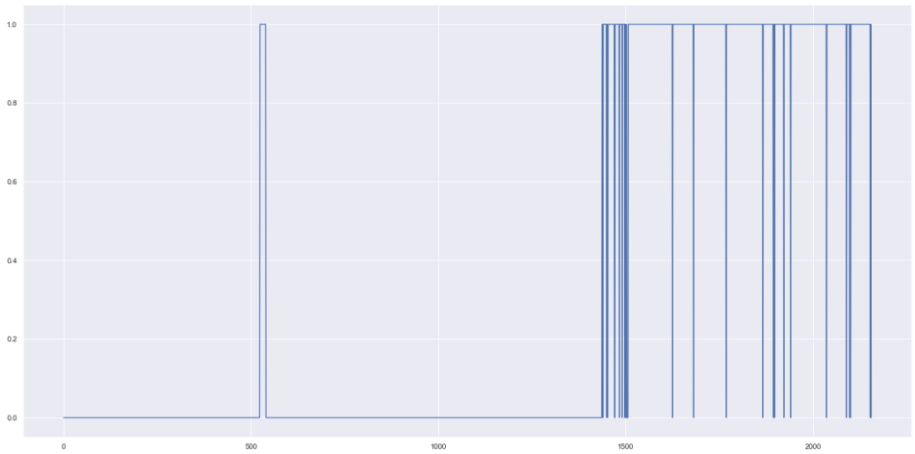


Figure 57 - Corresponding anomaly flagging

After that calculated the flagging anomaly by comparing with threshold value using the smoothed curve and acquired the final flagging from this approach.

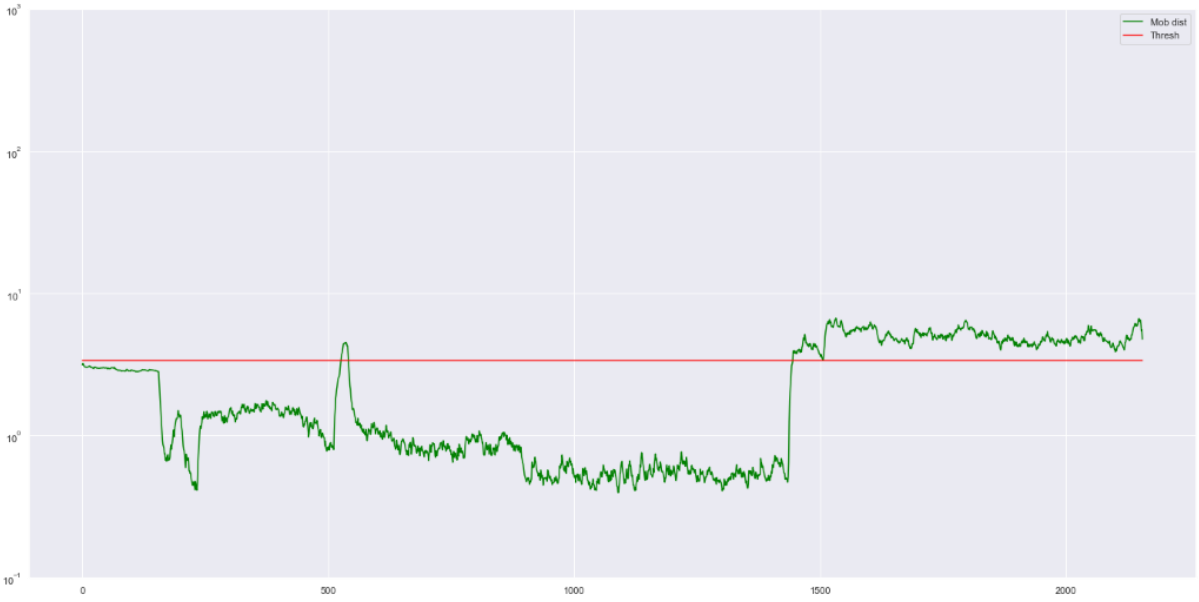


Figure 58 - Smoothed MD graph

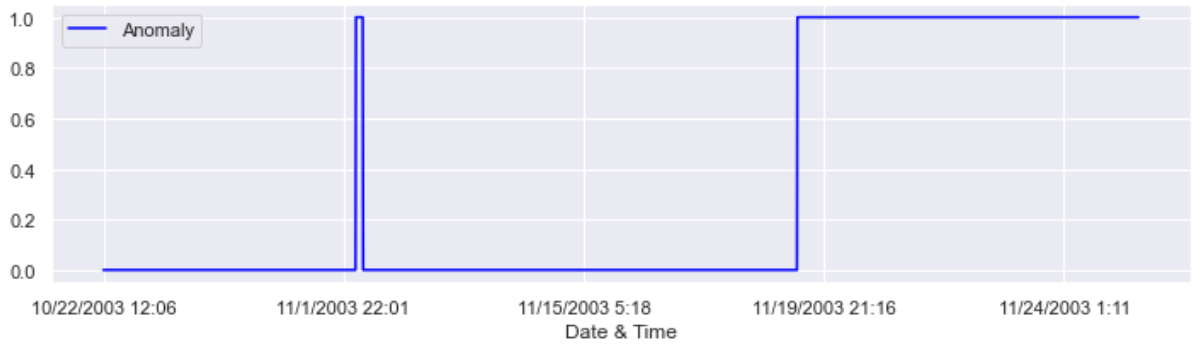


Figure 59 - Anomaly flagging corresponding to smoothed MD.

According to the provided information of selected data set “1st test” at the end of the test-to-failure experiment, inner race defect occurred in bearing 3 and roller element defect in bearing 4 in end of the data acquisition. Then consider about the resulted information the developed model could detected the failure approximately 6 days before the failure.

After all these model developing stages there implemented computational command for detect rising and falling edges in the anomaly graph and calculated flagging times and select the considering anomaly period. Then by assigning a time and indicate failure warning if the anomaly period exceeded the considered flagging time for schedule maintenance and indication process as appropriate output way.

3.8 Theoretical analysis model development

Thereafter, implemented a program for fundamental theoretical analysis to get a feedback about the fault indication period data. In this case as previously discussed used concepts related with failing frequencies of each bearing.

First, there loaded vibration data in time series domain by considering the range of date and time in selected flagging period data and apply fast furrier transform (FFT) to convert time series data into frequency domain to future developments.

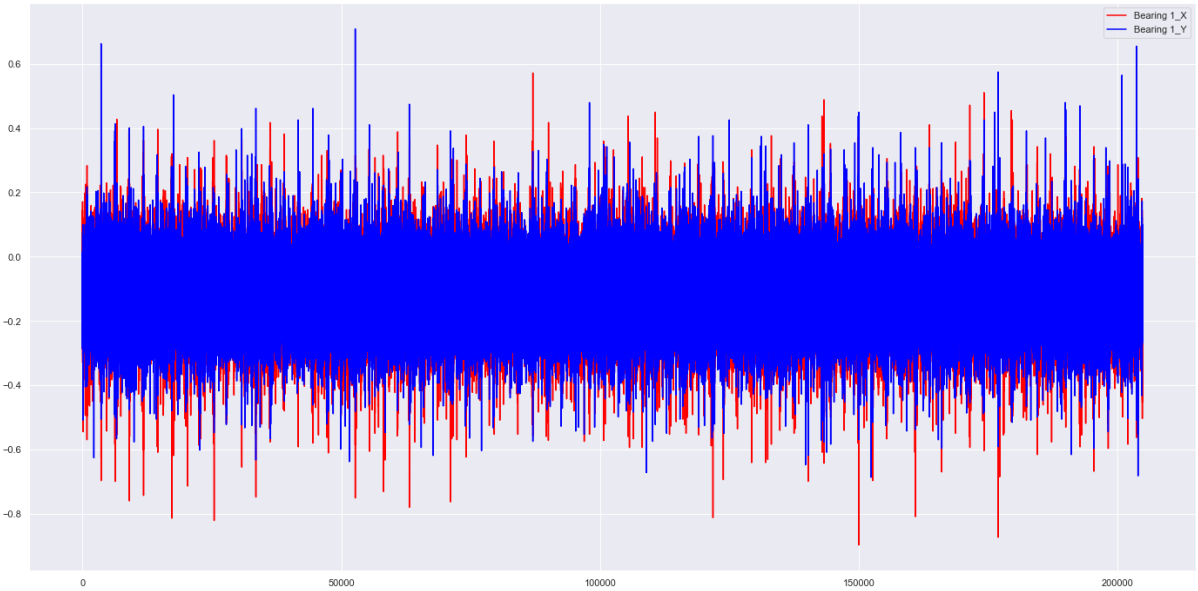


Figure 60 - Raw vibration data of bearing 1

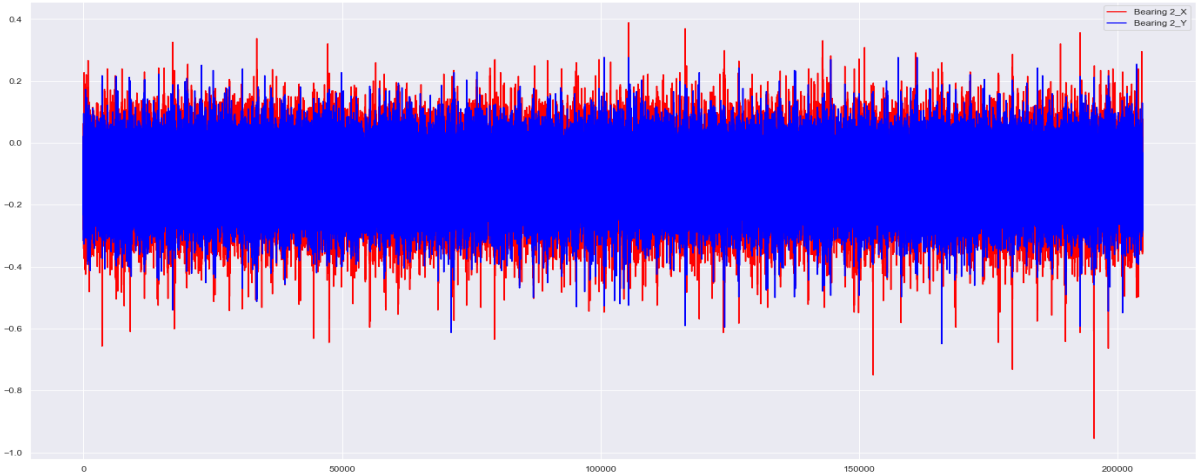


Figure 61 - Raw vibration data of bearing 2

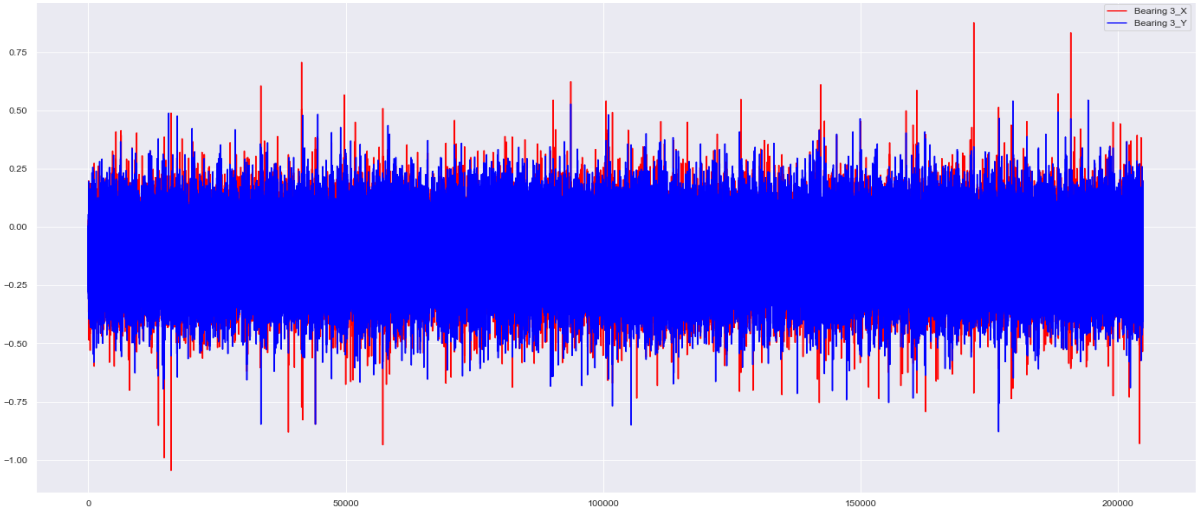


Figure 62 - Raw vibration data of bearing 3

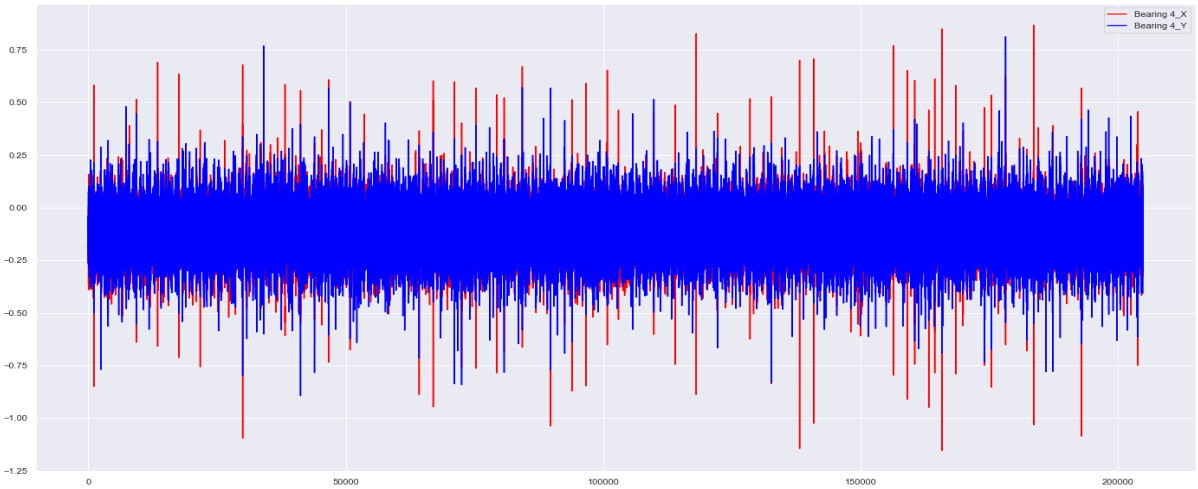


Figure 63 - Raw vibration data of bearing 4

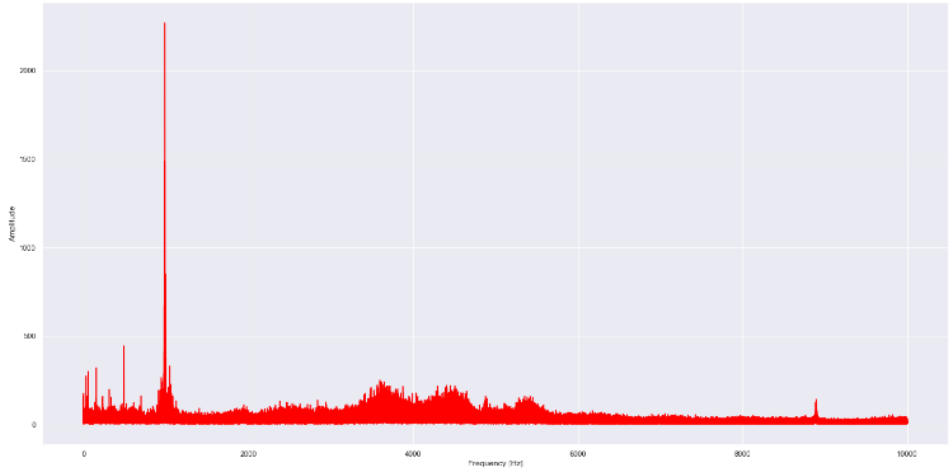


Figure 64 - Bearing 1_X data in frequency domain.

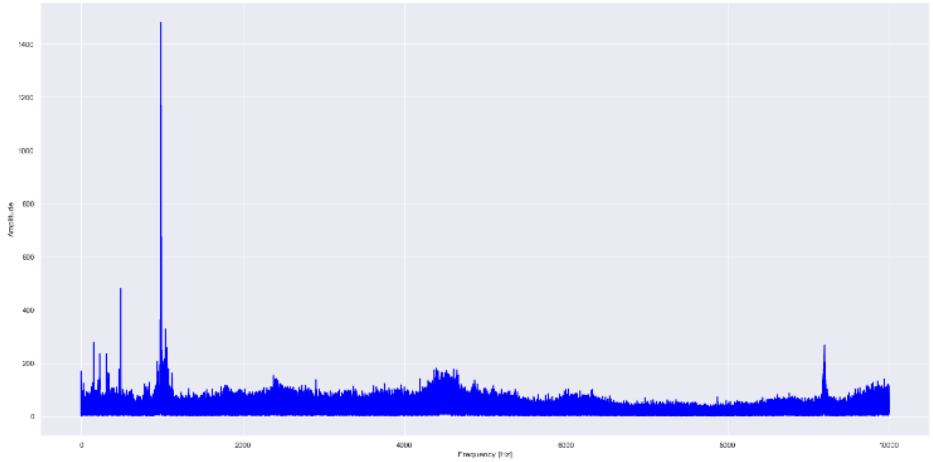


Figure 65 - Bearing 1_Y data in frequency domain.

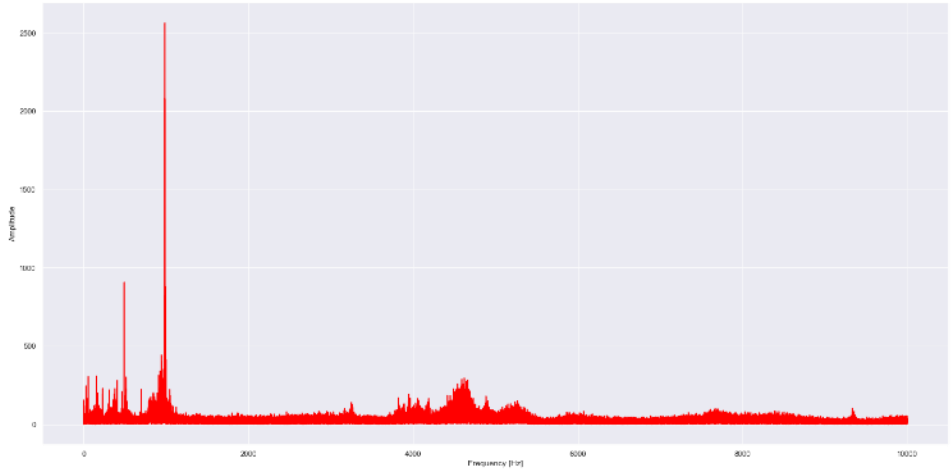


Figure 66 - Bearing 2_X data in frequency domain.

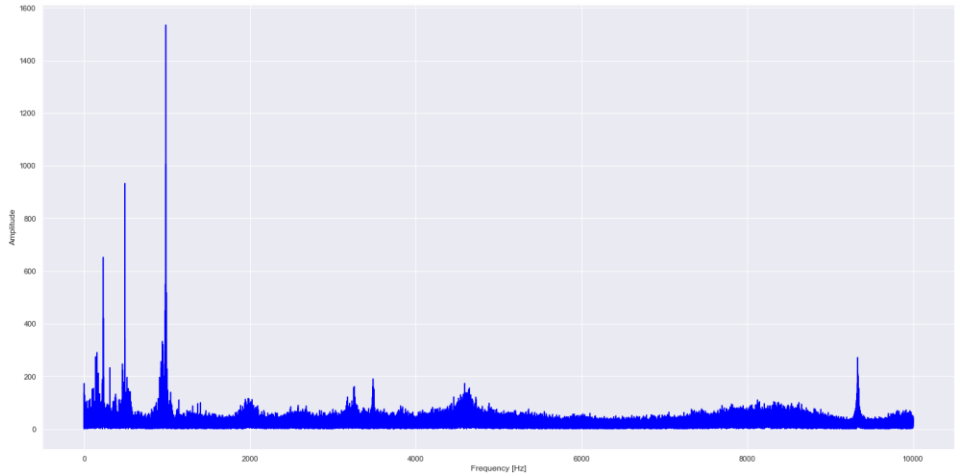


Figure 67 - Bearing 2_Y data in frequency domain.

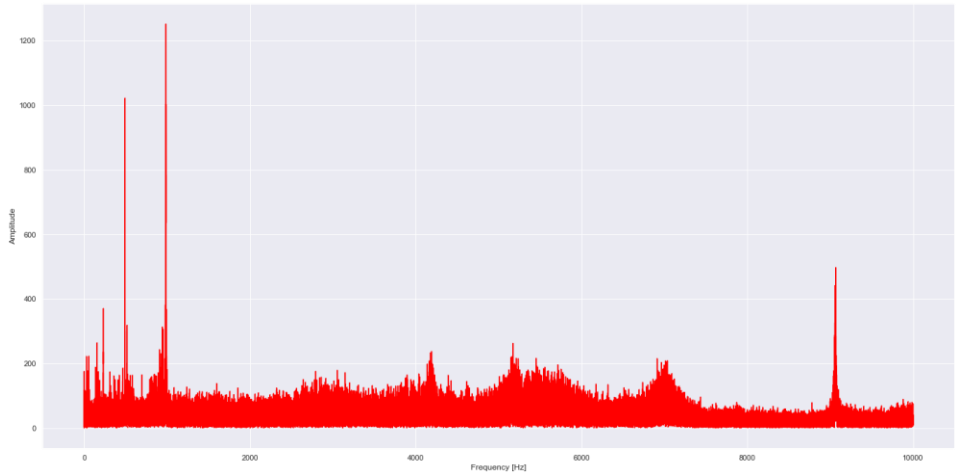


Figure 68 - Bearing 3_X data in frequency domain.

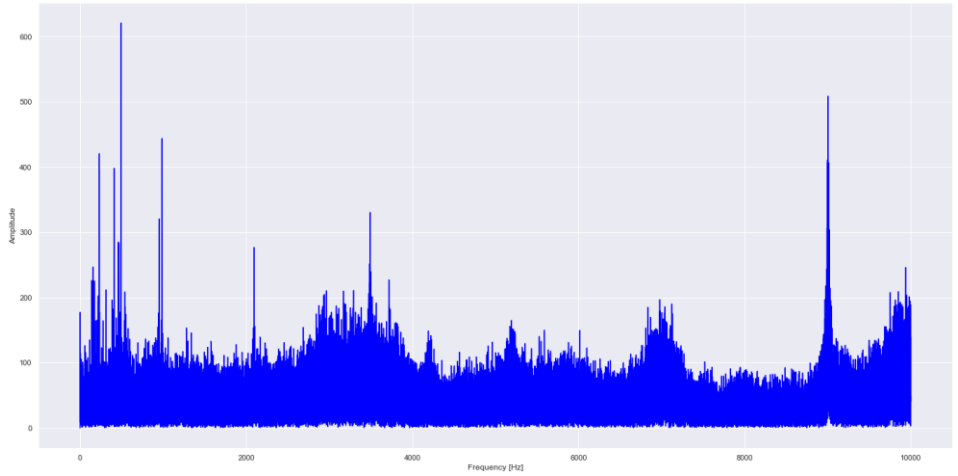


Figure 69 - Bearing 3_Y data in frequency domain.

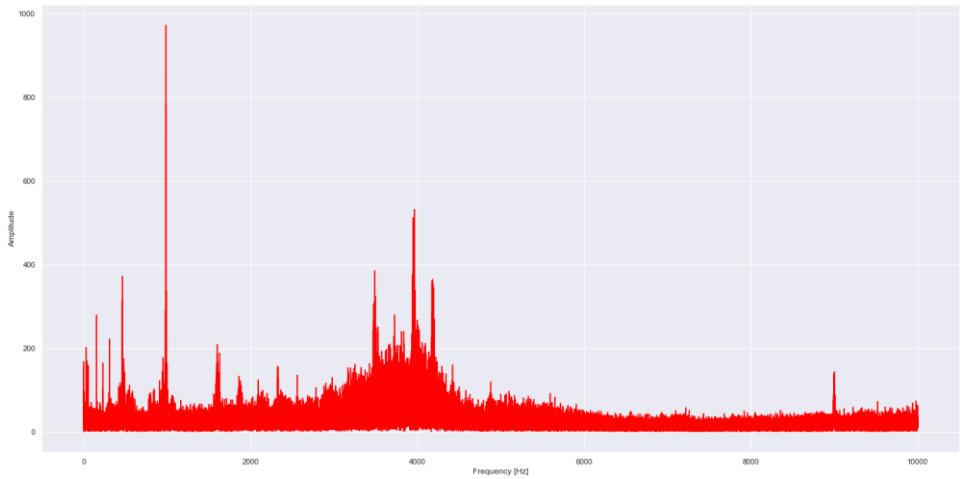


Figure 70 - Bearing 4_X data in frequency domain.

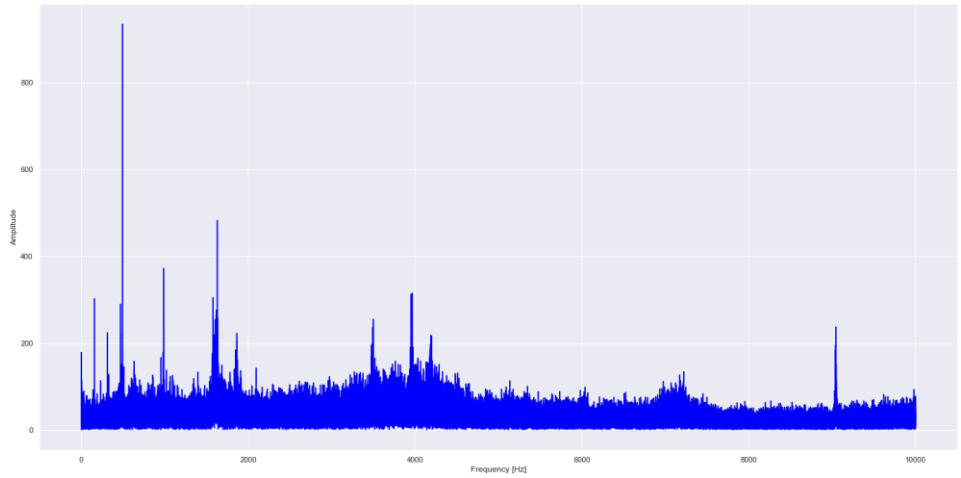


Figure 71 - Bearing 4_Y data in frequency domain.

Then received Bearing details of each selected from stored file with containing bearing channel number, model, ball diameter, pitch diameter, number of rolling elements, contact angle and calculate the BPFO, BPFI, FTF, BSF frequency values.

	Bearing channel No	Model	Ball Diameter	Pitch Diameter	Num of Rolling Elements	Contact Angle	BPFO	BPFI	FTF	BSF
0	1.0	ZA2115	0.0084	0.0715	16.0	15.17	236.4297	296.9036	14.7768	4.2012
1	2.0	ZA2115	0.0084	0.0715	16.0	15.17	236.4297	296.9036	14.7768	4.2012
2	3.0	ZA2115	0.0084	0.0715	16.0	15.17	236.4297	296.9036	14.7768	4.2012
3	4.0	ZA2115	0.0084	0.0715	16.0	15.17	236.4297	296.9036	14.7768	4.2012

Figure 72 - Bearing details.

Afterwards generate harmonics lines by referring calculated BPFO, BPFI, FTF, BSF frequencies and apply that harmonics into the frequency domain graphs and manually inspect and get a feedback about type of the failure and accumulate that data for upcoming

implementations which with capability to forecast the failure type. But in this application, there were only capable to use BPFO and BPFI frequencies in practical manner because of relatively FTF and BSF has small spaces between harmonic lines and for this kind of process there need to generate graphs with high detailed. But that process consumes more computational power and time and that would be resulted to decrease the efficiency of the developing system.

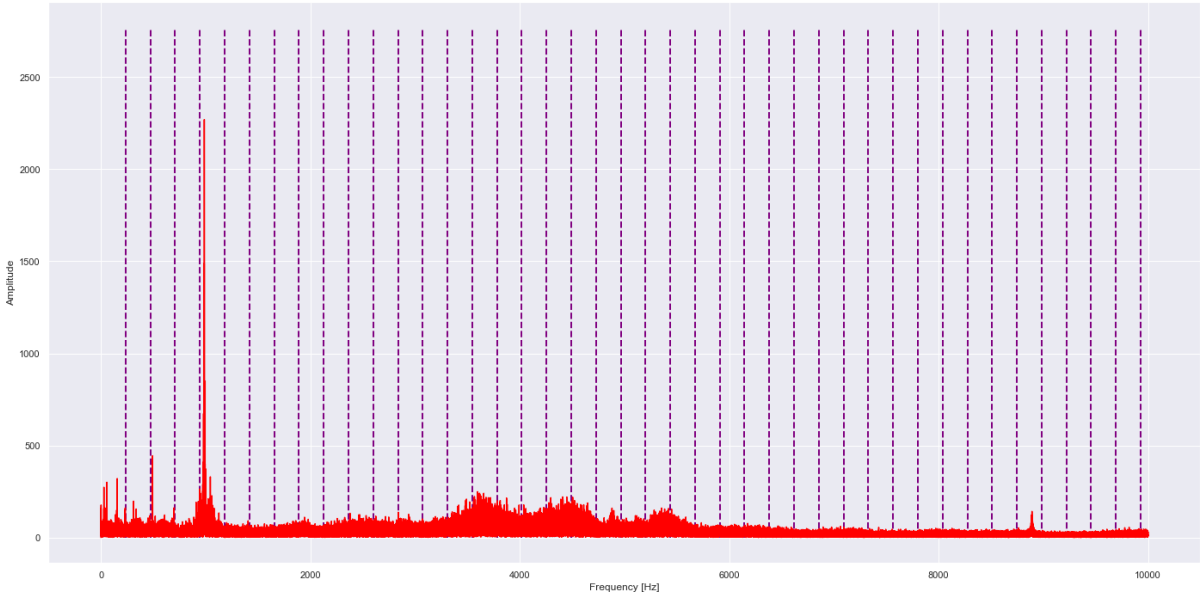


Figure 73 - BPFO harmonics in bearing 1_X frequency domain data.

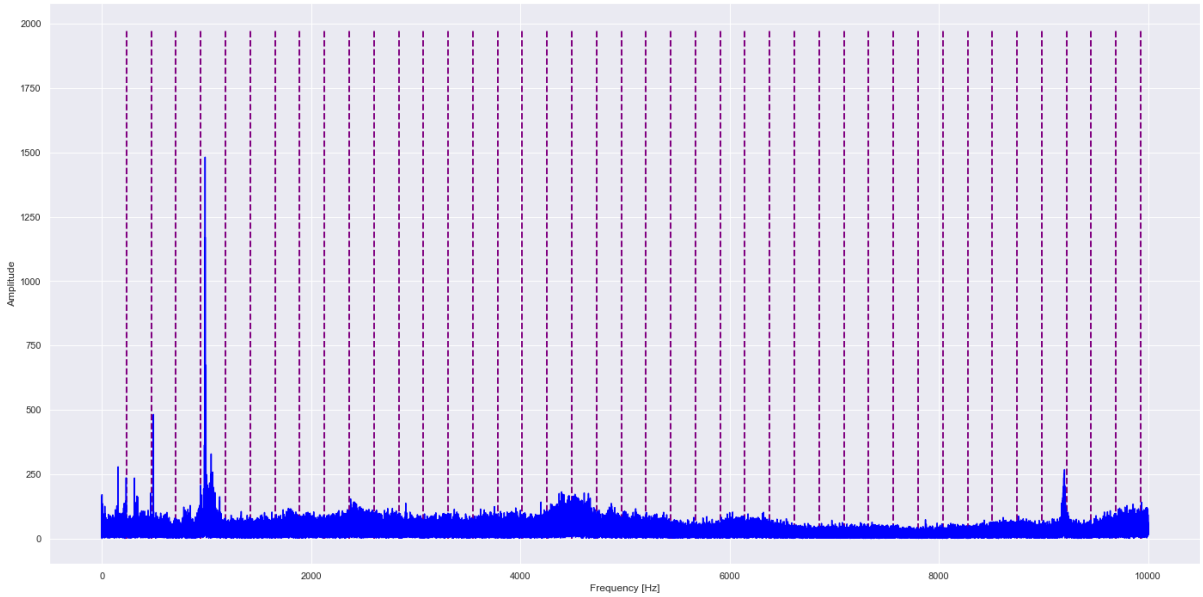


Figure 74 - BPFO harmonics in bearing 1_Y frequency domain data.

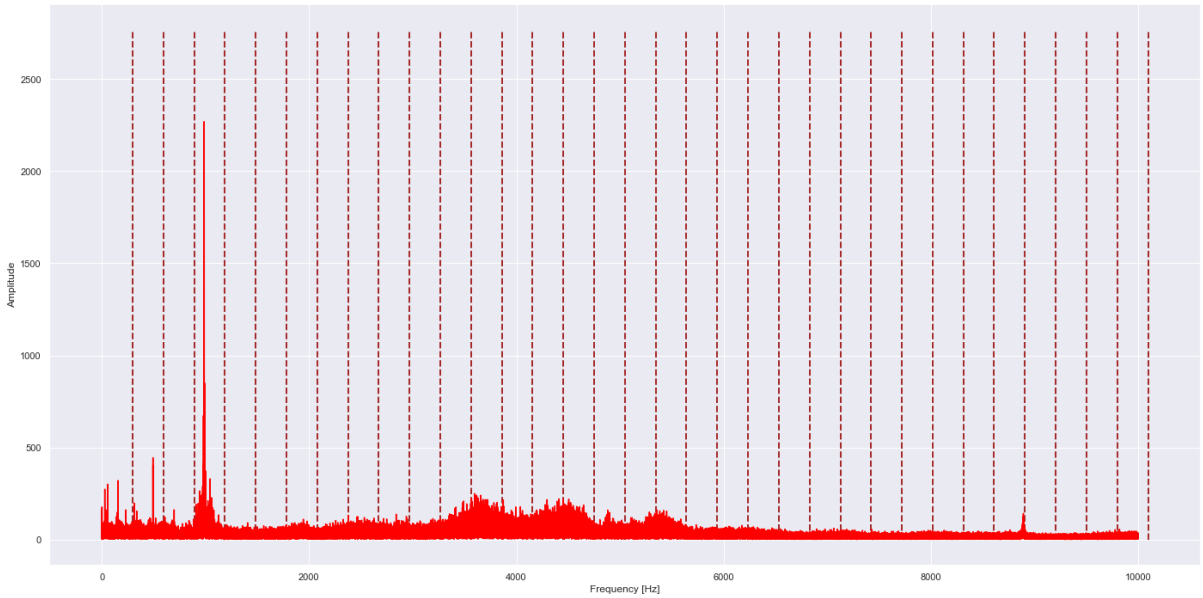


Figure 75 - BPFI harmonics in bearing 1_X frequency domain data.

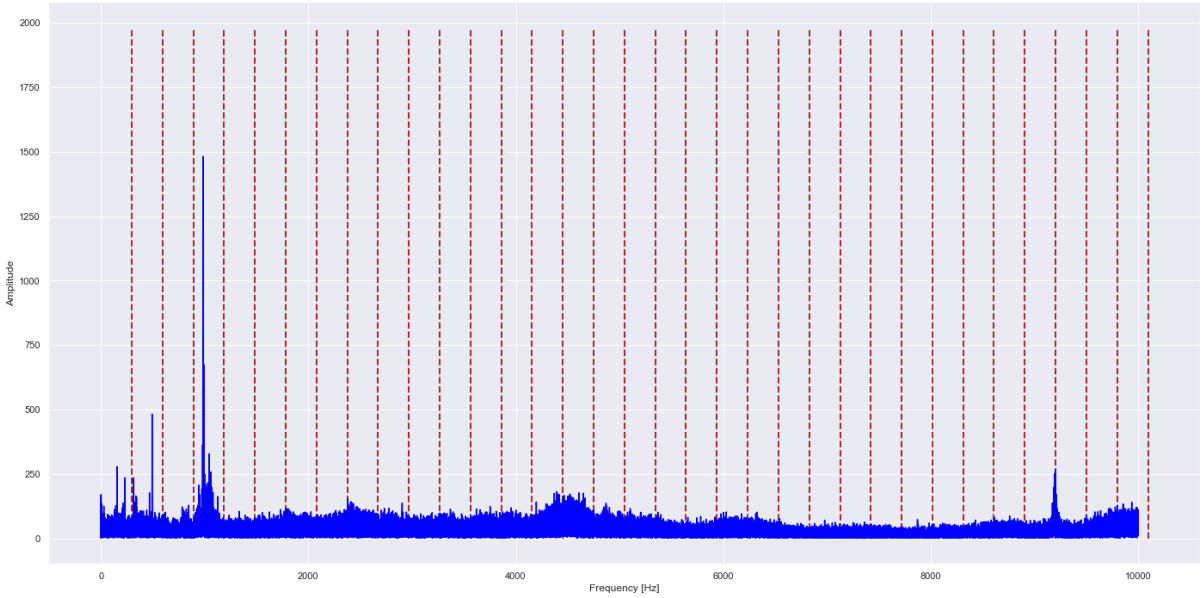


Figure 76 - BPFO harmonics in bearing 1_Y frequency domain data.

Then consider about the calculated BPFO, BPFI, FTF, BSF frequencies in the loaded bearing details data frame there could see BPFO and BPFI values in each bearing are contain with noticeably high values compared with the FTF and BSF values. Because of that this BPFO and BPFI contain visibly large gaps between each harmonic lines in visualization and helpful to identify the bearing failure type in theoretical analysis manner. But in FTF and BSF was difficult to analyze with human visual inspection with generated graphs with comparatively small resolution. Nevertheless, this analytical approach could deliver impression about bearing failure type as a secondary method.

3.9 Maintenance Scheduling

Thereafter if there were exceptional anomaly in the vibration data of each bearing the fault bearing/s data will update with indicating number of the relevant bearing with space separation. If there detected bearing 1,2 and 4 this detail stored as,

1 2 4

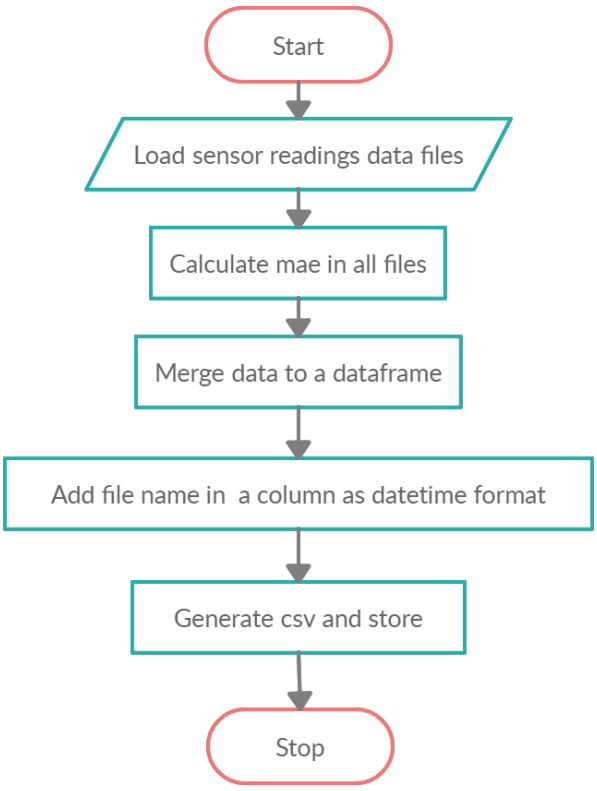
Figure 77 - Space separated defected bearings.

In the ML model development process, there were indicated bearing number 4 going to be fail in future. In this development process there were managed to achieve failure indication approximately 5 day before the actual failure.

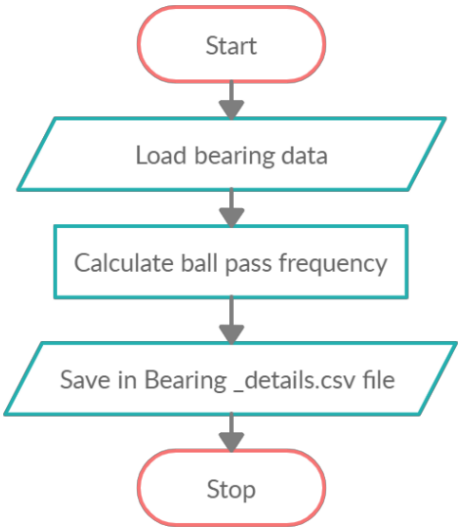
	Machine No	Date	Fault Bearing / s
0	BM02	2003-11-24 00:00:00	4

Figure 78 - Maintenance scheduling chart

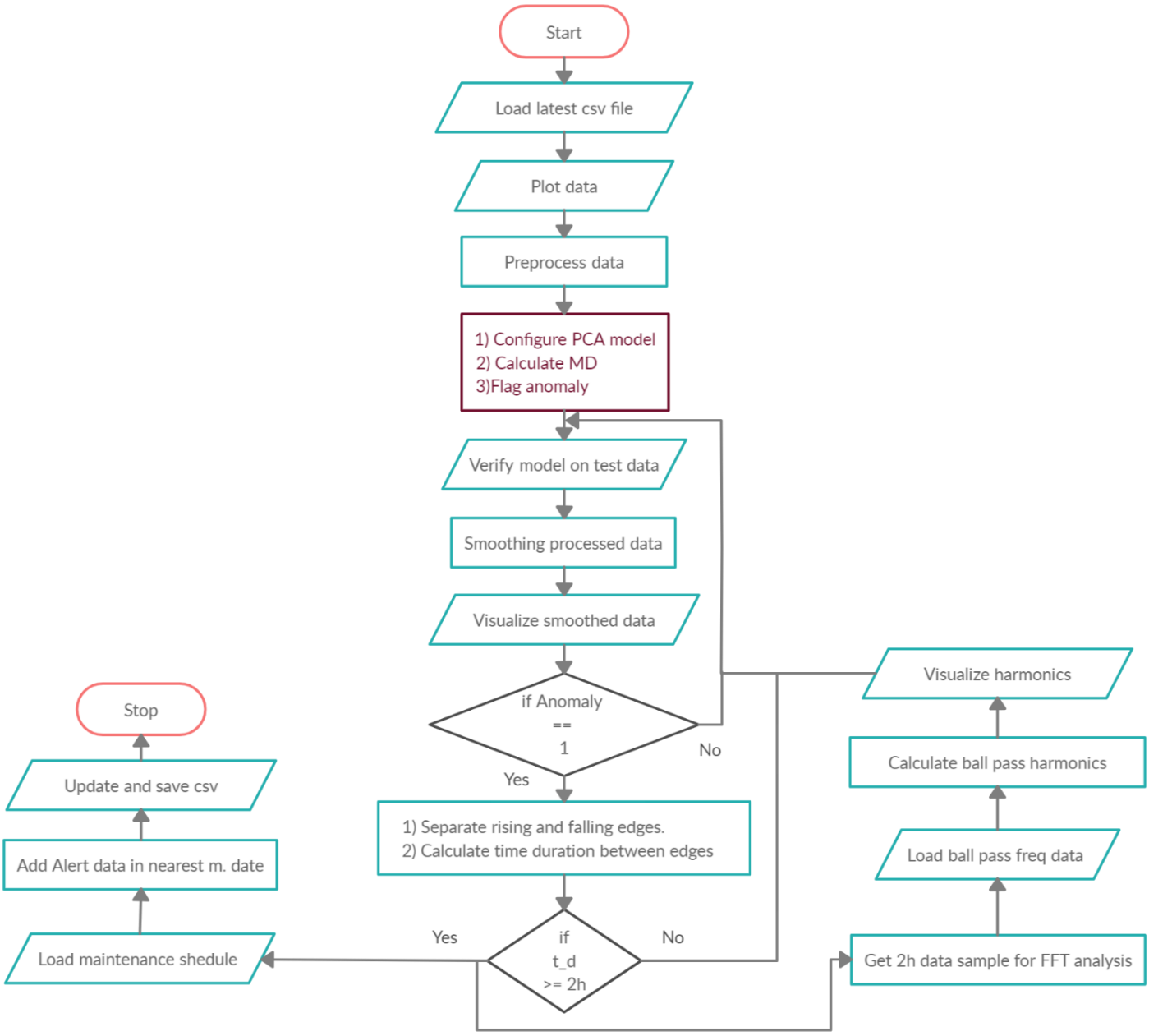
3.10. Flowchart of data preprocess part



3.11 Flowchart of theoretical analysis part



3.12. Flowchart of ML part



3.13. Data acquisition

Then afterwards selecting critical points, accelerometers, signal conditioner and PLC module along with analog card module and model development process the next most important step is to collect considerable amount of data to train and test developed model and validate and troubleshoot variable in proposed approach with considering the accuracy and visibility of the results.

At that moment, due to the COVID 19 pandemic situation and the ordered components associated with the project from china arrived late in end of the December month and the annual maintenance of the Bielomatik machine scheduled from in January, there had to limit data acquisition process in to 2 weeks duration. Also, it results to finalize the designed control box and due to verify the robustness and measurements are correct and because of that, for troubleshooting and validation there had to collect vibration data from the machine by manually executing the developed python script with inspecting near the machine for data collecting process and the assembled test bench.

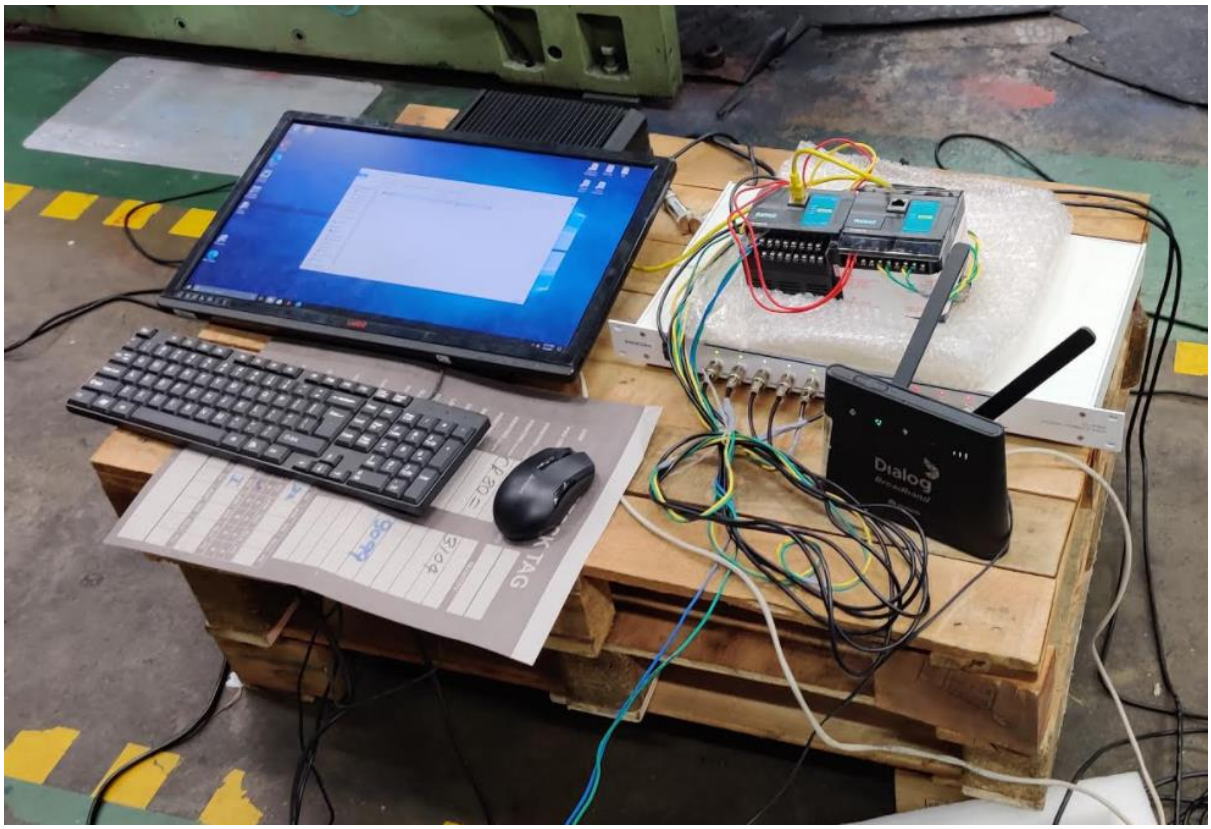


Figure 79 - Completed Test Setup

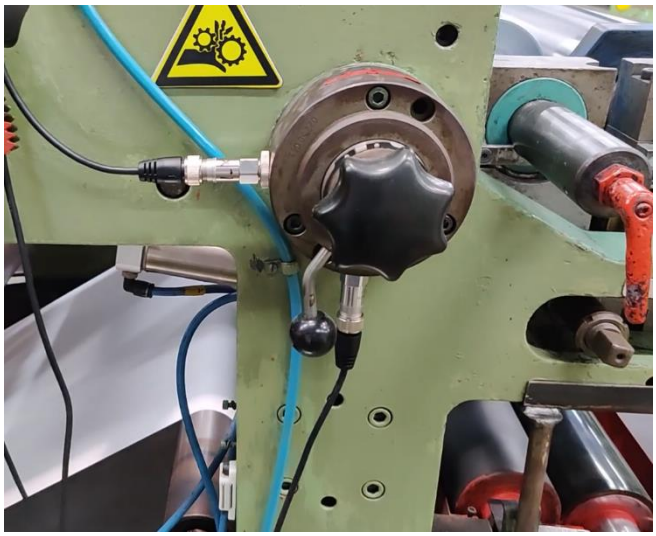


Figure 80 - Probe Connection for Critical Areas

When consider about the structure of the data transferring from PLC to Industrial PC or SBC, there are many communication protocols facilitate with both components. Some of them are, RS485, RS232, TCP/IP, Modbus TCP etc. In this application there were selected Modbus TCP communication protocol for transfer amplified vibration signal from signal conditioner to PLC connected analog card. Then developed a python script to collect that signals in given recording interval and recording time.

3.14 PLC and the Analog Expansion Module

3.14.1. Communication and Implementation

The selected PLC was Haiwell H16S2T with the analog expansion S08AI up to 8 inputs. The main reason for the Haiwell PLC selection is because the PLC has the inbuilt IoT support when connected to internet using Haiwell HMI or a CloudBox. Since the signal conditioners out was 4mA constant current with 0-10V signal, we wired the PLC accordingly.

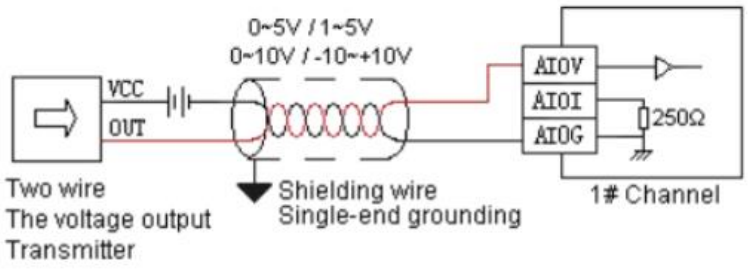
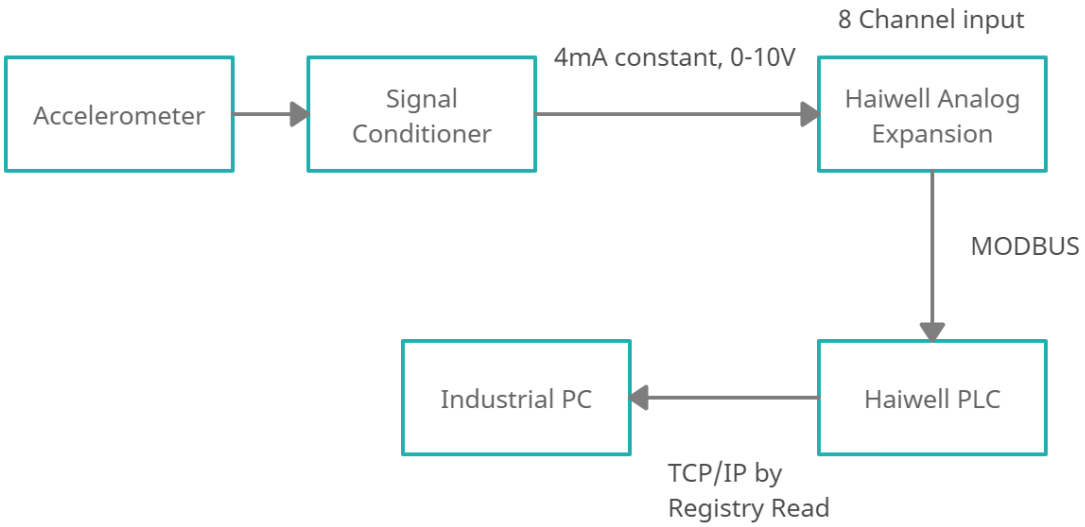


Figure 81 - Analog Expansion Wiring Diagram

The selected protocol was MODBUS.



Some hardware configurations were done to the analog module and the PLC, so the output signal can be fed into the Industrial PC using TCP.

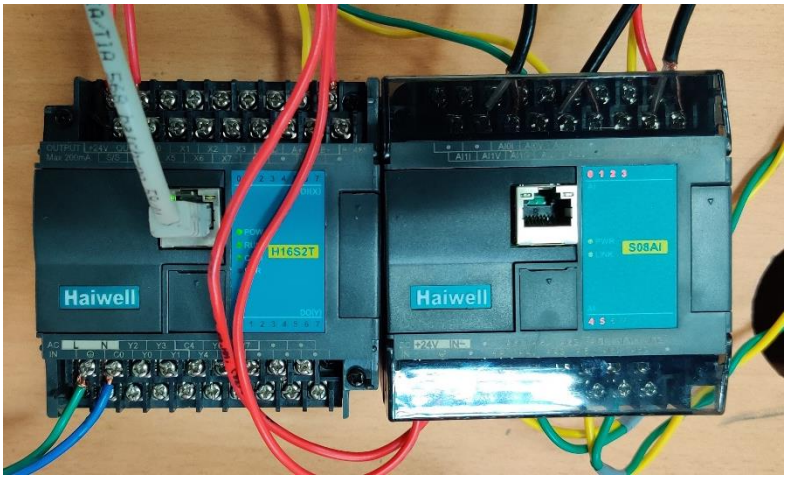


Figure 82 - Communication between Devices

MODR Block in the Haiwell PLC was used to read the Analog input of the PLC.

Figure 83 - MODBUS READ Block PLC

The analog expansion was considered as the Slave device 1. To read the Holding registers for the updating value, we used the exact registry read command. We tend to acquire a 16-bit value throughout the process. Each channel memory was assigned to registers for later use. The holding Registers for the 6 consecutive channels were 512, 513….,517 in the PLC.

Component use table

Component	Read	Write	Used component by instruction(R/W)	Component comments
M0	0	1	0 / 0	
M1	0	1	0 / 0	
M2	0	1	0 / 0	
M3	0	1	0 / 0	
M4	0	1	0 / 0	
M5	0	1	0 / 0	
V0	0	1	0 / 0	
V1	0	1	0 / 0	
V2	0	1	0 / 0	
V3	0	1	0 / 0	
V4	0	1	0 / 0	
V5	0	1	0 / 0	

Figure 84 - Memory allocation for Registries

3.14.2. Registry Read

In this implementation specifically used pymodbus library with calling pymodbus.client.sync and ModbusTcpClient modules. In this case the python script running device act as client device. Also used datetime library for get current timestamp for recording dataset and pandas library for handle that data in data frames with arranging data and save in csv format in specific locations in the directory.

Furthermore, a briefly description about the flow of the commands in the contain of the program as a start listen and collect data from assigned IP address of PLC and the predefined memory elements in the PLC ladder diagram which delivering X and Y axis data from the selected 3 critical points. Subsequently filter each received data and converted in to required data type and cluster value data from the received data pack. Then store that data in a new row on bottom of a defined data frame with corresponding date and time.

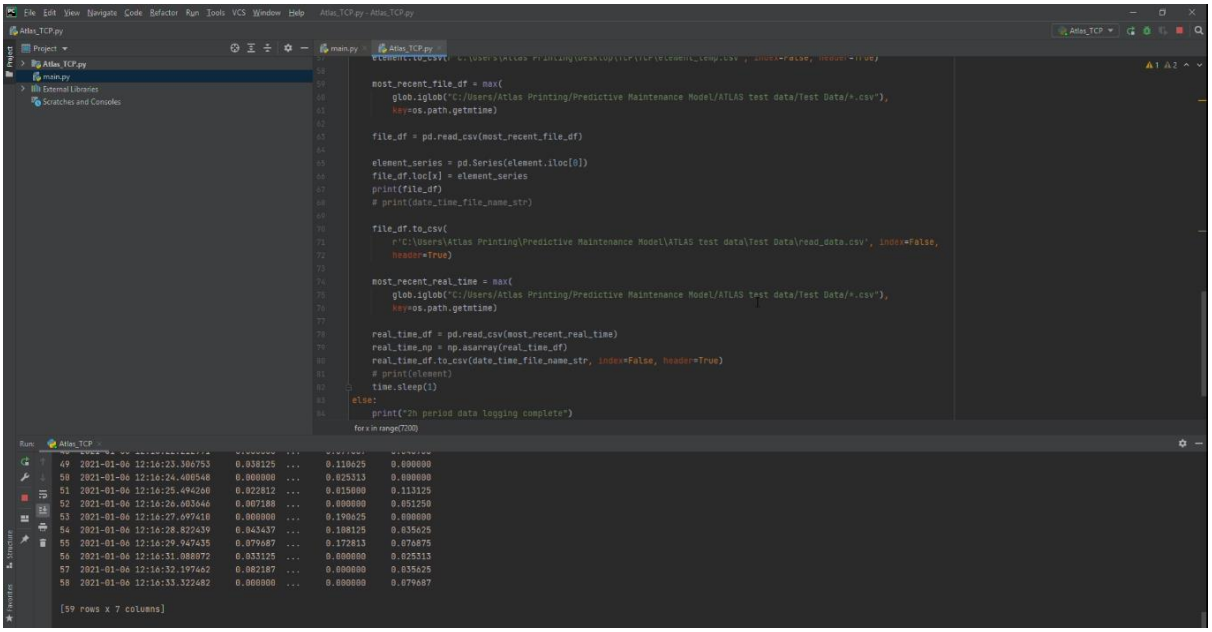


Figure 85 - Data collecting process 1

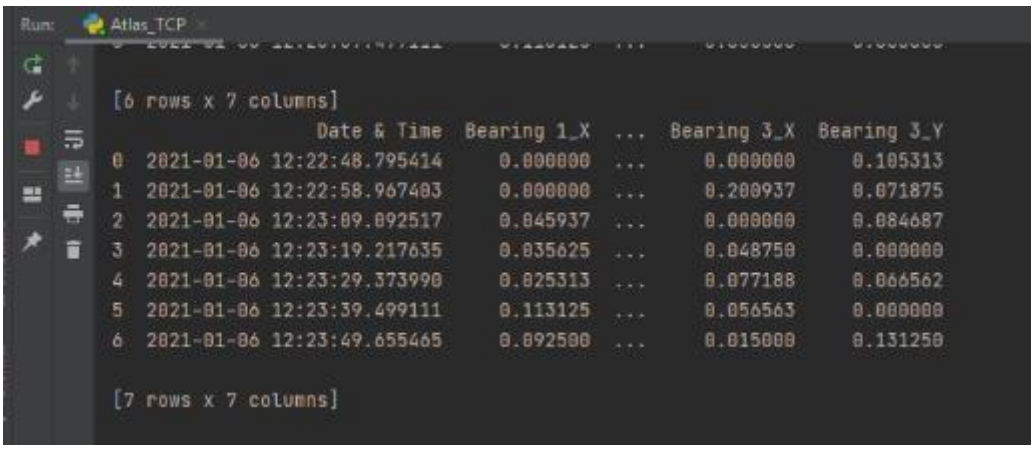
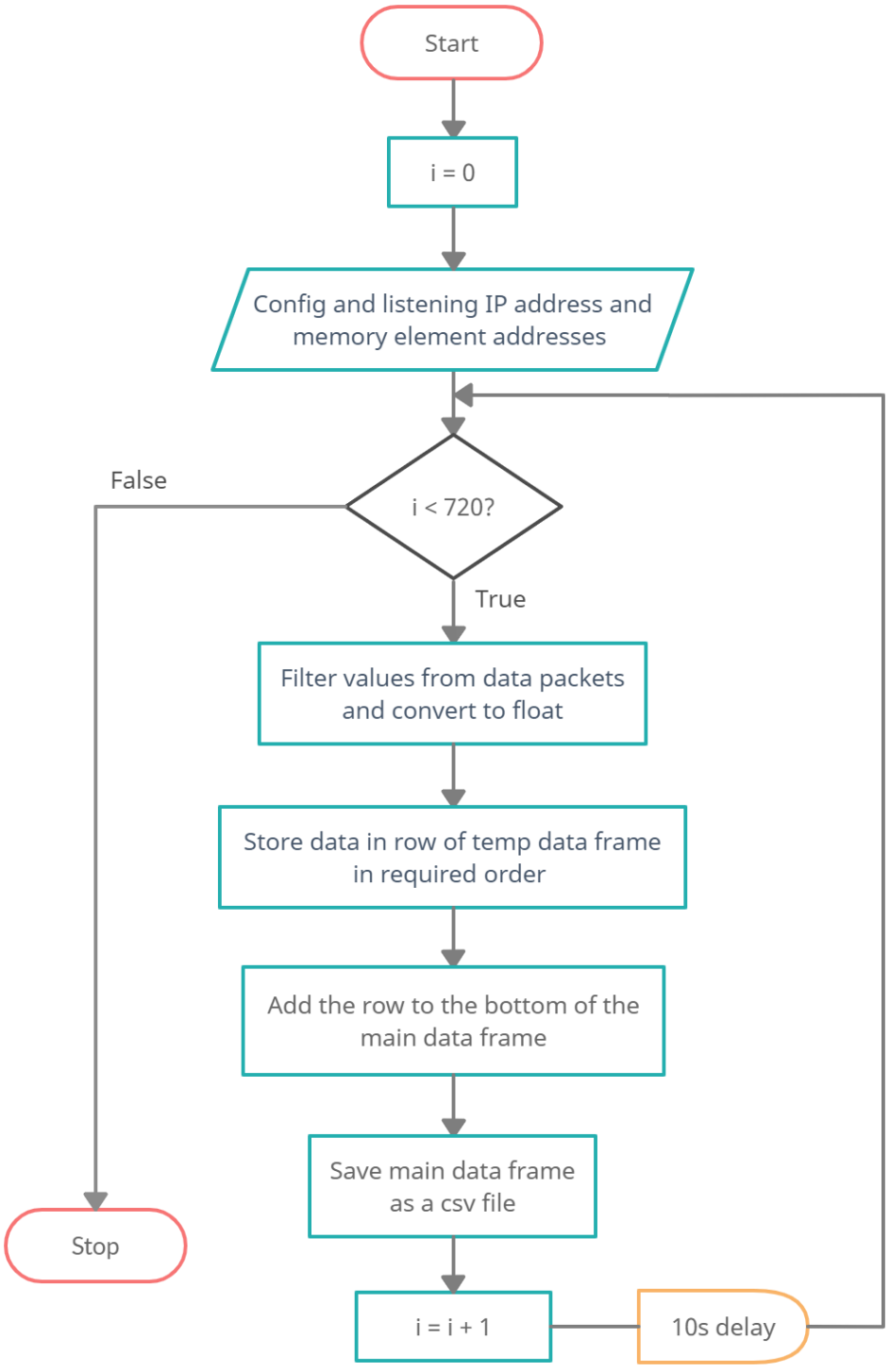


Figure 86 - Data collecting process II

Also, each csv file named and saved as the data recording starting date and time details in this data acquisition. In this process each recorded file included 2h worth vibration data in 10 seconds recording interval. In other words, each file contains 720 rows with vibration data in 10 seconds intervals.

3.14.3. Flowchart of data acquisition part



3.15 Developed model file architecture.

In the ML model development process, there mainly considered about the structure of the file directory in the data storage and program storing locations. Because to avoid malfunctioning due to the unnecessary file overlapping and expand capability of easy access to processed data for export into the dashboard and troubleshooting process. Following are brief description about each directory contained on the currently developed model.

1) ATLAS anomaly –

   This folder contains files of the generated anomaly distance data which means Mahalanobis distance data with calculated threshold value and the Boolean anomaly value of each MD compared with the threshold value. Correspondingly, the folder included smoothed data file of that previously stated variable in separate file name. These files are update and rewrite every ML model execution time and hold with the processed data of current testing data.

Date & Time	Mob dist	Thresh	Anomaly
11/19/2003 9:22	3.884466	3.366935	TRUE
11/19/2003 11:04	4.139639	3.366935	TRUE
11/19/2003 11:06	3.301553	3.366935	FALSE
11/19/2003 11:16	4.194975	3.366935	TRUE
11/19/2003 11:26	3.754678	3.366935	TRUE
11/19/2003 11:36	5.015271	3.366935	TRUE
11/19/2003 11:46	3.547327	3.366935	TRUE
11/19/2003 11:56	3.911633	3.366935	TRUE
11/19/2003 12:06	6.743076	3.366935	TRUE
11/19/2003 12:16	3.437522	3.366935	TRUE
11/19/2003 12:26	3.703546	3.366935	TRUE
11/19/2003 12:36	4.534012	3.366935	TRUE
11/19/2003 12:46	3.137618	3.366935	FALSE
11/19/2003 12:56	3.714782	3.366935	TRUE

Figure 87 - Anomaly Data Table

2) ATLAS bearing details –

   This folder included all monitoring joints' bearing model details, ball diameter, pitch diameter, number of rolling elements, contact angle and ball pass frequency values which are BPFO, BPFI, FTF, BSF along with corresponding channel number. These data stored in the file as csv format and principally use that data for theoretical analysis process in the bearing fault prediction.

3) ATLAS datasets –

   In this folder there contained the 3 unique test data of a test rig collect data under constant load and speed. This is the data which produced by the Center of Intelligent Maintenance Systems (IMS) of University of Cincinnati. As well, this is the data used for developing the model.

4) ATLAS flag date –

   This folder carried a small csv file including the detected flag details with their corresponding date and time for upcoming process and indication purposes.

5) ATLAS maintenance –

   This folder contained the weekly, monthly, and annual maintenance data with dates and machine number along with a fault bearing/s column. If there were flagged a anomaly, then process flagged data and predict which bearing/s going to be fail and save that bearing numbers on the nearest maintenance date.

6) ATLAS preprocessed data –

   This folder holds merged and normalized data of time series vibration data from each monitored bearing for feed to the developed ML model. This data stored in the file in csv format.

7) ATLAS samples –

   This directory saved row time series data of each bearing in separate files based on the channel number for easy visualization and fast accessible purpose.

8) ATLAS test data –

   The test subjected data of the defined ML model would locate into this folder. Typically, execution subjected data would be discovered on this folder.

9) ATLAS train data –

   The train dataset to find patterns in developed algorithm would be stored on this folder in csv format. This is the healthy condition vibration data of the targeted machine.

10) Firmware –

In this folder, it contains all the program files which related with the data acquisition, ML model for failure prediction, maintenance scheduling and data upload into the dashboard operations. Basically, these files are on .py format and. ipython format.

3.16 Developed predictive maintenance model architecture.

In the developed maintenance model architecture, there could point out 3 main components in the full program. They are,

1) Data acquisition part –

   In this part as formerly discussed, the objective of execution is to accumulate real time data from the mounted accelerometers and in default manual inspection data indicate as Boolean FALSE in data column and this data update by manual inspection of the machine critical points by using the Viber X2 Pro handheld bearing health monitoring device. In this health column FALSE shows there was no flagged failure.

Date & Time	Bearing 1_	Bearing 1_	Bearing 2_	Bearing 2_	Bearing 3_	Bearing 3_Y	Health
1/13/2021 12:10	0.099513	0.122111	0.138102	0.100253	0.131309	0.13146433	FALSE
1/13/2021 12:10	0.093587	0.120992	0.137637	0.099854	0.129782	0.1322191	FALSE
1/13/2021 12:10	0.098299	0.120583	0.136535	0.101543	0.131764	0.13315582	FALSE
1/13/2021 12:11	0.098602	0.121401	0.136519	0.101573	0.130797	0.1326808	FALSE
1/13/2021 12:11	0.098471	0.120959	0.136188	0.102181	0.132354	0.13342697	FALSE
1/13/2021 12:11	0.09818	0.120539	0.136015	0.10173	0.130403	0.13303819	FALSE
1/13/2021 12:11	0.097066	0.122316	0.135983	0.101044	0.130689	0.13321974	FALSE
1/13/2021 12:11	0.097954	0.120871	0.135467	0.101377	0.131181	0.13247561	FALSE
1/13/2021 12:11	0.097696	0.121053	0.135323	0.101754	0.131132	0.13321178	FALSE
1/13/2021 12:12	0.098002	0.121422	0.135265	0.101312	0.130708	0.13358289	FALSE
1/13/2021 12:12	0.097865	0.121001	0.135258	0.101864	0.129425	0.13170648	FALSE
1/13/2021 12:12	0.098033	0.120333	0.135255	0.101691	0.130471	0.13195937	FALSE
1/13/2021 12:12	0.097881	0.122168	0.135097	0.101282	0.129611	0.13189853	FALSE
1/13/2021 12:12	0.098784	0.121109	0.134913	0.101255	0.130182	0.13334128	FALSE
1/13/2021 12:12	0.097946	0.120355	0.134613	0.101218	0.13121	0.13286113	FALSE
1/13/2021 12:13	0.098254	0.121993	0.134414	0.103849	0.130263	0.13295849	FALSE
1/13/2021 12:13	0.097345	0.119996	0.133473	0.101321	0.131009	0.13270472	FALSE
1/13/2021 12:13	0.098271	0.121329	0.132886	0.102662	0.131104	0.13332218	FALSE

Figure 88 - Data Acquisition table

2) ML model and Maintenance Scheduling part –

   As previously evaluated, the PCA algorithm and corresponding MD calculation of each data point and bearing fault classification would be execute by provided python-based scripts in this part. This is the main functioning part of the whole model.

3) Data uploading into google drive through GCP –

   In this part data would upload in to the google drive and that file connect in to the google data studio-based dashboard by using GCP and a python-based script. In this application the uploading process scheduled in every 2h and the dashboard updating operation scheduled in to every 15 minutes periods.

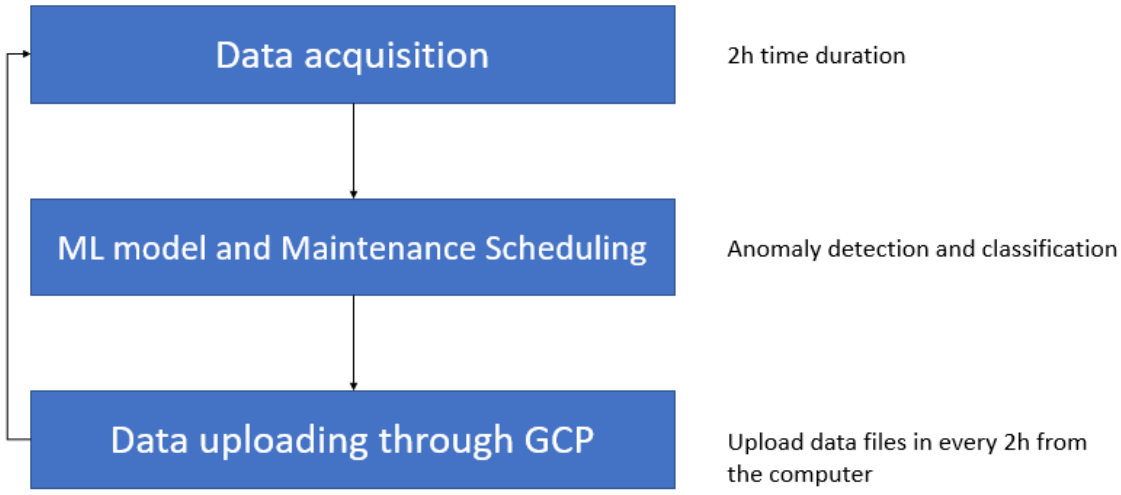


Figure 89 - Predictive maintenance model architecture

This above figure demonstrates the complete Predictive maintenance model in graphical manner. In this operation the whole program run in loop wise and each loop execute on 2h time duration. It means all data updating, calculation process run in every 2h during the operating time. For accomplish this process there were developed a python script which has capability intended for command to select and run each developed python scripts with essential time delays and proposed process sequences.

3.17. Controller Selection

A controller is a device which receives input signals from sensory device and process these values and determines appropriate output signals required by the control element to provide corrective actions withing the control loop of the controller.

3.17.1. Types of controllers

3.17.1.1. Microprocessors

Microprocessor is a central processing unit (CPU) which is on a single integrated circuit chip which handles the processing tasks on a system. It is a small circuit chip which contains millions of transistors, diodes, resistors, and other electronic components.

Limitations of microprocessors

- Over heating
- Does not support floating point operations.
- Minimum size of data
- Does not contain any internal peripherals.

3.17.1.2. Microcontrollers

Microcontroller is an integrated circuit which contains one or more CPUs along with other peripherals such as memory and programmable input/output peripherals. Microcontrollers are used widely used in automatically controlled products and systems.

Limitations of microcontrollers

- Cannot interface with higher power consuming devices directly
- Limitations on number of simultaneous executions
- Delicate internal components are vulnerable for industrial conditions

3.17.1.2.1 Why microprocessors and microcontrollers cannot be used.

The predictive maintenance system is based on machine learning concepts which requires high processing power, especially parallel processing capabilities which microprocessors and microcontrollers both lacks. To execute the machine learning code efficiently, a controller with much higher parallel processing capabilities is required.

In the other hand, since the predictive maintenance system will be implemented in a manufacturing plant, controller may be exposed for dust, temperature fluctuations and electromagnetic fields which could cause permanent damage for electronic components of the controller. Dust and other physical entities could be kept away from the controller by using an industrial grade enclosure but the damage they can be caused due to electromagnetic fields generated machinery and other kinds of sources cannot be kept away from the controller. When an electronic circuit was exposed for a electromagnetic field, it causes generating internal voltage spikes withing the electronic circuit if the electromagnetic field had enough power. Generated internal voltage spikes can be cause catastrophic failure of delicate internal electronic components.

In order to secure the electronic components of a circuit and to meet the industrial standards, we must add all the other electronic components and construct a protection circuit and protect the delicate electronic components of the circuit. Same thing applies for microcontrollers and all the other controllers when using them in industrial applications. It is possible to even use Arduino in industrial applications if protection circuits were included in the main circuit.

There are controllers with additional protective circuits inbuilt in the controller circuit specially designed for industrial applications such as Programmable Logic Controller (PLC), Single Board Computers (SBC), and Industrial Personal Computers (IPC).

3.17.1.3. Single board computers (SBC)

A single board computer is a complete computer with a microprocessor or microprocessors, RAM, ROM, input/output ports and other necessary components of a functional computer. These SBCs' are designed to meet the industrial standards. Same as Arduino microcontrollers, these can operate very easily since external peripherals could be connected and use them by plug-and-play method hence, SBCs' are becoming popular across the globe.

Because of the popularity, many companies are introducing their own SBCs' to the market so, there is a wide variety of SBCs with different capabilities. ASUS, Google and Nvidia are three companies which made three successful powerful SBCs.

- ASUS – Tinker Board
- Google – Coral Dev Board
- Nvidia – Jetson Nano

3.17.1.3.1 ASUS Tinker Board

ASUS Tinker Board is an SBC in a small form factor, but it can offer leading performance when it is compared with its class. It is compatible with many platforms hence IoT enthusiasts, hobbyists and DIY enthusiasts often use Tinker Board for their projects.

Figure 90 - ASUS Tinker Board

Features and Performance.

ASUS Tinker Board is a powerful single board computer which outperform many other single board computers from performance. It is powered with the Rockchip RK3288 processor which is a quad core powerful processor. Managing tasks, loading applications, and switching between tasks can be handled without any performance failure due to its 2GB LPDDR3 dual channel RAM also it a equipped with a SD 3.0 expandable microSD card so the operating system, installed applications and file storage will perform with a significant faster read and write speeds.

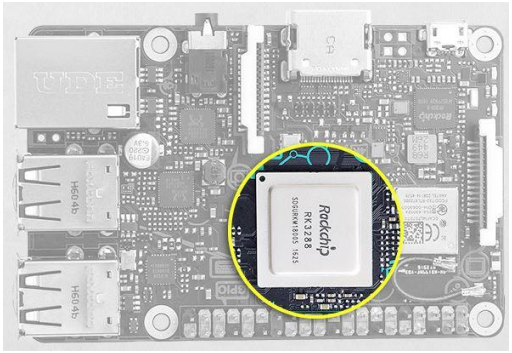


Figure 91 - GPU of ASUS Tinker Board

The Graphic Processor Unit (GPU) of the ASUS Tinker Board is an ARM-based Mali™-T764 GPU which is a powerful GPU and yet it is an energy efficient design. Since it consumes less power and normally CPU and GPU consume most of the energy consumption, this ARM-based GPU reduces the overall energy consumption of the SBC. The GPU allows wide range of

applications such as high-quality media playback, computer vision, gesture recognition, image stabilization and many more.

ASUS Tinker Board also offers maker friendly IoT connectivity. It is equipped with DSI MPI connection for displays and touchscreens which is great to display a dashboard for user interfaces in an IoT project for real-time data monitoring. Tinker Board is also capable of offering machine vision applications allowing secondary CSI MIPI connection to compatible cameras. It also ensures consistent Ethernet performance through a dedicated bus resource. To ensure minimal interference and improved radio performance, the Tinker Board offers shielded integrated Wi-Fi and Bluetooth controller.

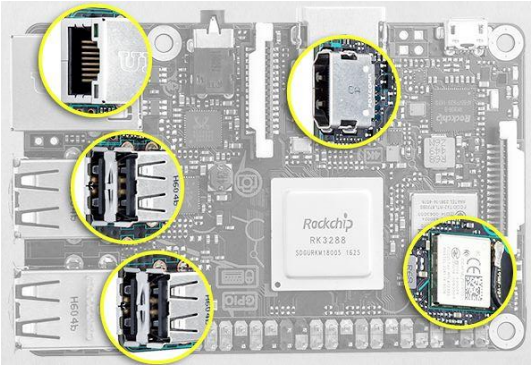


Figure 92 - Other Peripherals in ASUS Tinker Board

3.17.1.3.2 Google Coral Dev Board

Google Coral Dev Board is a powerful single board computer which can perform fast machine learning (ML) interfacing and other applications in a small form factor. It can also be used to prototype embedded systems, test the system, optimize the system, and then scale the prototype to the production using the on-board Coral System-on-Module (SoM) combined with a custom PCB hardware.

Figure 93 - Google Coral Dev Board

Features and Performance.

The removable System-on-Module of the Coral Dev Board contains eMMC, SOC, wireless radios, and the Edge TPU which provides a complete system to prototype IoT devices and other embedded systems that demands fast machine learning interfacing.

The CPU of the Coral Dev Board is NXP i.MX 8M SoC which is a quad Cortex-A53, Cortex-M4F chip. The specialty of The Coral Dev Board is that it is powered by another co-processor which aids machine learning programs to execute faster and efferently. This co-processor is a design of Google and while it enhances the performance of ML interfacing, it maintains the power consumption in a lower rate. This ML accelerator is called as “Google Edge TPU coprocessor: 4 TOPS (int8)”.

The GPU of The Coral Dev Board is an integrated GPU which is GC7000 Lite Graphics and it is equipped with a 8 GB eMMC flash memory and a microSD card slot to expand memory if needed. The RAM of the Coral Dev Board is 1 GB LPDDR4 RAM which is slightly under the average of the RAM capacity when it is compared with the other competitive single board computers. The Coral Dev Board with 4 GB LPDDR4 RAM variant is already announced and it will be available in the local market.

The Coral Dev Board offers user friendly connectivity through USB connectivity, audio connectivity and video connectivity ports. Specially there is an USB type-C power port to power the Coral Dev Board via a 5 V DC supply. For USB connectivity, it consists of a USB 3.0 type-C OTG port, USB 3.0 type-A host port and an USB 2.0 type B serial console port. For video connectivity, it consists of a full-size HDMI 2.0a port, 39-pin FFC connector for MIPI DSI display and 24-pin FFC connector for MIPI CSI 2 camera port so it easy to implement computer vision applications. There are peripheral devices specially designed for the Coral Dev Board by Google to enhance the performance and applicability (e.g. Coral Camera and Coral Environmental Sensor)

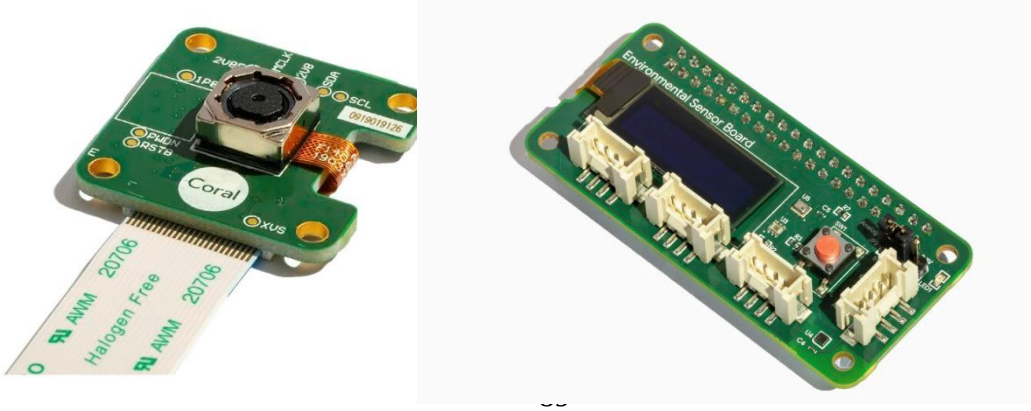


Figure 94 - Coral Sensor Modules

For audio connectivity the Coral Dev Board consists with a 3.5 mm audio jack, digital PDM microphone and a 2.54mm 4-pin terminal for stereo speakers.

3.17.1.3.3 Nvidia Jetson Nano

Nvidia Jetson Nano is a powerful single board computer which specially designed to run multiple neural networks in parallel for applications like image classification, segmentation, object detection, and speech processing. Nvidia delivers this powerful performance in a small form factor single board computer which only consumes 5 watts of power. This power efficiency allows the user to add a small battery pack or add more sensors or peripheral devices to the project and increase the flexibility of the project.

Figure 95 - Jetson Nano

Features and Performance.

Nvidia Jetson Nano is a very powerful single board computer which surpass most of the single board computers when it comes to performance. Specially, considering the performance of machine learning and deep learning applications, Jetson Nano is the best single board computer to its price.

It comes with I/O s ranging from GPIO to CSI and it makes developers easier to connect variety of sensors to enable variety of AI applications and increase the flexibility of the project. When the connectivity of the Jetson Nano is considered, it comes with USB 3.0 type A ports, USB 2.0 micro-B port, HDMI/Display port also gigabit ethernet connectivity. There are 2 MIPI-CSI camera connectors for stereo vision enabling variety of machine vision applications.

Nvidia company is a popular company when it comes to manufacturing most powerful GPUs for computers. GPU performance is a key feature for artificial intelligence applications. Jetson

Nano is equipped with NVIDIA Maxwell architecture GPU with 128 NVIDIA CUDA cores. This is the main reason that the Jetson Nano is a special single board computer among all the other competitive single board computers.

CPU and GPU are both processing units, but both are specialized in handling different computational types. CPU is a processor which is capable of handling "General Computations" better while the GPU is a processor which is capable of handling "Specialized Computations" better. When it comes to parallel processing, GPU is more capable of handling parallel tasks than a CPU.

Parallel computing is a computational method which separate a particular main computation into several number of smaller computations. Those smaller computations will be carried out simultaneously. Then calculated results of the smaller computations will be recombined to form the result of the main computation. The number of smaller tasks that the main task can be broken into depends on the number of cores on the processor. Cores are the units that computations take place on the processor.

Usually CPUs have four, eight and sixteen cores while GPUs of personal computers have thousands. This proves that GPUs are better to handle parallel processing applications. In parallel computing, an embarrassingly parallel or perfectly parallel task is one where almost no effort is required to separate the main task into several smaller tasks. These smaller tasks are independent from each smaller task. This same scenario is happening in neural networks so neural networks are considered as embarrassingly parallel or perfectly parallel. Many of the operations of neural networks can be divided into smaller sub-tasks and can be executed parallelly and simultaneously and then can be join all the results to form the result of the main computation.

This is why Compute Unifield Device Architecture (CUDA) cores in the GPU of the Jetson Nano are important. NVIDIA company has designed CUDA as a software platform within their GPU hardware to make developers work easier to create software that accelerates computations using the parallel processing power of their GPUs. Their GPU hardware is what enables the ability of parallel computing while CUDA is the software layer which provides an API for developers.

Jetson Nano is also equipped with NVIDIA JetPack™ which consists of board support package, NVIDIA CUDA, Linux OS, and TesorRT™ software libraries for many applications

such as deep learning, computer vision, multimedia processing, GPU computing and many other applications.

3.17.1.4 Industrial Personal Computers (IPC)

An industrial personal computer or an IPC is a specially designed rugged computer to meet the demanding environmental conditions that requires to perform in harsh industrial conditions without any performance losses. These demanding environmental conditions are, protection from liquids and dust, extreme temperatures, high shock, and vibration.

Other than these conditions, when there is heavy machinery that consume high energy with heavy motors and other electrical components which can generate electromagnetic fields, electronic components with delicate electronic components are vulnerable to these electromagnetic fields. These electromagnetic fields could generate voltage spikes withing the circuitry and burn the delicate electronic components. That is why developer boards like Arduino cannot be used in industrial environments. IPCs are designed considering all the above conditions and many other to enhance the performance delivery of the IPC disregarding the environmental conditions.

Compared with other client systems, IPCs offer greater customization according to the requirement, reliability of the system, scalability, and the longer product life cycle also IPCs can also be compatible with other platforms such as computer vision systems. Generally, IPCs are designed to be high energy efficient. Since higher energy consumption like in normal personal computers are not present in an IPC many IPCs are designed to be fanless. It reduces the size of the IPC and almost every IPC comes in a smaller form factor so that its ideal for space-constrained environment.

3.17.1.4.1 Xingtac MPC-2018 Industrial PC

Xingtac is a company which provides customization services according to their customer requirements. They develop their own brand Xingtac Industrial panel PCs, embedded PCs, and industrial monitors.

Xingtac MPC-2018 an industrial embedded PC designed by them according to the requirement of an industrial standard qualified IPC. It is a powerful computer in a smaller form factor, and it is powered by a DC power supply.

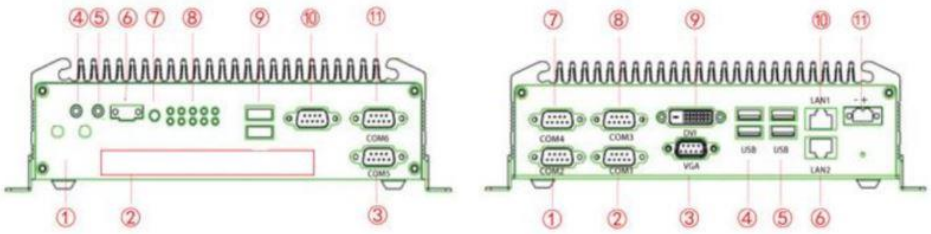
Figure 96 - Industrial PC

This IPC is powered by an Intel Brodwell U series processor (Intel Core TM i5-5200U, 2.2 GHz) which is an energy efficient CPU. The GPU is an integrated graphic card which is Intel HD4000/HD5000 series graphic card. Both the CPU and the GPU can handle a handling a heavy AI application without losing the performance. It also comes with a 4 GB DDR3L RAM and it can be upgraded all the way up to 16 GB and there is a 64GB SATA HDD storage drive.

There are gigabit ethernet ports, two RS-232/485 interfaces, four RD-232 interfaces, four USB 3.0 ports and two USB 2.0 ports. Inside the IPC there are 12 GPIOs (3.3V, 24mA) which are not accessible directly. To access the GPIOs the outer casing should be removed.

The max power consumption is rated as 18W and it weighs 2.5 kg and dimensions are 243x135x66 mm. It is safe to operate this IPC in a range of -20 ^{0}C to 70 ^{0}C and it can be stored in a range of -30 ^{0}C to 80 ^{0}C temperature range.

I/O Ports:



1.Antenna 2. Extended area for serial port and parallel port 3. COM5 4. Line-out 5. Mic-in 6. PB 7. Power button 8.LED indicators 9. 2xUSB 10. GPIO 11. COM6

1.COM2 2. COM1 3. VGA 4. 2xUSB 5. 2xUSB 6. LAN2 7. COM4 8. COM3 9. DVI 10. LAN1 11. Power

Figure 97 - I/O Configuration of IPC

All these features confirm that it is a rugged industrial PC which is capable of survive in most of the harsh environmental conditions that may occur in an industrial workplace.

3.17.2 Comparing Selected Single Board Computers

ASUS Tinker board, Google Coral Dev Board and NVIDIA Jetson Nano are three most powerful single board computers in the market. Each has good CPU and GPU performance when comparing them to other single board computers. To select the best out of these three SBCs other features also should be considered other than the CPU and GPU performance.

The main processing task which the selected SBC should be able compute during the execution of the program of this project is a machine learning task to predict the failure of a bearing in the Bielomatik machine. Because of that comparison is done mainly focusing on the CPU and GPU performance along with the RAM, storage, energy efficiency, price, and the availability.

	Tinker Board ✖ ASUS	Jetson Nano ✖ Nvidia	Coral Dev Board ✖ Google
			
SoC	Rockchip RK3288	Nvidia Erista	NXP i.MX 8M
Primary processor specs	ARM Cortex-A17 (32-bit) 1.8GHz quad core	ARM Cortex-A57 MPCore (64-bit) 1.43GHz quad core	ARM Cortex-A53 (64-bit) 1.5GHz quad core
GPU	Mali-T764	NVIDIA Maxwell with 128x CUDA cores	Vivan GC7000 Lite
RAM size	2GB	4GB	1GB
Built-in RAM	✔	✔	✔
Internal storage	✖	16GB	8GB
Removable storage	MicroSD	Other format	MicroSD

Figure 98 - Comparison of SBCs

3.17.2.1 CPU comparison

- Asus Tinker Board - Rockchip Quad-Core RK3288 processor
- Google Coral Dev Board - NXP i.MX 8M SoC
- NVIDIA Jetson Nano - Quad-core ARM A57

The CPU what Google offers with their Coral Dev Board performs basically like the Raspberry Pi single board computer. It is a 64-bit ARM Cortex A53 processor which is a low-end mobile processor. The Rockchip RK3288 SoC of the ASUS Thinker Board is equipped with a 32-bit ARM Cortex A-17 processor which is a mid-range mobile processor while the NVIDIA Jetson Nano offers a 64-bit ARM Cortex A57 MP Core processor. Out of all these CPUs, Jetson Nano offers the best CPU, and it is a high-end mobile/server processor.

CPU IP	Target	Estimated DMIPS/MHz	big.LITTLE	Shipping in Devices/Systems
Cortex A57	High-end mobile/servers	5*	Yes (w/ A53)	2015
Cortex A53	Low-end mobile	2.3	Yes, LITTLE, w/ A57	2H 2014
Cortex A17	Mid-range mobile	4.0*	Yes, big, w/ A7	Early 2015

Figure 99 - CPU Comparison of SBCs

3.17.2.2 GPU Comparison

ASUS tinker board is equipped with integrated graphics processor which is ARM® Mali™-T764 GPU*1. It is not designed especially for machine learning applications as Googles Coral Dev and NVIDIA Jetson Nano. Google Coral Dev has integrated GC700 lite graphics GPU which is a powerful GPU and it also equipped with a special ML accelerator; Google Edge TPU coprocessor. It can perform four trillion operations per second. NVIDIA Jetson Nano is powered by a NVIDIA Maxwell architecture with NVIDIA CUDA core GPU. These CUDA cores allows the Jetson Nano to outperform the almost every single board computer with its amazing AI applications execution performance.

3.17.2.3 Other Comparisons

- ASUS Tinker Board and Google Coral Dev both comes with Wi-Fi and Bluetooth, but NVIDIA Jetson Nano does not offer Wi-Fi withing itself. But it is possible connect a external Wi-Fi card to enable the feature.
- RAM comparison
  - ASUS Tinker Board - 2 GB
  - Google Coral Dev Board - 4 GB

- NVIDIA Jetson Nano - 1 GB

- Storage comparison
  - ASUS Tinker Board - None (only external microSD card)
  - Google Coral Dev Board - 8 GB
  - NVIDIA Jetson Nano - 16 GB

- Price comparison
  - ASUS Tinker Board - $ 70
  - Google Coral Dev Board - $ 149
  - NVIDIA Jetson Nano - $ 99

Taking all the facts into consideration, it is possible to choose Jetson Nano is the most suitable single board computer for this project. It has the better CPU performance, better GPU performance that needs to execute the machine learning program and it has more RAM and storage. Only drawback is that it lacks a Wi-Fi connectivity, but that issue could easily be solved by connecting an external Wi-Fi card to the Jetson Nano.

3.17.2.4 Availability

ASUS Tinker Board is available in the Sri Lankan market, but Google Coral Dev and the Jetson Nano boards are not available in the market. Those has to be purchased online from an online store. Due to the Covid-19 pandemic all the shipments are delayed and will take roughly around 6 to 8 weeks to deliver.

Despite the availability, NVIDIA Jetson Nano is the most suitable single board computer for this project. Since it is not possible to buy a Jetson Nano, it is needed to choose an alternating controller. Two controllers left to choose between are the ASUS Tinker board and the Industrial PC. In this situation, performance was taken into consideration despite other facts because to test the system, a controller with good performance is needed. IPC is considerably larger than a single board computer, so it was decided to configure the system using the IPC and test and optimize the system. Later, as a further improvement for the project, a Jetson Nano will be purchased and re-configure the system and optimize further.

3.18. Designing an Uninterruptible Power Supply for the Raspberry Pi

3.18.1. Introduction

What is an Uninterruptible Power Supply?

Uninterruptible power supply, commonly known as UPS is an electrical apparatus which can provide emergency electrical power to a load in case of failure of the main power supply. A UPS differs from an auxiliary or emergency power supply from the ability to provide instantaneous backup power to the load which auxiliary or emergency power supplies will take at least few seconds or sometime few minutes. Usually, uninterruptible power supplies are designed to provide backup electrical power to the load for a short duration (usually for few minutes) to back up the current and processed data.

Why is a UPS needed for this system?

The single board computer (SBC) used in this system; the Raspberry Pi 4, runs the machine learning code and process the data acquired from sensors. If any power failure occurred, the current progress of the program would be lost along with the data acquired and unsaved processed data. Despite the loss of data, if a power failure occurs, current program of the SBC would be interrupted, and it may force to reset the program over. This could be prevented by simply keeping the SBC online using a UPS during a power failure until the backup generator of the plant to take over.

3.18.2 Minimum requirement

In case of a power failure in the manufacturing plant, it will take 2 minutes to automatically start generators, so the minimum requirement of the UPS is to provide backup power to the Raspberry Pi at least for 2 minutes to save the progress and to keep it online.

However, if we need to change the source code of the program or to do any changes to the Raspberry Pi, it is needed to keep the SBC powered for several minutes or maybe for an hour or two. Otherwise, every time any change to the SBC to be done, the whole system should be powered up, so it is important to design a UPS that can power the SBC for 2 hours.

3.18.3 Calculations

Raspberry Pi 4 B has a more powerful CPU and a GPU comparing to other Raspberry Pi models, so it is expected to consume more power. Raspberry Pi 4 B operates in 5V and it is recommended to use a power adapter which can deliver up to 3 amperes.

At idle, Pi 4 B is expected to draw 3.4 watts which is 17% more power consumption than its predecessor Raspberry Pi 3 B+. Under load, it is expected to increase the power consumption up to around 7.6 watts. Since it is a machine learning code executes as the source code here, it

will push the Pi 4 B to its limits. Hence, for calculations of the UPS, it is important to use the maximum power ratings of the Raspberry Pi 4 B.

To decide the battery capacity for the UPS, following factors should be considered:

- Maximum power drawn by the Raspberry Pi
- The duration that the UPS should power the Raspberry Pi

As previously mentioned, Raspberry Pi will draw 3A when it is in the maximum load at 5V. By using the power equation, maximum power drawn by the Raspberry Pi at its maximum load can be calculated,

$$P = VI$$

$$P = 5\,V * 3\,A$$

$$P = 15\,W$$

Considering the UPS may need to power the Raspberry Pi for 2 hours without charging the batteries,

$$Energy\ capacity\ of\ battery\ cell\ (E) = P * No\ of\ hours$$

$$E = 15\,W * 2\,h$$

$$E = 30\,Wh$$

Battery capacity is given in milli ampere hours (mAh), so 30Wh should be converted to mAh unit by using the following equation,

$$Q(mAh) = \frac{1000 * E\ (Wh)}{V\ (v)}$$

$$Q = \frac{1000 * 30\,Wh}{5v}$$

$$Q = 6000\,mAh$$

Battery cell with not less than 6000mAh should be available in the UPS.

3.18.4 Market Research - UPS

1. **Raspberry Pi Power Management and UPS HAT**

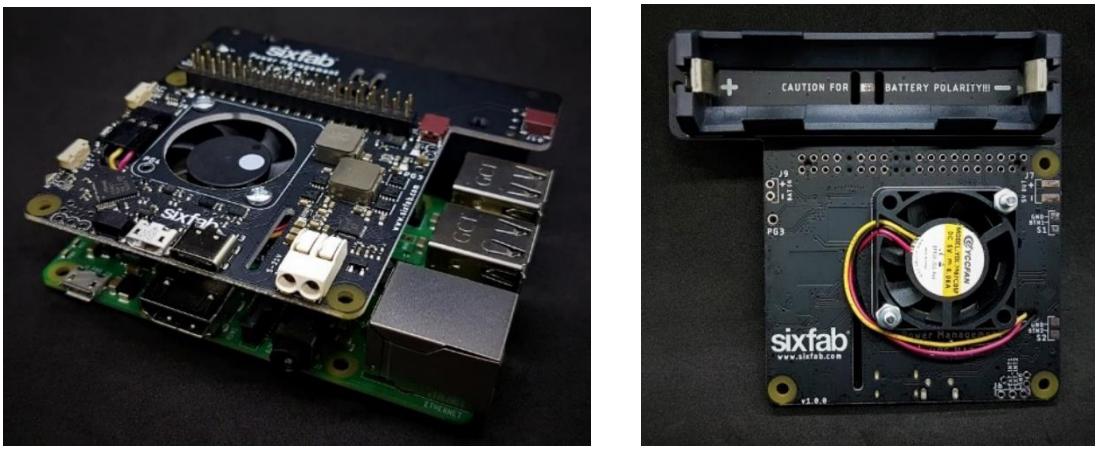


Figure 100 - Raspberry Pi Power Management and UPS HAT

Power management and UPS HAT designed by sixfab is a product designed to solve the problem of power supply, a persistent challenge for most projects that use Raspberry pi as the SBC. Out of many power supply solutions out in the market, this product comes highly rated and can address many of the critical challenges including real-time monitoring the of UPS HAT along with the Raspberry Pi remotely. This could be achieved through configurating the UPS HAT with their cloud which is "Sixfab Power Software". Then it is possible to monitor the device remotely. This can be purchased from their website for $69 (product price, excluding the shipping cost), hence in Sri Lankan rupees the current price (05.11.2020) would be Rs.12700.

Diagnostics and features of the power software:

- Charging status battery percentage, and battery health.
- Main power source plugged or unplugged.
- Input/Output voltage level and current consumption.
- Temperature of Raspberry Pi, HAT, and Battery holder board.
- Cooling fan working status and RPM

Features:

- Input, system, and battery current monitoring.
- Battery protection circuit enhances with integrated temperature sensors.
- Thermal Shutdown and input, system, battery overvoltage protection.
- Smart cooling fan with pulse signal feedback.

- Battery level indicator LEDs with charging/discharging status.

2. Uninterrupted Power Supply UPS HAT for Raspberry Pi

Figure 101 - 2. Uninterrupted Power Supply UPS HAT for Raspberry Pi

This power supply is also designed to power the Raspberry Pi seamlessly and can charge the batteries while powering the SBC. It can be connected to from the top of Raspberry pi it provides full access to all 40 GPIO pins of the board by extending them. This power supply is also packed with invaluable features that can be used to monitor the UPS as well as the Raspberry Pi in real time. This can be purchased from their website for $19.99 (product price, excluding the shipping cost), hence in Sri Lankan rupees the current price (05.11.2020) would be Rs. 3700.

Features:

- I2C bus communication, monitoring the battery voltage, current, power, and remaining capacity in real time.
- Multi battery protection circuits: over charge/discharge protection, over current protection, short circuit protection and reverse protection.
- Onboard 5V regulator, up to 2.5A continuous output current.
- Battery warning indicators.

3. PiJuice HAT

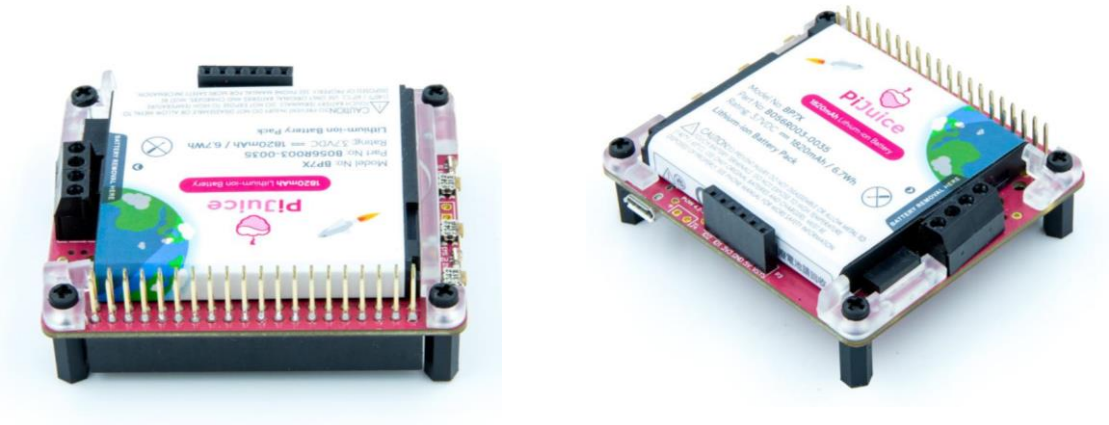


Figure 102 -PiJuice HAT

PiJuice is a compact UPS HAT that allows the Raspberry Pi to be more portable and easier to use. There is an onboard real time clock (RTC) along with an integrated microcontroller which will manage soft shut down functionality and true low power deep sleep state.

By default, PiJuice comes with a BP7X Li-ion battery with 1820mAh battery which can power the Raspberry pi up to 30 to 90 minutes, if additional running time is needed it is possible to buy high-capacity Li-ion batteries along with the PiJuice. PiJuice uses five GPIO pins (power and I2C) of the Raspberry Pi and other pins are easily accessible since all of the GPIO pins are extended. This can be purchased from their website for £47.99 GBP (product price, excluding shipping cost) hence, in Sri Lankan rupees the current price would be Rs. 11200. If any additional batteries are needed, prices are as follows,

- PiJuice 5000mAh battery → £ 25 (Rs. 6000)
- PiJuice 12000mAh battery → £ 30 (Rs. 7300)

Features:

- Integrated Real Time Clock
- Onboard intelligent on/off switch
- Low power deep-sleep state with wake on interrupt/calendar event
- Programmable multi-colored RGB led (x2) and buttons (x3) with super simple user-configurable options.

- Full power management API available to Raspberry Pi OS with auto shutdown capability when running low on batteries.
- Charge via on-board micro-USB or via the Raspberry Pi micro-USB

Considering the UPS requirement for this system, it is clear that not all the functions of above products are needed. Only essential key features are needed, and all those functions added more value to their products, so the prices are higher except the UPS by WaveShare which costs Rs. 3700. Still, any of these power supplies are available in the Sri Lankan market. To acquire them, power supplies should be purchased through their websites and may take at least one to two months for delivery.

Considering the requirements, an UPS with all the requirements for the system can be design and fabricated using locally available electronic components at a price not exceeding Rs. 2300 so the best thing is to design a UPS as for our requirements. For further improvements for the system are needed, any of those UPS HATs can be purchased and can upgrade the system.

3.18.5 Component Selection

Selection of components is done considering the following factors.

- Requirements of the UPS
- Safety and quality
- Availability
- Price

As the first component, the battery should be selected since the selection of other components should be done according to the battery cell.

3.18.6 Selecting the battery.

When selecting a battery, it is important to consider the parameters involved in operation of batteries. Since there is no perfect battery that can be used for any kind of projects, each battery type performs differently according to the application it is being used. Most importantly, performance of the system depends on the source of power, so the battery must be chosen precisely. Following parameters were taken into consideration while choosing the battery.

- Battery Voltage

- Battery Capacity
- Safety
- Cost
- Availability

3.18.6.1 Voltage

The battery voltage is one of the most important parameters needs to be considered when selecting a battery. Higher voltage than a selected battery cell can be obtained by stacking two or more batteries in series, so more batteries may need to acquire the required voltage if the correct battery was not chosen.

The Zinc-Carbon battery and Nickel-metal hydride batteries delivers a nominal voltage of 1.2V to 2V whereas the lithium-based batteries can deliver a nominal voltage of 3.2V to 4V. If higher voltage is needed, lead-acid batteries can deliver a nominal voltage of 12V. Since the Raspberry Pi operates in 5V, it is better to use two lithium-based batteries in series or use a one cell and step-up the voltage up to 5V using a boost converter.

3.18.6.2 Battery Capacity

Battery capacity determines the runtime of the electronic component that the battery can offer. The power or the capacity of the battery is expressed in Watt-hours (Wh) or many commercial batteries are expressed in ampere-hour rating (Ah) or milli ampere-hour rating (mAh). According to the calculations, to maintain two hours of runtime for the Raspberry Pi, a battery with not less than 6000mAh of capacity is needed. Lithium based battery cells with higher capacity are common in the market and can be found in many types of batteries. Most common lithium-based batteries are Li-ion batteries and Lipo (Lithium Polymer) batteries.

3.18.6.3 Safety

Safety of the battery is an important parameter to consider when selecting a battery since if a battery fails it could cause fire or may explode and can damage the entire system. Also, the chemistry used in battery decides the toxicity of the battery when it is disposed. Nickel-Cadmium and Led acid batteries are toxic and should not be disposed to the environment so, they should be recycled. Li-ion and Lipo batteries also contains toxic chemicals but in very small amount per battery. They are not harmful as Nickel-Cadmium or Led acid batteries but Li-ion and Lipo batteries also should not be disposed to the environment.

Lithium based batteries comes in different forms. Most common types in the market are Pouch cell and Cylindrical Cell. They can handle external forces acts on the cells differently because of the different physical strengths of these cells.

Figure 103 - Battery options

Physical damages to Li-ion or Lipo batteries can cause fire and may explode as well if the state-of-charge (SOC) of the battery was higher. In electro chemical batteries, there is a separator which separates electrolytes in negative and positive electrodes. Any physical damage with considerable amount of force can damage this separator. It allows the electrolyte in both electrodes to infuse and cause internal short circuit. Lithium is a highly reactive metal and if its exposed to air, lithium undergoes a rapid reaction and dissipate energy as heat. If the battery got punched by a sharp tool battery could be exploded. In this case Pouch cell batteries can easily get punched if it was not handled carefully but cylindrical cell batteries can handle a considerable amount of force.

Figure 104 - Rigidity of Cylindrical Cells

3.18.6.4 Cost and Availability

Cost of pouch cell batteries are much higher than cylindrical cell batteries. A pouch cell Lipo battery with 5000mAh capacity will cost more than Rs8000. Pouch cell battery Li-ion cells and 12V Li-ion battery pack with same capacity will cost more than Rs 7000. Cylindrical cell Li-ion batteries are much cheaper and can be found in higher capacity batteries in smaller footprint. 3.7V , 5800mAh Li-ion 18650 batteries can be found in the market just for Rs500. 18650 is very common type of Li-ion battery and is being used as laptop batteries for many years.

Figure 105 - Li-ion 18650

Considering all the above parameters, 18650 Li-ion battery can be selected as the safest and the bast value to the price. So, 18650 battery is selected to be used in the UPS. The requirement of the UPS is an output of 5V and a minimum capacity of 6000mAh. Both requirements cannot be satisfied with one 18650 battery so at least 2 batteries are needed.

To increase the voltage, 2 battery cells should be connected in series, but 6000mAh capacity cannot be achieved since connecting 2 battery cells in series does not affect the capacity. The capacity will be equal to the capacity of one battery cell.

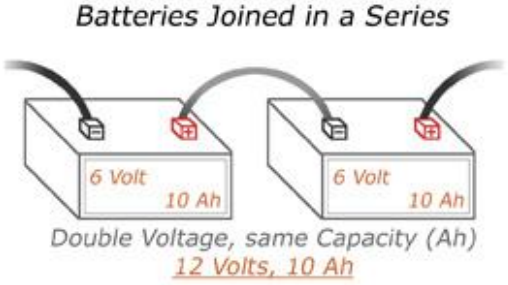


Figure 106 - Batteries Joined in series

The capacity will be doubled if two cells were connected in parallel but cannot achieve the 5V output.

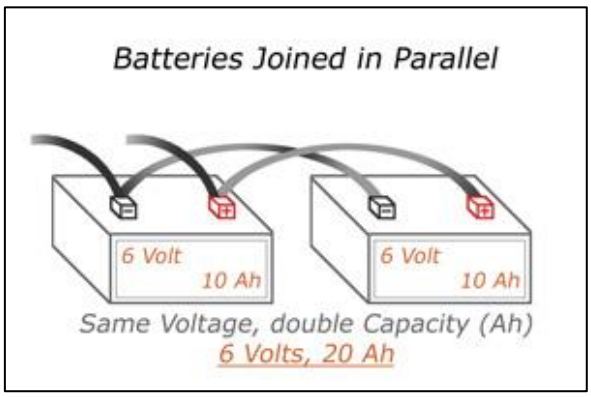


Figure 107 - Batteries joined in parallel.

Only way to increase the capacity is to add more batteries, but only way to increase the voltage is not adding more batteries. It is possible to boost the 3.7V voltage up to 5V and obtain a stable voltage using a step-up voltage regulator also known as boost converter.

3.18.7 Selecting the Boost Converter

When selecting the boost converter, it should be considered following parameters.

- Output voltage (5V)
- Output current (3A)
- Circuit protection

XL6009 DC-DC Buck Boost Converter Module is capable of driving a 4A load with good line and load regulation. The main switching component, XL6009 IC is an efficient switching regulator which can provide a variable voltage output (5V to 35V) and output efficiency is significantly higher.

Figure 108 - Buck - Booster

The footprint of the circuit board is also smaller and space-saving. It comes with short circuit protection but there is no protection for reverse connection so, connecting it to the battery must be done carefully.

3.18.8 Selecting the Charging Circuit

It was decided to use two 18650 battery cells in parallel and form a single battery cell. If those two batteries were connected in series, it forms two battery cells. Multiple cell battery requires a balance charging circuit for charging. Since there is only one cell, regular charging circuit can be used here.

Li-ion and Li-Po batteries requires a constant current and voltage to charge. If the voltage or the current is not stable or as required, it can cause permanent damage to the battery cells. Little overcharge or over discharge can cause destruction of the battery so selecting the charging circuit must be done carefully.

When selecting the charging circuit, following parameters should be considered.

- Charging voltage
- Charging current
- Circuit protection

TP4056 battery charging circuit is a cheap module which is common in the market. It is a Li-ion battery charger for a single cell battery. It consists of several important features along with circuit protection. It also includes two status outputs indicating charging in progress and charging completion. The working input voltage is 4V to 8V and can charge a single battery cell in 1A current rate.

Figure 109 - Charger Circuit

Features of the TP4056 Charging Circuit:

- Overcurrent protection
- Overcharge protection (cutoff the connection when fully charged)
- Temperature protection (Disconnect the charger if the temperature of the battery increased higher than normal)
- Soft charging
- Output charging current is adjustable.
- Automatic recharge (keeps batteries optimally charged when connected to a charger)

3.18.9 Schematic diagram

A schematic diagram is a picture of a setup that shows the whole setup in a simple way using symbols which makes it easy to understand and makes changes if necessary. It only describes the significant and most important components of the system, though some details in the diagram may also be exaggerated or introduced to facilitate the understanding of the system. The complete wiring diagram will be not included in the diagram, but the line diagram will be included.

When developing an electronic system, schematic diagram plays a major role since it shows the electrical connections and functions of a specific circuit arrangement by the means of graphic symbols. After designing, during the initial steps of the fabrication process, the schematic diagram is used to trace the circuit and the location of the components. The circuit and the entire components placement can be done following the schematic diagram, so creating a schematic diagram is an important task.

The schematic diagram for the UPS is created using the DS Electricals software which is a similar software as the SolidWorks Electrical, but it is a free software.

3.18.9.1 Schematic diagram of the UPS circuit

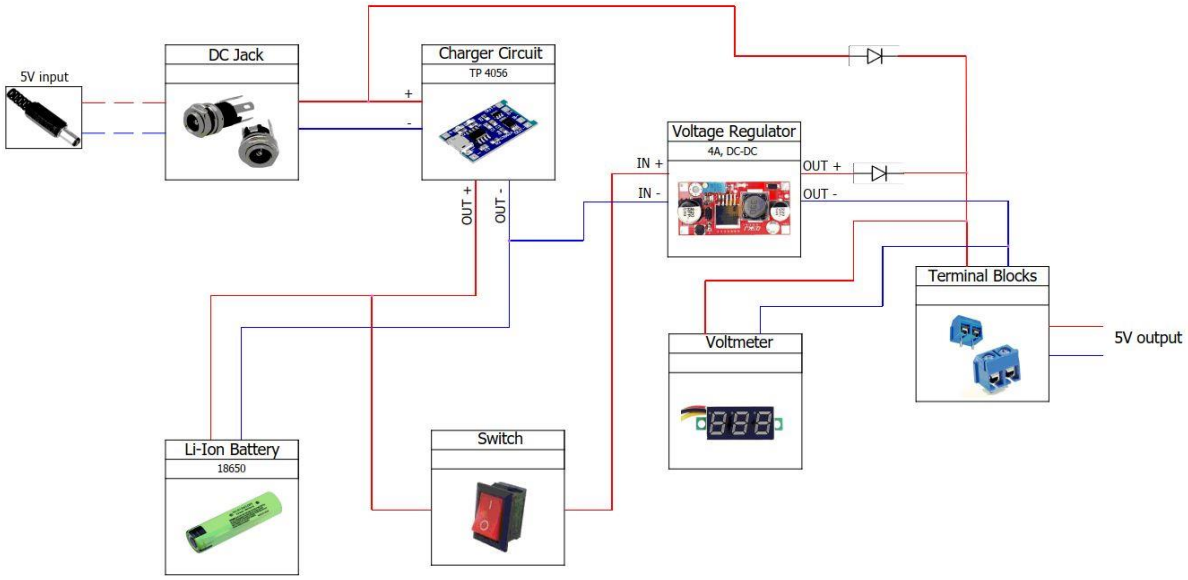


Figure 110 - Schematic Diagram of the UPS Module

- Input voltage to the UPS system will be supplied from a power supply unit in the control panel. 5V input from the power supply is connected to the UPS through a DC jack.
- A DC rocker switch is used to disconnect batteries from the boost converter, so it is possible to switch-on and switch-off the UPS manually.
- Mini voltage meter module is used to measure the output voltage of the UPS. It makes easy to calibrate the boost converter to 5V since it has not inbuilt display to read the output voltage. It is also important to monitor the output voltage of the UPS real-time. If any change in the output voltage can be detected and calibrate it again to 5V. this is important to provide a stable output of 5V to the Raspberry Pi.
- Terminal blocks which are used to distribute the output are connected to the output of the boost converter as well as to the input 5V connection, so Raspberry Pi is powered through the external power supply which is in the control panel when there is no power failure in the grid. At this time batteries will be charged if needed but batteries will not provide current to the Raspberry Pi. Whenever a power failure in the grid occurs, Raspberry Pi will be powered through the batteries. During the power failure, since batteries are connected to the external power supply through the output of the boost converter, the battery current can be drained through that path. To prevent that from happening, two diodes (1N5822) are being used.

3.18.10 3D model

The final stage is to create a 3D model of the UPS. There are several advantages of creating a 3D model before fabrication.

- Better for project approvals
- East to re-model and corrections
- Less rework
- Increased productivity

The main two reasons to create a 3D model for the UPS other than the above advantages are,

- Create an enclosure to the UPS.
- Finalize the overall layout.

3.18.10.1 UPS Circuit

The size of the dot board and the positioning of the drill holes can be finalized before the fabrication starts by modeling a 3D model. Also, having the component layout of the circuit will makes the process more efficient.

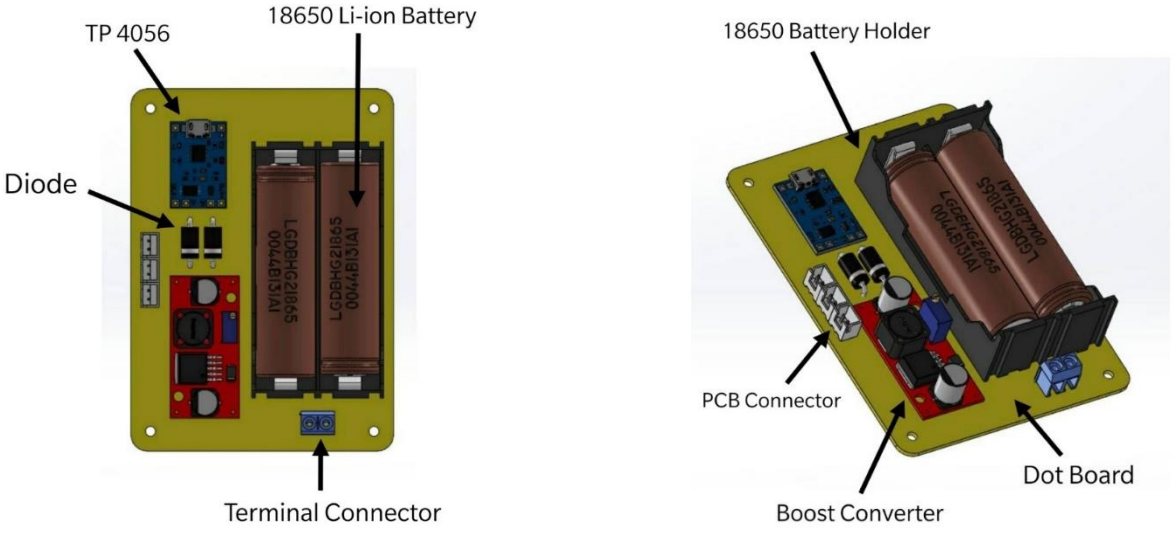


Figure 111 – 3D Model UPS Circuit

3.18.10.2 Enclosure

The enclosure of the UPS will be an Acrylic Tab and Slot Enclosure. This can be fabricated using different types of acrylic sheets with different thicknesses.

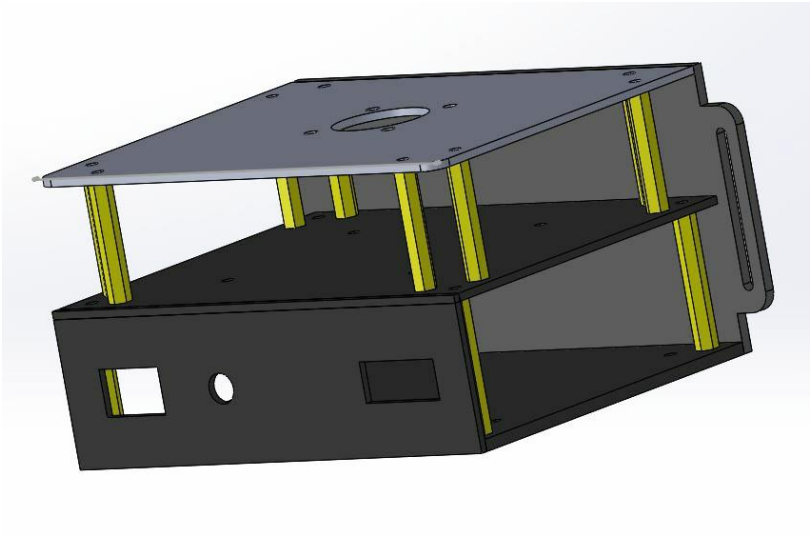


Figure 112 - Enclosure for UPS module and SBC

The 3D model of the enclosure is designed according to the dimensions of all the other components as the initial step, then tabs and slots can auto generate using the tab and slot tool of SolidWorks.

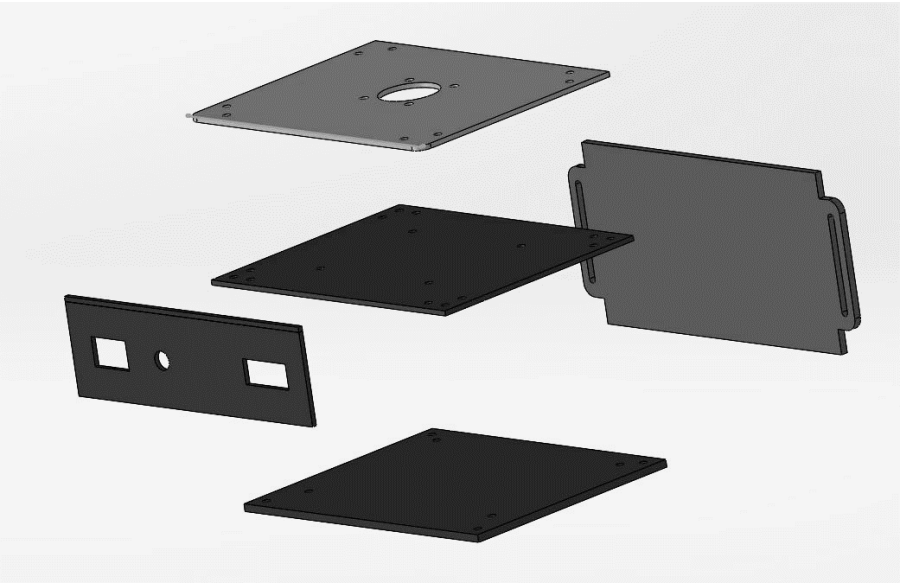


Figure 113 - Expanded model of enclosure

Then each and every part of the enclosure box is saved in the format of DXF. DXF is the file format which is readable in Laser Cutting machines. Then all the parts can be cut using required acrylic sheet. Final step is to glue all tabs and slots to complete the UPS enclosure.

3.18.10.3 Complete 3D Model

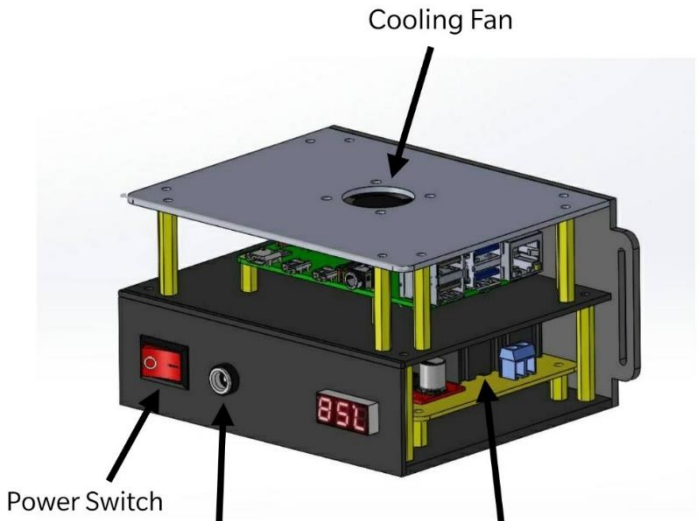


Figure 114 - Complete model of SBC enclosure I

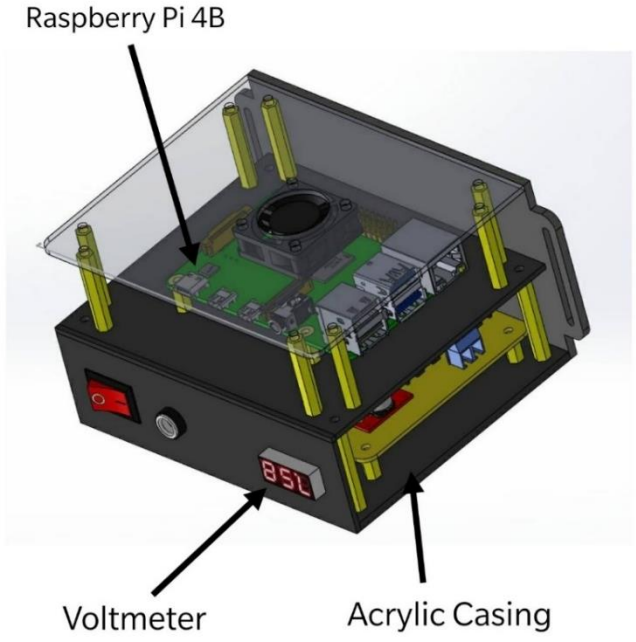


Figure 115- Complete model of SBC enclosure II

3.19 Control Box Design

3.19.1 Introduction

What is a control box?

Industrial equipment and machines require defined functions and orderly control to achieve their various process goals and electrical control panels are used to achieve this.
Electrical Control Panel is a combination of electrical components which are dedicated to achieving various mechanical functions of industrial equipment and machines. Panel structure and electrical components are the two main categories in an electrical control panel. Electrical components are selected according to the requirements of the industrial equipment and machines controlled by the electrical control panel. The structure of an electrical control panel is consisting of two main parts: the enclosure and the back panel. The enclosure is the box of the electrical control panel. Typically, industrial enclosures are made of aluminum or stainless steel with a powder coat layer. Plastic enclosures are also available in the market. Material of the enclosure should be decided according to the application, safety rating and properties.

- Indoor/outdoor use
- Waterproof/water resistance
- Dust/solid contaminants proofing.
- Hazardous conditions rating
- Explosion proof rating

Back panel is a metal, or an acrylic sheet mounted inside the enclosure. It provides the structural support for DIN rail mountings and wire ducts. DIN rails are used to mount all electrical devices and wire ducts are used for routing and organization of wires.

Figure 116 - A typical control box I

Figure 117- A typical control box I

3.19.2 The requirements of an Electrical Control Panel

The Bielomatik P490 is the main machine in the Atlas Axillia manufacturing plant which prints exercise books. This predictive maintenance system will be used to monitor the Bielomatik P490 machine so this system will be using in industrial conditions.

The enclosure box should protect its internal electrical components from dust, solid contaminants so it should be a dustproof enclosure. Also protecting electrical components from oil water splashes is a requirement. Since the system will only use in indoors the enclosure box does not required to be waterproof but it should be water resistance since oil or water splashes during maintenance can be damage the internal electrical components.

The enclosure must withstand any physical damages that can be caused by external impacts. Hence, a plastic enclosure will not be suitable for such conditions.

Considering the above-mentioned ratings and properties, most suitable enclosure will be an aluminum enclosure with a powder coat layer.

Another requirement of the Control box is not to keep any controlling methods (i.e., reset push button, power ON/OFF switch etc.) outside of the enclosure so any changes to the system can only be done by the machine operator who is responsible for this system.

The Bielomatik P490 is a large-scale machine so the critical points which are selected to be monitored by this system are located in several meters away. Standard control panels are mounted on the particular machine or near the machine. It is possible to mount this system near the Bielomatik machine and mount the vibration sensors permanently in critical points and then connect them to the control panel. If so, this system will be limited to the Bielomatik machine. Since there is another Bielomatik machine in the Atlas manufacturing plant and several other large-scale machines which this system could be used for prediction maintenance, this system is designed to be portable.

3.19.3 Schematic diagram

3.19.3.1 Design Spark Electrical Software

Design Spark Electrical is an Electrical CAD software which allows the user to design the line diagram, power circuit diagram, control circuit diagram and other required documents that needs to document the control panel as a complete project.

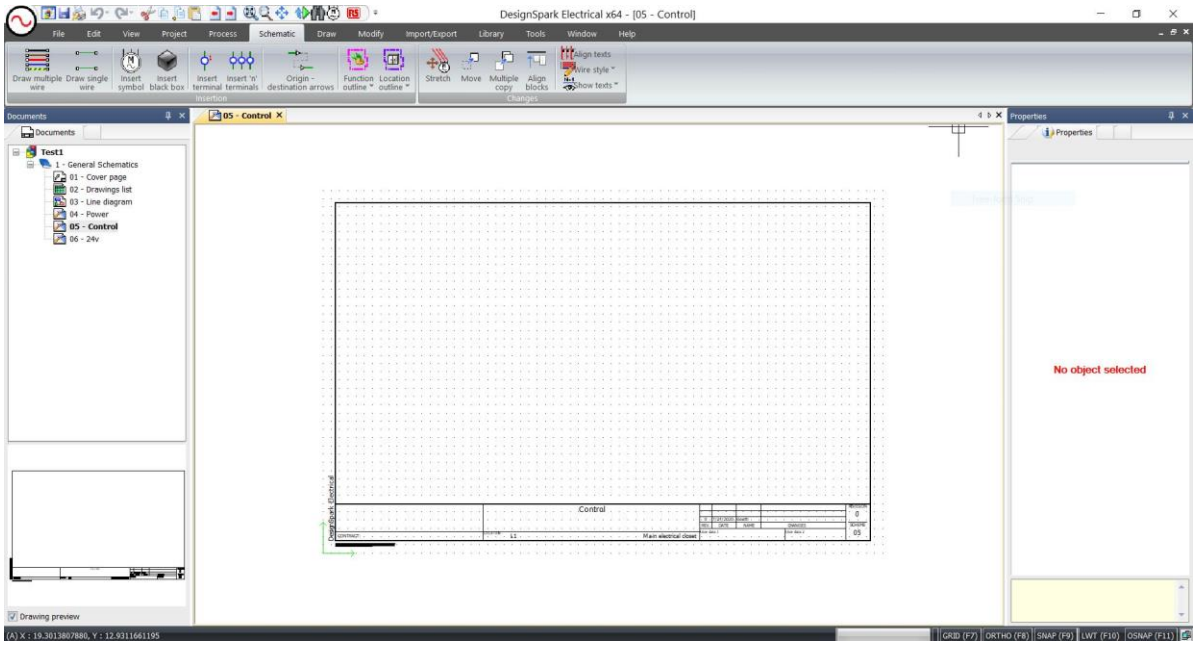
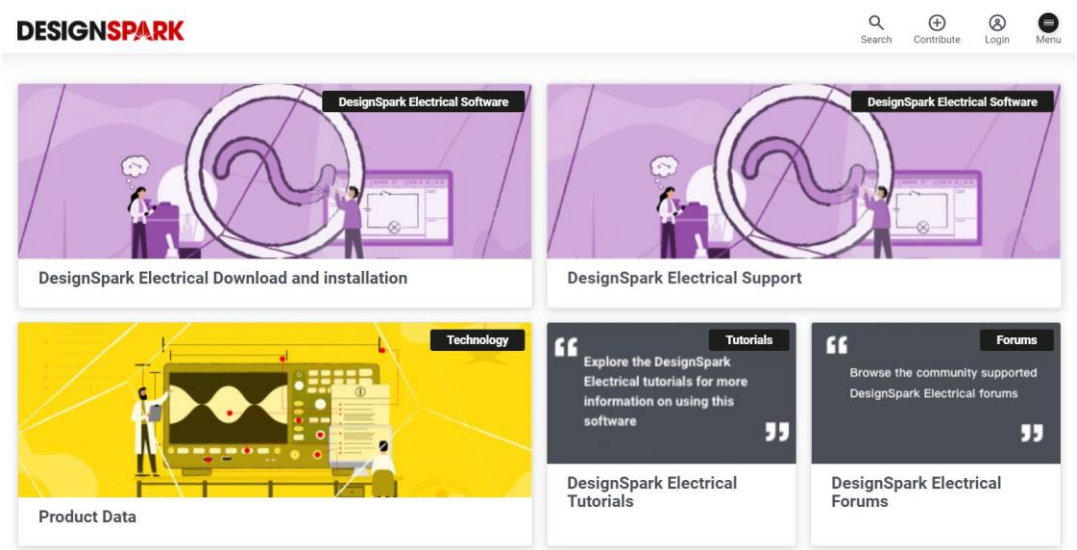


Figure 118 - Design spark Electrical software

The user interface is very similar to Solid Works Electricals, which is a professional electrical CAD software. Most of the functions of Solid Works Electricals are included in this software and this software is an open-source software so any one can download it through their website. Even a new user can get familiarize with the software since there are tutorials provided on the website.



Design Sparks offers a wide range of features that could be used to optimize the design process. Tasks such as device numbering and wire numbering are completely automated which will save huge amount of time when it comes to designing a complex electrical circuit with huge number of electrical components.
A vast range of symbols are available in withing the software and the integral part library has more than 250 000 parts with millions more available in the Design Spark Electrical online portal, so user can immediately find and apply it in their design immediately. Also, if anyone could find the part as required there is a function to build a custom library. Design Spark Electrical will automatically generate a Bill of Material for each project, including every part in the design and it can be output as an excel file.
The final design of the electrical diagram of the control panel of this system was designed using the Design Spark Electrical software. Most of the symbols were available withing the software and missing symbols were custom designed. Only the line diagram of the electrical circuit was designed because if the entire circuit were designed, the control circuit would be complex. The line diagram was designed to use it as the Worksheet when the Control Panel is being fabricating.

3.19.4 Designing a custom enclosure.

Designing the custom enclosure was done using the Solid Works software. Before designing the custom enclosure, a 3D model of the internal electrical circuit; how the components should be arranged to reduce the enclosure size, was modeled. This model helps to acquire the dimensions of the custom enclosure and also it is beneficial to have the 3D model to get aid when the fabricating the control panel.

After modeling the internal components, the dimension of the enclosure box obtained then the 3D model of the enclosure was designed. This model will be provided to a custom control panel enclosure manufacturer and get manufactured the custom enclosure.

3.19.4.1 3D model of the custom enclosure

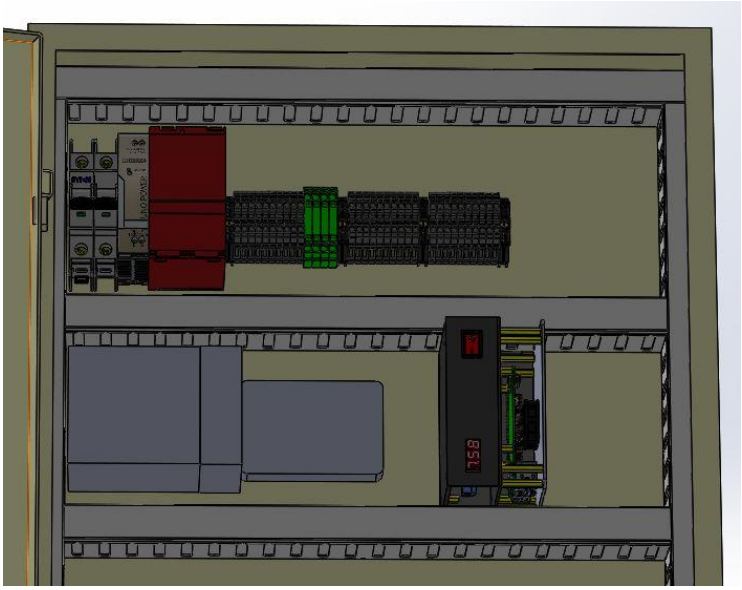


Figure 119 - Designed Control box I

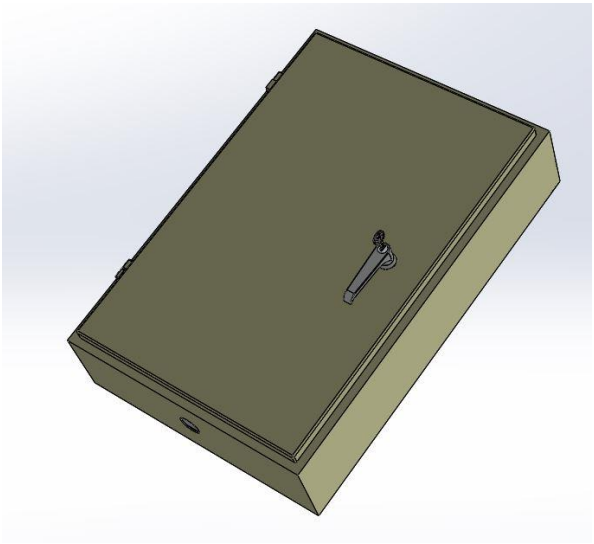


Figure 121- Designed Control box II

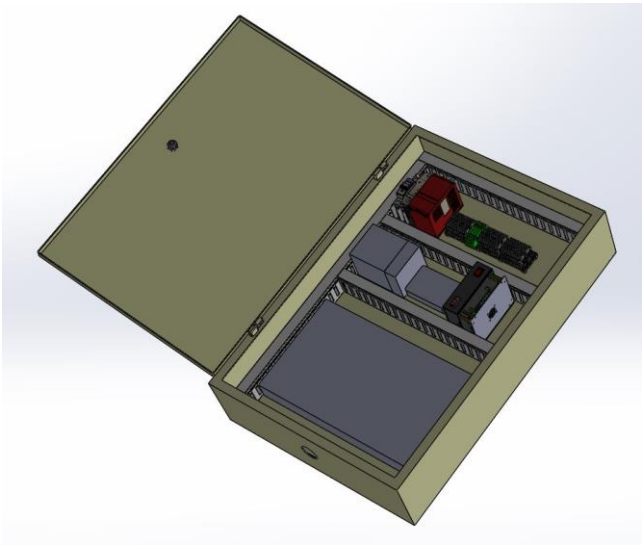


Figure 120- Designed Control box III

CHAPTER 4: RESULTS AND DISCUSSION

4.1 Real time data acquisition process

In this operation, the were achieved to record of 82 hours' worth vibration data from Bielomatik machine on 04/01/2021 09:46:14 to 13/01/2021 14:10:43 recording duration. This all data saved as in csv format in local memory space of the industrial PC. Each dataset file contained 2hrs data in 10 seconds reading period.

As previous chapter stated, the data acquisition process done by the manual procedure because of the belated arrival of components and the customized control box. Because of that there had to stay nearby the test rig and the Bielomatik machine in whole process for protection of both and troubleshooting of recently implemented system. Also, this region had high noise condition and due to that as company regulations normally a person has maximum 6hrs time per day for work in that region and there was compulsory to wear safety ear buds whole time near the machine. Consequence of that situations most collect data files has time gaps between each file due to the manual execution latency and other safety concerns.

As detailed, there were contained 41 data files and latest 6 of them used as test data in this validation operation and rest of the data used as train data for train the developed ML model and calculate threshold value for anomaly detection.

Further Validation of readings using Viber – X2 Pro, Handheld Vibration monitor device.

Figure 122 - Viber X2 pro

It was taken measurements manually also for the verification of the reading's samples.

4.2 Developed ML model validation.

Following figure is represent the resulted accuracy values of the latest 6 data files after running through the developed ML model.

Date & Time	Accuracy
13/1/2021 00:13	50.417
13/1/2021 02:25	87.778
13/1/2021 04:53	71.389
13/1/2021 07:18	88.75
13/1/2021 10:41	69.722
13/1/2021 13:10	86.945

According to the table there are slight fluctuation in accuracy between 50 to 90 regions. Then observing the reasons there were recognized some special conditions and operations assisted to these variations. Especially after job changing in printing section, many preventive maintenance related adjustments etc.

These are the graphically detailed representation of resulted final MD data and Boolean anomaly flag data of latest 6 datafiles in received datasets.

1) Data recording started time – 13/1/2021 00:13
    a) Smoothed MD visualization

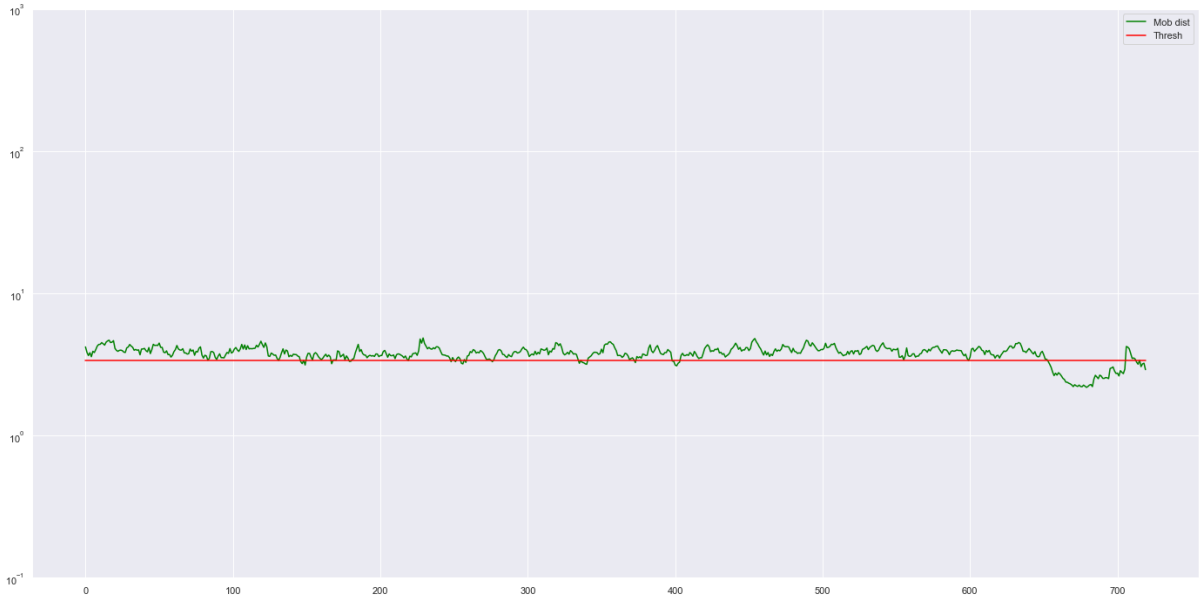


Figure 123 - 13/1/2021 00:13 smoothed MD

b) Boolean anomaly flag

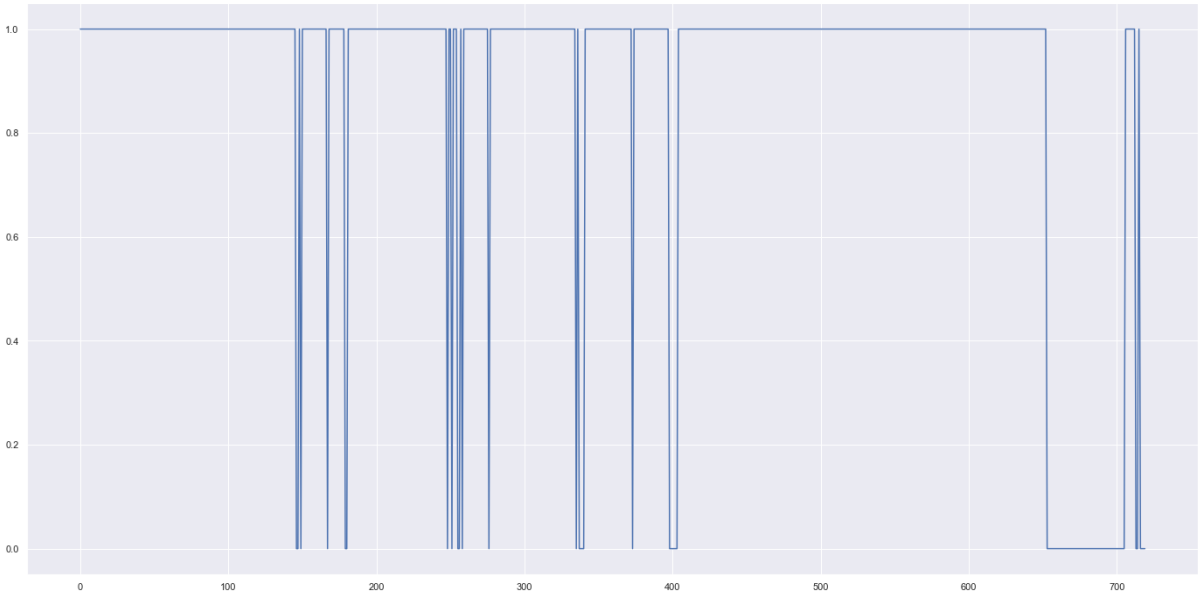


Figure 124 - 13/1/2021 00:13 Boolean anomaly

2) Data recording started time – 13/1/2021 02:25

a) Smoothed MD visualization

Figure 125 - 13/1/2021 02:25 smoothed MD

b) Boolean anomaly flag

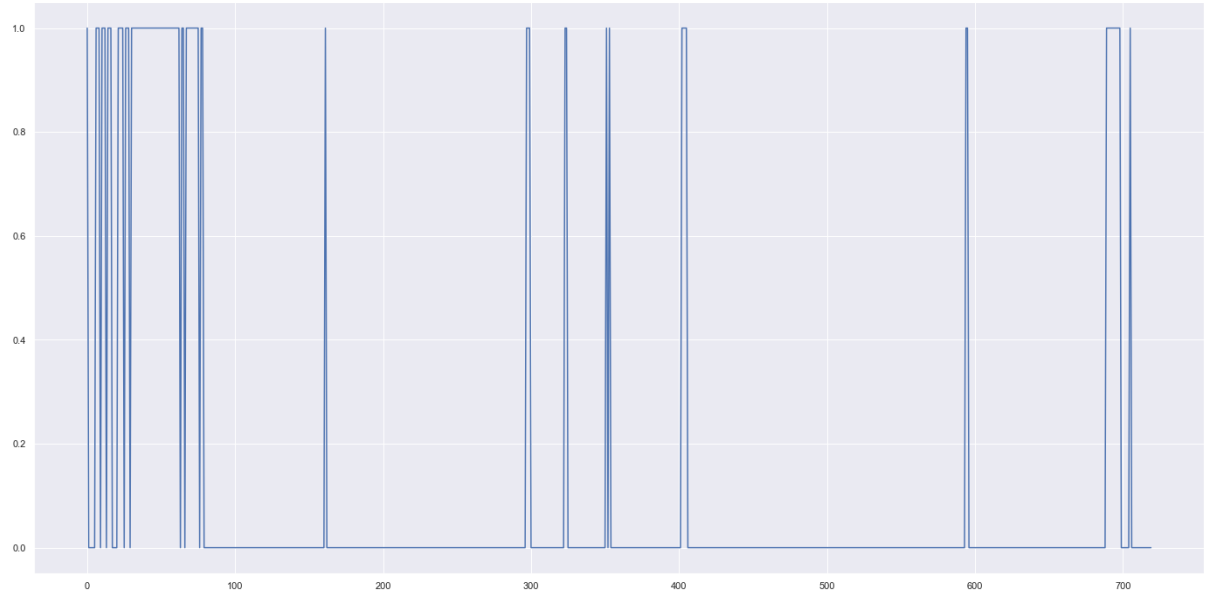


Figure 126- 13/1/2021 02:25 Boolean anomaly

3) Data recording started time – 13/1/2021 04:53

a) Smoothed MD visualization

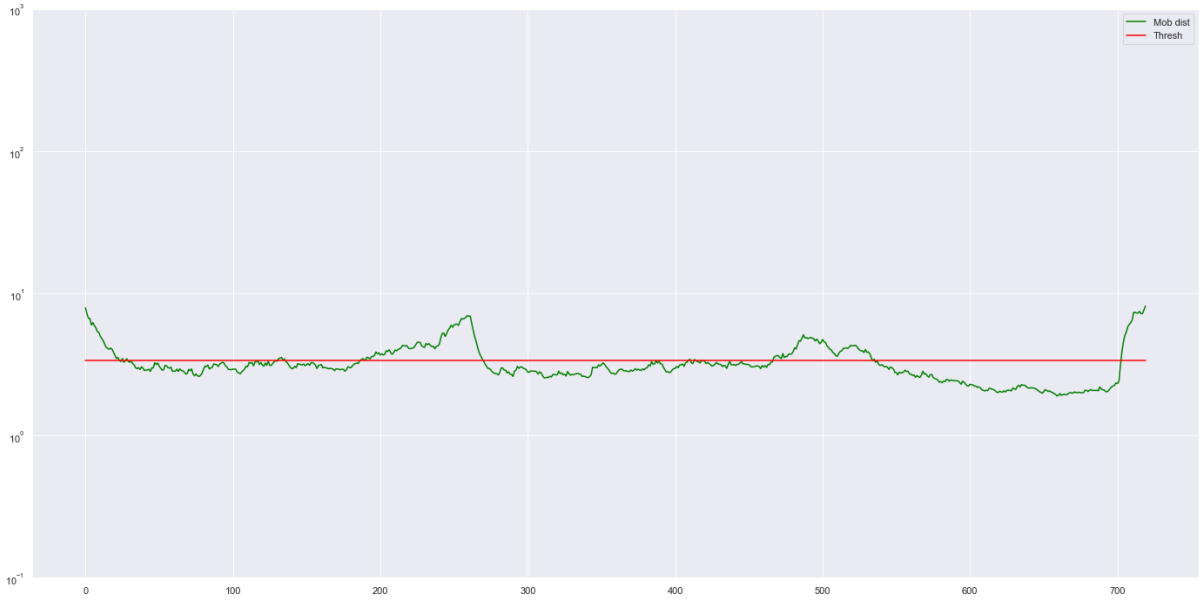


Figure 127- 13/1/2021 04:53 smoothed MD

b) Boolean anomaly flag

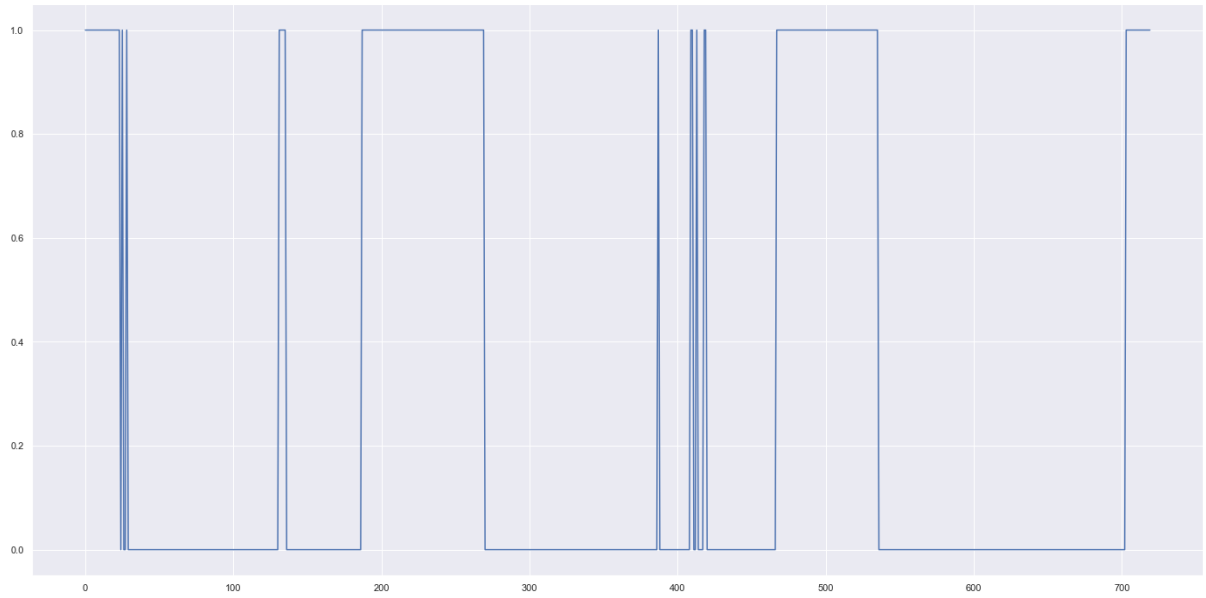


Figure 128- 13/1/2021 04:53 Boolean anomaly

4) Data recording started time – 13/1/2021 07:18

a) Smoothed MD visualization

Figure 129- 13/1/2021 07:18 smoothed MD

b) Boolean anomaly flag

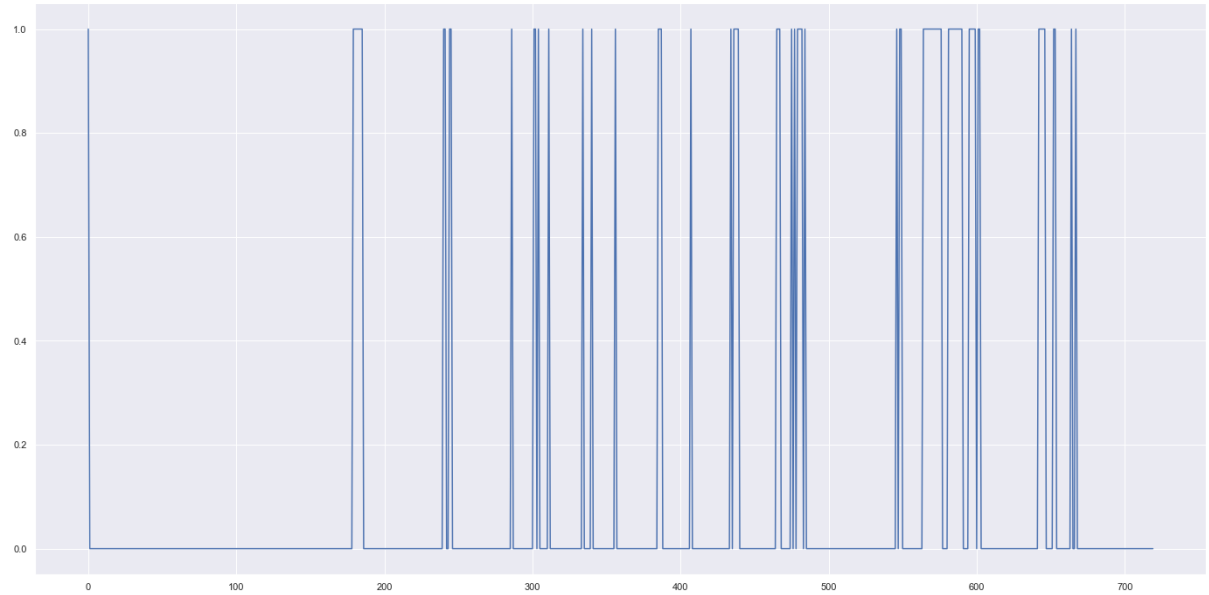


Figure 130- 13/1/2021 07:18 Boolean anomaly

5)

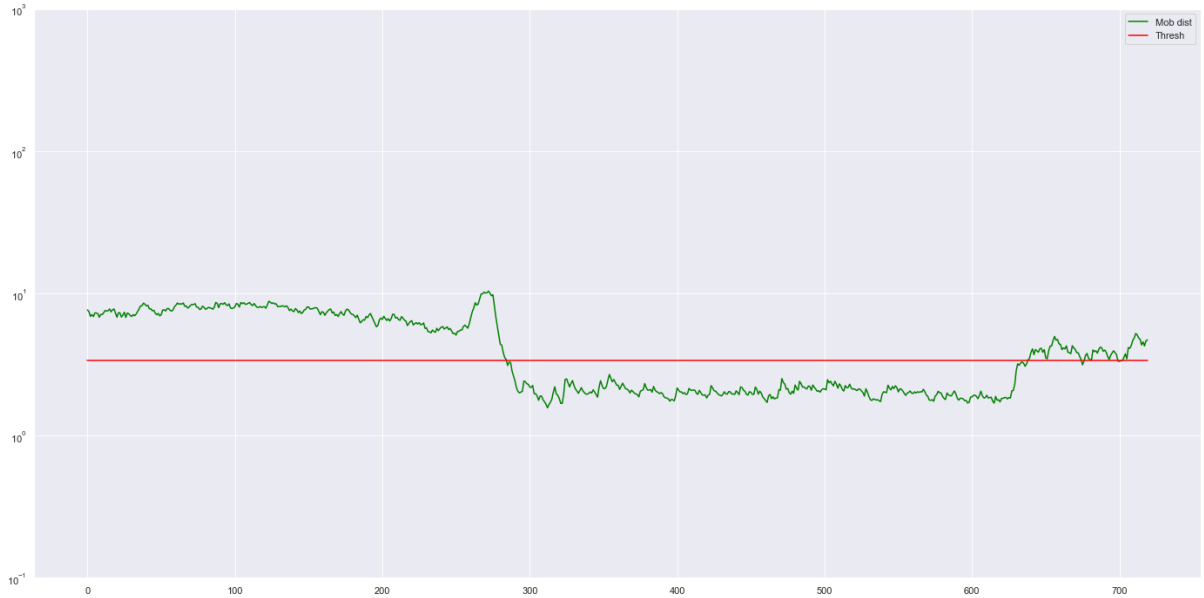


Figure 131- 13/1/2021 10:41 smoothed MD

b) Boolean anomaly flag

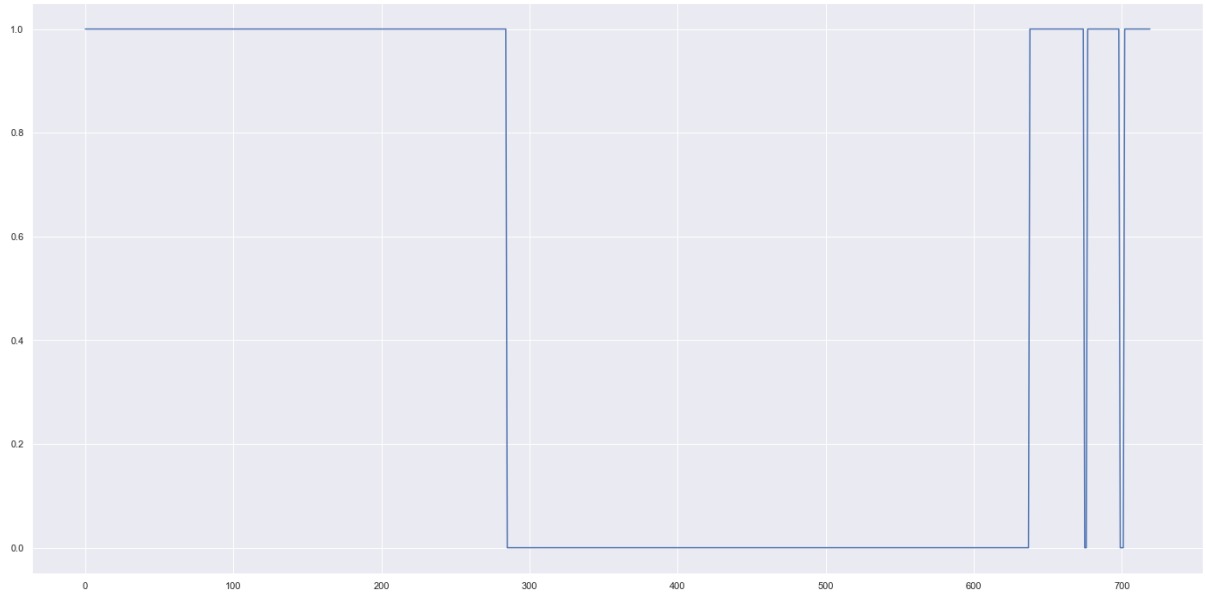


Figure 132- 13/1/2021 10:41 Boolean anomaly

6) Data recording started time – 13/1/2021 13:10

a) Smoothed MD visualization

Figure 133- 13/1/2021 13:10 smoothed MD

b) Boolean anomaly flag

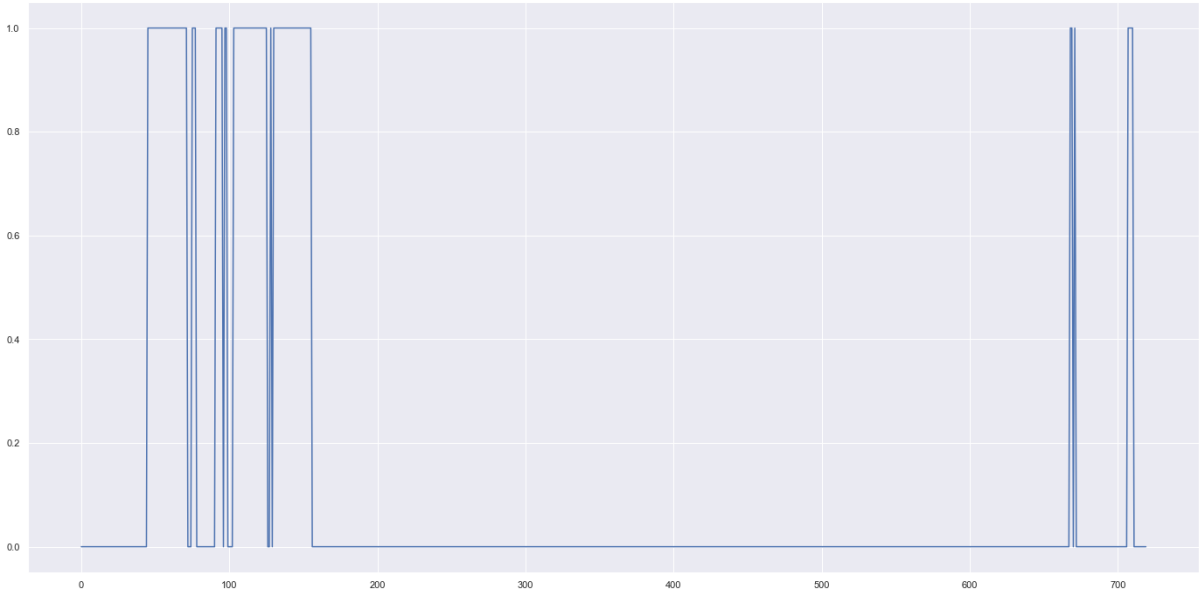
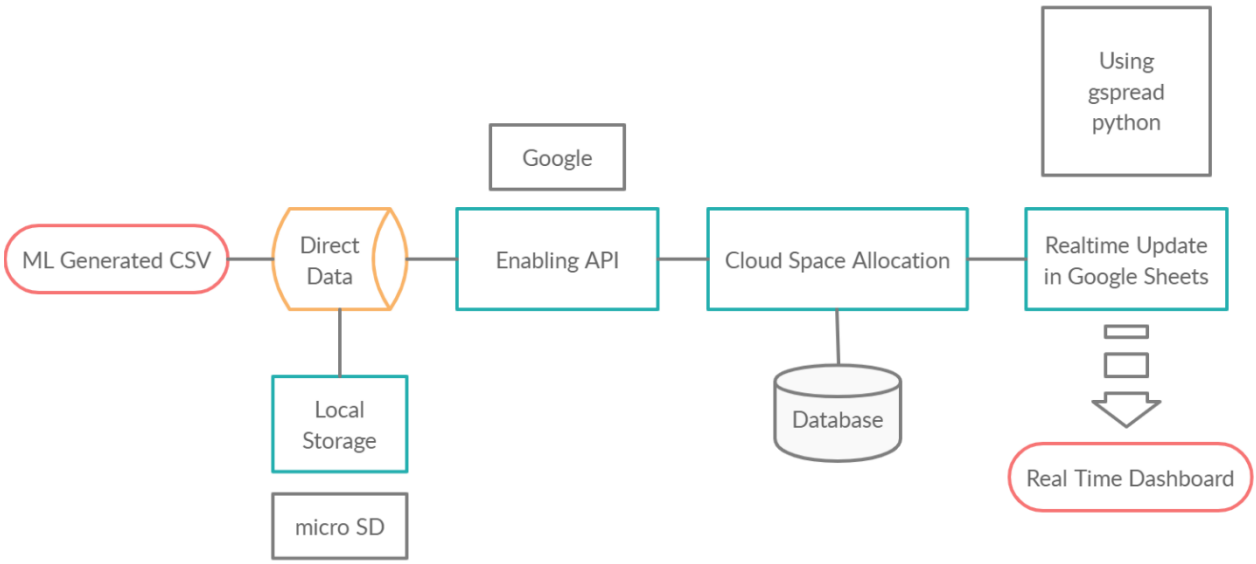


Figure 134- 13/1/2021 13:10 Boolean anomaly

4.3 Predictive Maintenance Model and the Maintenance Dashboard

The outcome of the project is to introduce the predictive maintenance model in replacement of the traditional preventive maintenance. The system should be implemented as it can be sensible foot the maintenance and the management staff. We introduced a maintenance dashboard using IoT for Realtime monitoring of the machine condition and maintenance scheduling.



1. Dashboard using MQTT based platform.

The total industrial grade solution will be the MQTT based online dashboard, but with limited resources it was switched to a Free Google Dashboard with online database integrated with Google Sheets.

2. Realtime Dashboard using Google Data Studio

Data Studio is a free tool that turns the data into informative, easy to read, easy to share, and fully customizable dashboards and reports.

We used Google Sheets as the main data connector, but to make it fully automated we implemented a python script so the ML generated CSV can be directly uploaded into google sheets then it can act as the main connector.

We used “gspread” and “oauth2client.service_account” to import the data from the CSV files.

```
import gspread
from oauth2client.service_account import ServiceAccountCredentials

scope = ["https://spreadsheets.google.com/feeds",
'https://www.googleapis.com/auth/spreadsheets',
         "https://www.googleapis.com/auth/drive.file",
"https://www.googleapis.com/auth/drive"]

credentials =
ServiceAccountCredentials.from_json_keyfile_name('client_secret.json',
scope)
client = gspread.authorize(credentials)

spreadsheet1 = client.open('Maintenance')

with open('Maintenance_shedule.csv', 'r') as file_obj:
    content = file_obj.read()
    client.import_csv(spreadsheet.id, data=content)

spreadsheet2 = client.open('Maintenance_Schedule')



with open('Maintenance_schedule.csv', 'r') as file_obj:
    content = file_obj.read()
    client.import_csv(spreadsheet.id, data=content)

spreadsheet3 = client.open('Channel_Data')

with open('2021_01_13_12_10_34.csv', 'r') as file_obj:
    content = file_obj.read()
    client.import_csv(spreadsheet.id, data=content)
```

CHAPTER 5: CONCLUSION AND RECOMMENDATIONS

5.1 Data acquisition

In this case, due to the pandemic condition the data acquisition had to limit in to 2 weeks duration. In these kinds of application there need to require considerable amount of data for mainly in training purpose for generate most optimized threshold value. By this method, the system can decrease errors and gain accuracy of the system. Furthermore, in this system principally the classification done by healthy and defected condition. For reason of that that the training data would be collected on the machine's finest health condition. In view of that fact, the training dataset acquisition would be performed recently after the annual maintenance of the machine would be more advantageous.

Moreover, by performing continuous and automated data acquisition process would be helpful to detect most of possible vibration patterns on the system with numerous working conditions. Because these captured data would be straightly effect on outlier detection and cause to improve the effect of the ML model to the maintenance scheduling and condition monitoring process.

5.2 ML model development and maintenance scheduling

Mainly, the ML model developed based on the receive dataset from the IMS, University of Cincinnati on this project. The data sampling rate preprocessing and various operations consist on the software model was based on the nature of the test data on the selected resource. Then after the implementation this model was deployed with Bielomatik machine related data and in time series visualization there could see some variation differences between both of data. In the test rig data, this system built intended of experiment of health of bearings purpose and in this apparatus, there were generally eliminated most of unnecessary effects and vibrations. But In this industry-based application there need to deal with this kind of noises. Because of that this developed model may have some complications related with facing non considered obstacles and the point of that there require some updates and troubleshoots sessions through some considerable time to for finely maintain to eliminate occurring problems and define proper parameter values in each related operation.

As a brief explanation about the developed ML model, PCA algorithm use to dimensionally reduction and Mahalanobis distance use to measure distance between each specific point on considered distribution. In this case, there were developed model to reduce dimension n to 1 by PCA for future analysis. In this approach the whole system considers as one unit in failure

prediction. It means if there any bearing had symptoms to be fail the model will warn about a system has a bearing fail. In the future development with reasonable amount of real time data there must be consider about extend the system to monitor each joint with successful failure prediction. But now lack of historical data is the main cause of limitation in bearing health prediction.

For more sufficient computation accuracy and efficient computational power there is essential to consider about compression ratio in PCA data compression. M. Mrowczynka, et. al (2020) stated that if sufficiently high compression ratio used, the noise introduced by the compression is high enough to contain noise that distorts the original data.

5.3 Maintenance Dashboard

In the developed maintenance dashboard, it is proposed to include,

1. Machine Condition Monitoring
2. Predictive Maintenance Dashboard
3. Realtime Prediction of Bearing Condition
4. Past Maintenance Data Access
5. Maintenance Scheduling

5.3.1 The implemented Dashboard

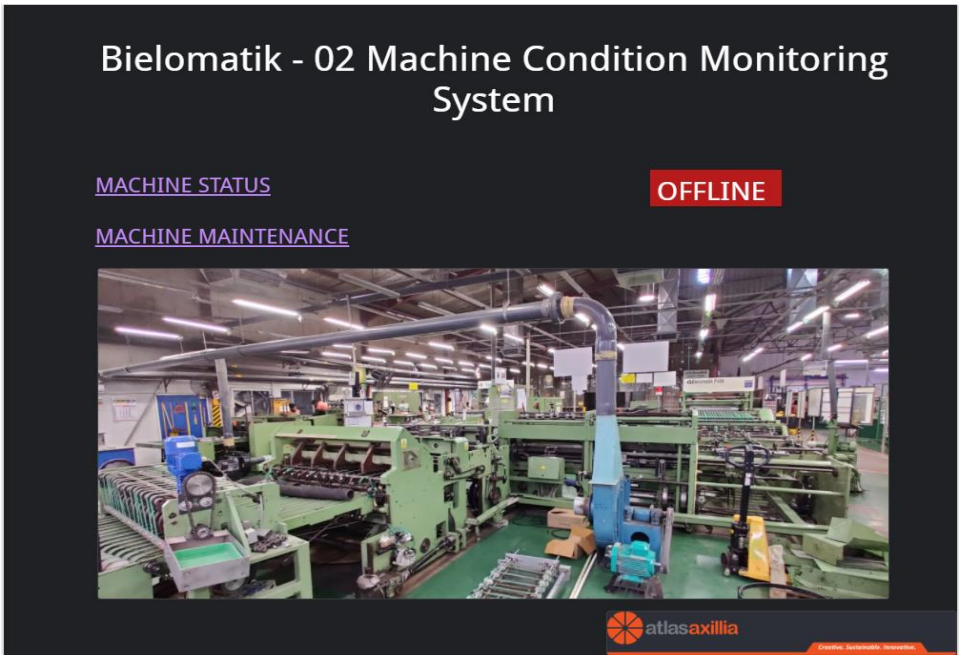


Figure 135 - Maintenance Dashboard Home Page

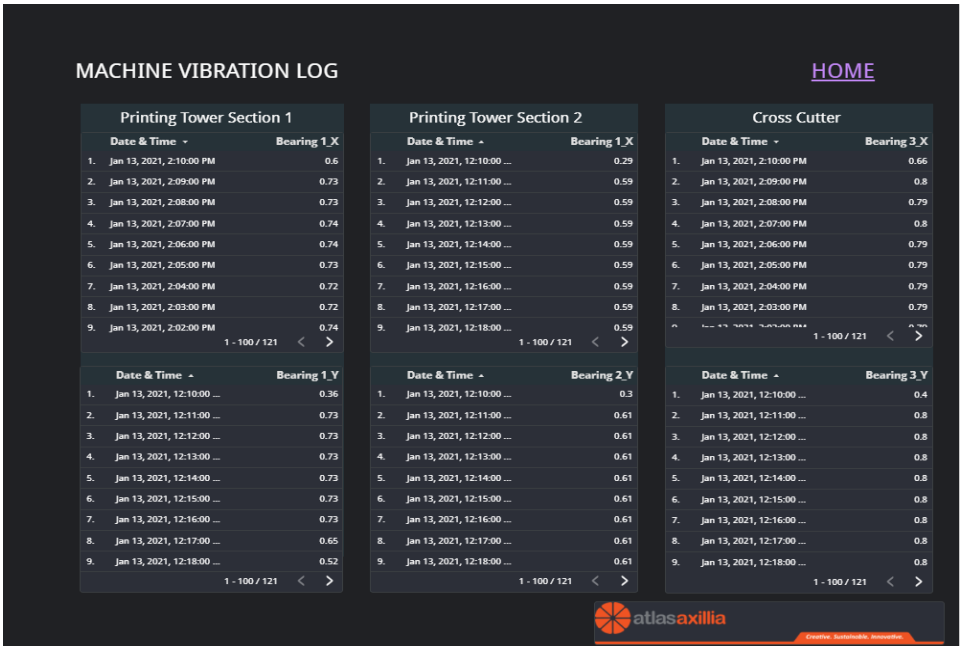


Figure 136 - Log Note

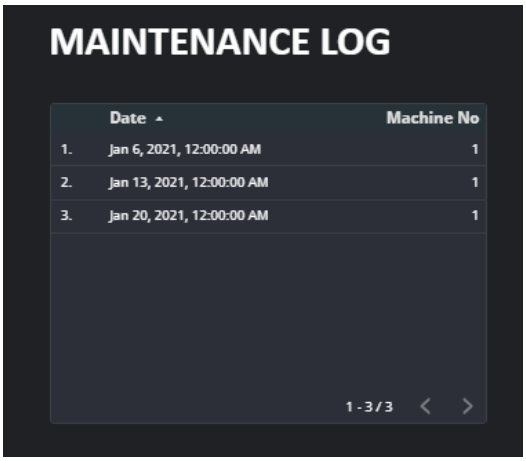


Figure 137 - Maintenance Scheduling

References

- Carnero, M. C. (2006). An evaluation system of the setting up of predictive maintenance programmes. *Reliability Engineering and System Safety*, *91*(8), 945–963. https://doi.org/10.1016/j.ress.2005.09.003
- Chiu, Y. C., Cheng, F. T., & Huang, H. C. (2017). Developing a factory-wide intelligent predictive maintenance system based on Industry 4.0. *Journal of the Chinese Institute of Engineers, Transactions of the Chinese*
- *Institute of Engineers,Series A*, *40*(7), 562–571. https://doi.org/10.1080/02533839.2017.1362357
- Hariharan, V., & Srinivasan, P. S. S. (2009). Vibration analysis of misaligned shaft -ball bearing system. *Indian Journal of Science and Technology*, *2*(9). https://doi.org/10.17485/ijst/2009/v2i9/2952144
- Jayaswal, P., Wadhwani, A. K., & Mulchandani, K. B. (2008). Machine fault signature analysis. *International Journal of Rotating Machinery*, *2008*. https://doi.org/10.1155/2008/583982
- Kostyukov, V. N., & Kostyukov, A. V. (2015). Real-time condition monitoring of machinery malfunctions. *Procedia Engineering*, *113*, 316–323. https://doi.org/10.1016/j.proeng.2015.07.272
- Lennvall, T., Gidlund, M., & Akerberg, J. (2017). Challenges when bringing IoT into industrial automation. *2017 IEEE AFRICON: Science, Technology and Innovation for Africa, AFRICON 2017*, 905–910. https://doi.org/10.1109/AFRCON.2017.8095602
- Okano, M. T. (2017). IOT and Industry 4.0: The Industrial New Revolution. *ICMIS-17 - International Conference on Management and Information Systems*, *September*, 75–82.
- Sabato, A., Niezrecki, C., & Fortino, G. (2017). Wireless MEMS-Based Accelerometer Sensor Boards for Structural Vibration Monitoring: A Review. *IEEE Sensors Journal*, *17*(2), 226–235. https://doi.org/10.1109/JSEN.2016.2630008
- Spendla, L., Kebisek, M., Tanuska, P., & Hrcka, L. (2017). Concept of predictive maintenance of production systems in accordance with industry 4.0. *SAMI 2017 - IEEE 15th International Symposium on Applied Machine Intelligence and Informatics, Proceedings*, 405–410. https://doi.org/10.1109/SAMI.2017.7880343
- Utz, A., Walk, C., Stanitzki, A., Mokhtari, M., Kraft, M., & Kokozinski, R. (2018). A High-Precision and High-Bandwidth MEMS-Based Capacitive Accelerometer. *IEEE Sensors Journal*, *18*(16), 6533–6539. https://doi.org/10.1109/JSEN.2018.2849873

- McKinney, W., n.d. *Python for Data Analysis*. 2nd ed. Sebastopol, CA 95472: O'Reilly Media, Inc.
- Yang, H., Mathew, Y. and Ma, L., 2002. Intelligent Diagnosis of Rotating Machinery Faults. *3rd Asia-Pacific Conference on Systems Integrity and Maintenance*, 3rd Asia-Pacific Conference on Systems Integrity and Maintenance, pp.25-27.
- Ali, Y., 2018. Artificial Intelligence Application in Machine Condition Monitoring and Fault Diagnosis. *Intechopen*, [online] 14. Available at: <http://dx.doi.org/10.5772/intechopen.74932> [Accessed 18 December 2020].
- Amruthnath, N. and Gupta, T., 2018. A Research Study on Unsupervised Machine Learning Algorithms for Early Fault Detection in Predictive Maintenance. *2018 5th International Conference on Industrial Engineering and Applications*.
- Wu, S., Wu, C., Wu, T. and Wang, C., 2013. Multi-Scale Analysis Based Ball Bearing Defect Diagnostics Using Mahalanobis Distance and Support Vector Machine. *Entropy*, 15(2), pp.416-433.
- Sohaib, M. and Kim, J., 2018. Reliable Fault Diagnosis of Rotary Machine Bearings Using a Stacked Sparse Autoencoder-Based Deep Neural Network. *Shock and Vibration*, 2018, pp.1-11.
- Qiu, H., Lee, J., Lin, J. and Yu, G., 2006. Wavelet filter-based weak signature detection method and its application on rolling element bearing prognostics. *Journal of Sound and Vibration*, 289(4-5), pp.1066-1090.
- Luashchuk, A., 2019. 8 Reasons Why Python is Good for Artificial Intelligence and Machine Learning. [online] Medium. Available at: <https://towardsdatascience.com/8-reasons-why-python-is-good-for-artificial-intelligence-and-machine-learning-4a23f6bed2e6> [Accessed 21 January 2021].
- Beklemysheva, A., n.d. Why Use Python for AI and Machine Learning?. [online] Steelkiwi.com. Available at: <https://steelkiwi.com/blog/python-for-ai-and-machine-learning/#:~:text=Python%20offers%20concise%20and%20readable,developers%20to%20write%20reliable%20systems.&text=Python%20code%20is%20understandable%20by,build%20models%20for%20machine%20learning> [Accessed 31 January 2021].
- K. Zhou, Taigang Liu and Lifeng Zhou, "Industry 4.0: Towards future industrial opportunities and challenges," *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, Zhangjiajie, 2015, pp. 2147-2152, doi: 10.1109/FSKD.2015.7382284.

- E. Manavalan, K. Jayakrishna, A review of Internet of Things (IoT) embedded sustainable supply chain for industry 4.0 requirements, Computers & Industrial Engineering, Volume 127, 2019, Pages 925-953, ISSN 0360-8352, https://doi.org/10.1016/j.cie.2018.11.030.
- De Reus, R., Gulløv, J.O. and Scheeper, P.R., 1999. Fabrication and characterization of a piezoelectric accelerometer. *Journal of Micromechanics and Microengineering*, *9*(2), p.123.
- Gesing, A.L., Alves, F.D.P., Paul, S. and Cordioli, J.A., 2018. On the design of a MEMS piezoelectric accelerometer coupled to the middle ear as an implantable sensor for hearing devices. *Scientific reports*, *8*(1), pp.1-10.
- Tiwari, M., Gupta, K. and Prakash, O., 2000. Effect of radial internal clearance of a ball bearing on the dynamics of a balanced horizontal rotor. Journal of sound and vibration, 238(5), pp.723-756.

APENDIX

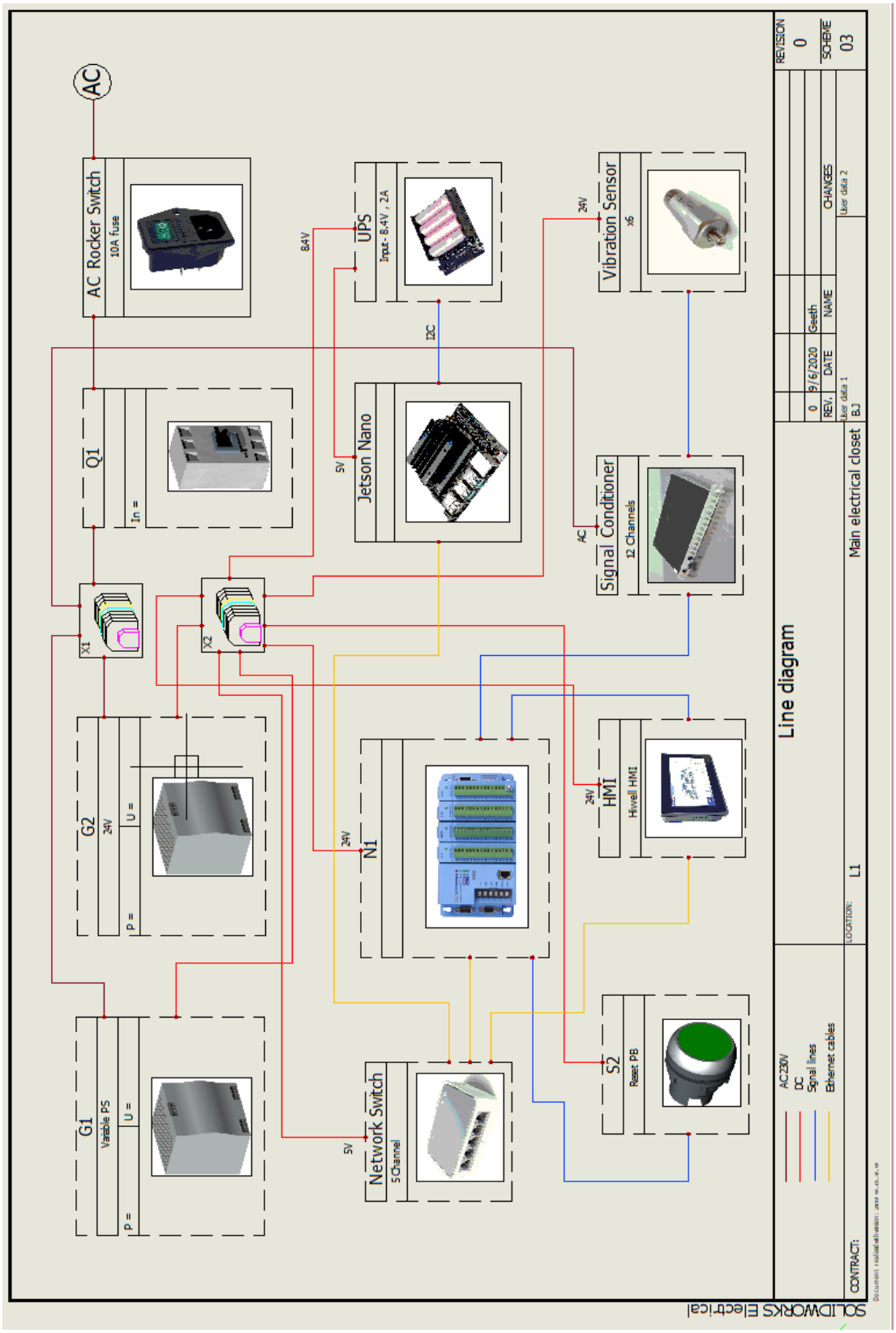
APPENDIX-A: TECHNICAL DRAWINGS

XL6009 DC-DC Buck Boost Converter Module

Specifications

Current	4A max (we recommended 1A only) No-load - 18mA
Efficiency	<94% (greater the pressure, the lower the efficiency)
Frequency	Switching frequency 400KHz
Input Voltage	3V ~ 32V
Length x Width x Height	43mm x 21mm x 14mm
Output Voltage	5V ~ 35V

Diagram of the Final Electrical Circuit



APPENDIX-B: ELECTRONIC COMPONENT SPECIFICATIONS

Piezoelectric Accelerometer (Isolated)

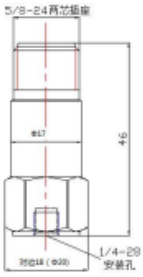
Model: CA-YD-187T02

Features:

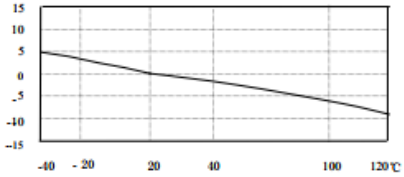
- Built-in IEPE, low noise, strong resistance of disturbance
- Isolated, Floating, industrial site monitoring
- 2-pin MIL-C-5015 two-pin socket output

Specifications:

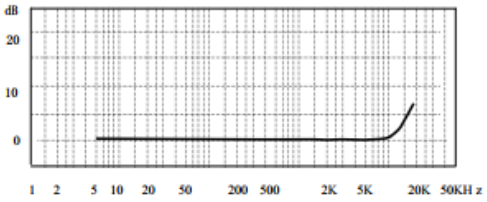
Sensitivity（20±5℃）	100 mV/g
Measurement Range（pk）	±50 g
Max Transverse Sensitivity	＜5%
Frequency Response(±10%)	1～10,000 Hz
（±3dB)	0.4～12,000 Hz
Mounting Resonant Frequency	≥30 KHz
Amplitude linearity	≤1%
Operating Temp. Range	-40～+120 ℃
Temperature Response	See Graph
Shock Limit(pk)	3000 g
Noise(rms)	＜50 μV
Output Impedance	＜100 Ω
Operating Voltage(Constant current source)	18～28 VDC
Operating Current	2～10 mA
DC Bias Voltage	12±2 VDC
Case Isolation	$\geq 10^8$ Ω
Mounting	1/4-28
Sensing Element	Piezoelectric Ceramics
Sensing Geometry	Shear
Case Material	304 Stainless steel
Weight	~52 g
Output	Top 5/8-24 2-pin Socket
Accessories	
Certificate	Calibration parameters
1/4-28 Stud	one
3106A(5015)Plug, Cable(RVVP)	one (3m)，one end free



Dimensions



Temperature Response



Frequency Response

IEPE Signal Conditioner (Multi-channels)

Model: YE3826A

Features:

- Selectable Current Source 4 or 10 mA to meet different requirement
- Selectable Gain 1 or 10
- Operation status display
- Wide Frequency Range
- Standard 19' Inch Case, Multi-channels Structure
- Suitable for industrial monitoring

Specifications:

Input channel	12
Input Voltage	≤±10VP（IEPE Transducer Output）
Excitation Source	Voltage:+24VDC； Current：4mA, 10mA±10%(Internal Setting, Default Setting 4mA)
Upper frequency	100kHz (Cutting Frequency -3dB±1dB、-12dB/OCT)
Lower frequency	≤0.3Hz
Gain	1 or 10 Selectable Switch
Output amplitude	≤±10VP/5mA
Accuracy	≤±1%
Noise	≤1mVrms
Temperature	Operate temperature: 0°C~40°C Storage temperature: -55°C~85°C
Moisture	Maximum 80%R.H.
Power supply	AC：AC220V±10%50Hz0.2A
Dimensions	430mm(W)×44mm(H)× 240mm(D)
Weight	About 2kg
Connection	Input: BNC, Output: BNC/DB25
Accessories	
Input/output cables	12 Each
Power supply cable	One

T Series - Standard PLC MPU (-e : Built-in Ethernet port)

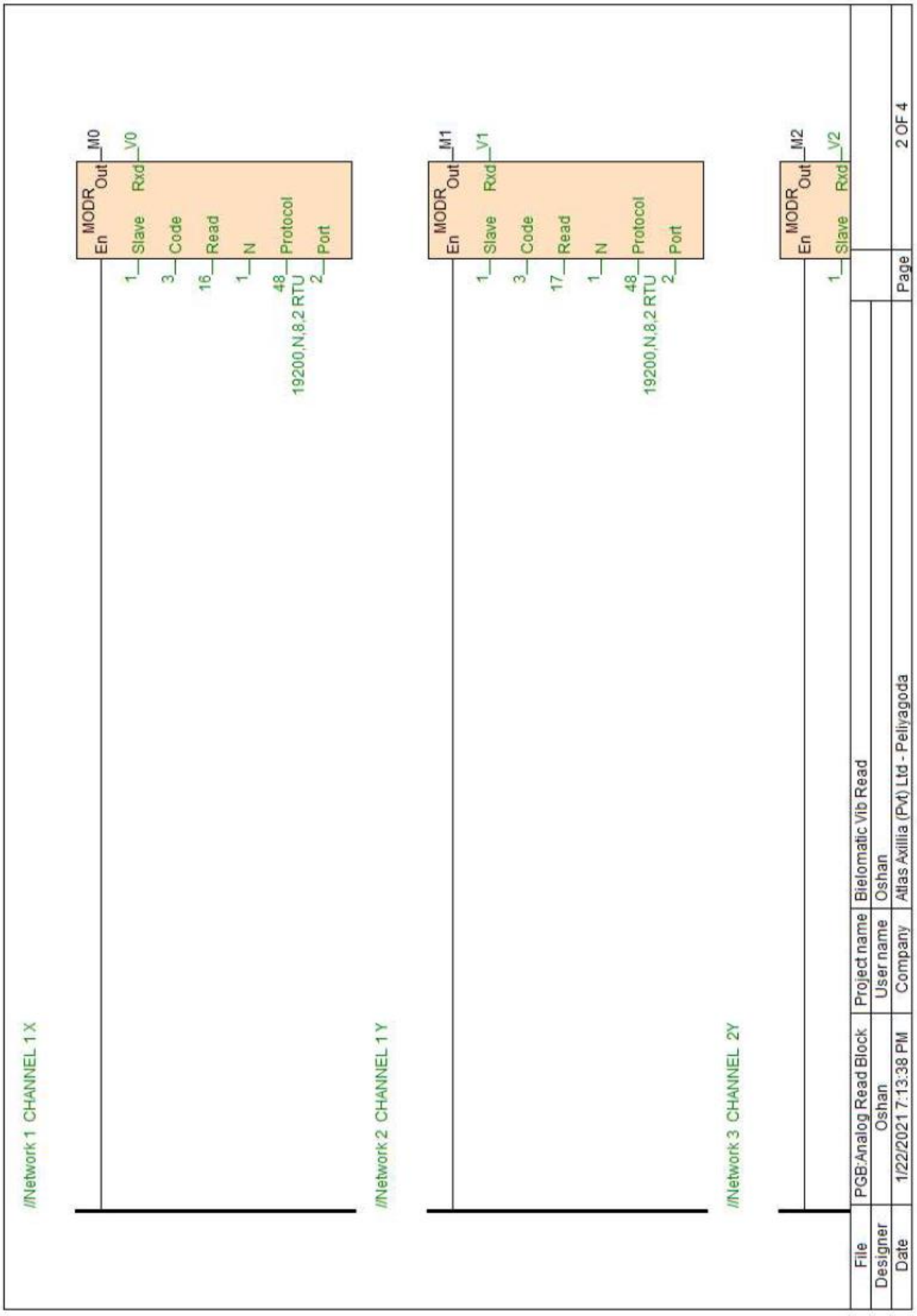
Ethernet Model		Model		Specification						Dimension WxHxD
24V DC	220V AC	24V DC	220V AC	DI	DO	Pulse Input	Pulse Output	COM port	Max exp.	
T16S0R-e	T16S2R-e	T16S0R	T16S2R	8	8 Relay	2 Channels A/B phase (4 points) 200KHz		RS232+RS485 , Max 5 ports	7	 93×95×82mm
T16S0T-e	T16S2T-e	T16S0T	T16S2T	8	8 Transistor NPN	2 Channels A/B phase (4 points) 200KHz	2 Channels A/B phase (4 points) 200KHz	RS232+RS485 , Max 5 ports	7	
T16S0P-e	T16S2P-e	T16S0P	T16S2P	8	8 Transistor PNP					
T24S0R-e	T24S2R-e	T24S0R	T24S2R	16	8 Relay	2 Channels A/B phase (4 points) 200KHz		RS232+RS485 , Max 5 ports	7	 131×95×82mm
T24S0T-e	T24S2T-e	T24S0T	T24S2T	16	8 Transistor NPN	2 Channels A/B phase (4 points) 200KHz	2 Channels A/B phase (4 points) 200KHz	RS232+RS485 , Max 5 ports	7	
T24S0P-e	T24S2P-e	T24S0P	T24S2P	16	8 Transistor PNP					
T32S0R-e	T32S2R-e	T32S0R	T32S2R	16	16 Relay	2 Channels A/B phase (4 points) 200KHz		RS232+RS485 , Max 5 ports	7	
T32S0T-e	T32S2T-e	T32S0T	T32S2T	16	16 Transistor NPN	2 Channels A/B phase (4 points) 200KHz	2 Channels A/B phase (4 points) 200KHz	RS232+RS485 , Max 5 ports	7	
T32S0P-e	T32S2P-e	T32S0P	T32S2P	16	16 Transistor PNP					
T48S0R-e	T48S2R-e	T48S0R	T48S2R	28	20 Relay	2 Channels A/B phase (4 points) 200KHz		RS232+RS485 , Max 5 ports	7	 177×95×82mm
T48S0T-e	T48S2T-e	T48S0T	T48S2T	28	20 Transistor NPN	2 Channels A/B phase (4 points) 200KHz	2 Channels A/B phase (4 points) 200KHz	RS232+RS485 , Max 5 ports	7	
T48S0P-e	T48S2P-e	T48S0P	T48S2P	28	20 Transistor PNP					
T60S0R-e	T60S2R-e	T60S0R	T60S2R	36	24 Relay	2 Channels A/B phase (4 points) 200KHz		RS232 + RS485, Max 5 ports	7	
T60S0T-e	T60S2T-e	T60S0T	T60S2T	36	24 Transistor NPN	2 Channels A/B phase (4 points) 200KHz	2 Channels A/B phase (4 points) 200KHz	RS232+RS485 , Max 5 ports	7	
T60S0P-e	T60S2P-e	T60S0P	T60S2P	36	24 Transistor PNP					

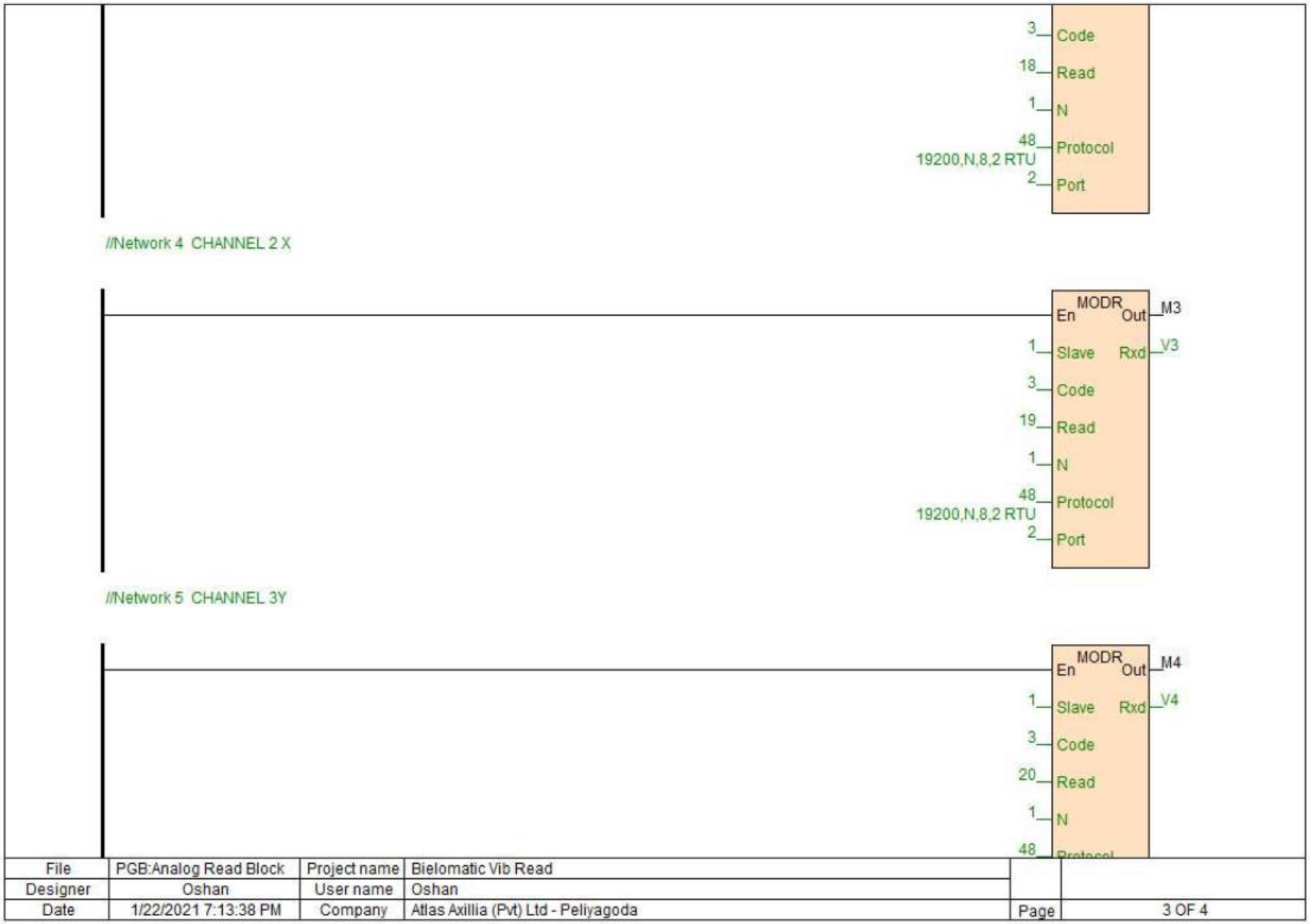
Analog Modules (-e : Built-in Ethernet port)

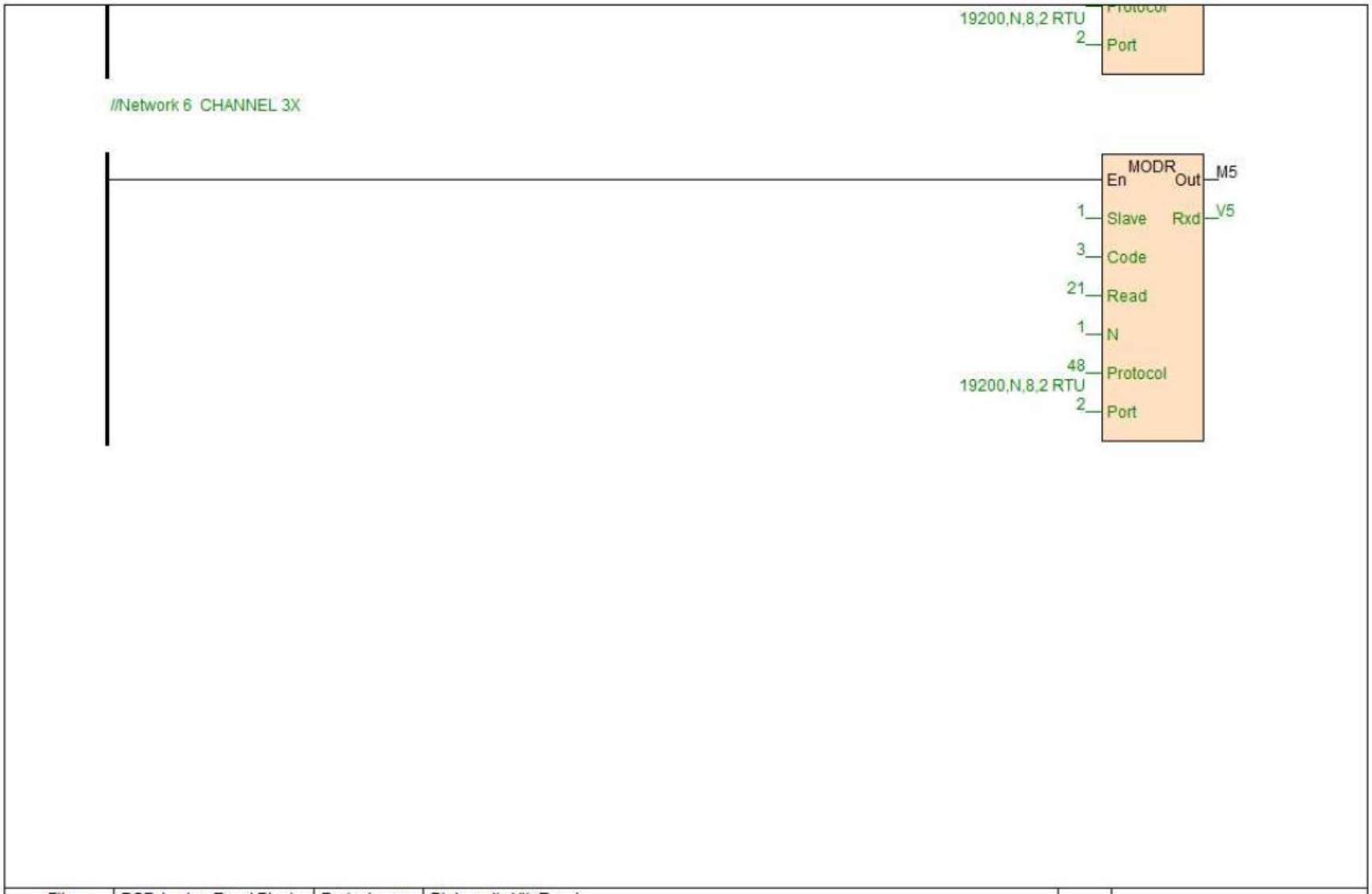
Ethernet Model	Model	Specification				Dimension WxHxD
24V DC	24V DC	AI	AO	Conversion Accuracy	Communication	
	S04AI	4		12 bits	RS485, support remote IO function	 70×95×82mm
	S04AO		4	12 bits	RS485, support remote IO function	
	S04XA	2	2	12 bits	RS485, support remote IO function	
S08AI-e	S08AI	8		12 bits	RS485, support remote IO function	 93×95×82mm
S08AO-e	S08AO		8	12 bits	RS485, support remote IO function	
S08XA-e	S08XA	4	4	12 bits	RS485, support remote IO function	

- Can be used as extension module for any Haiwell PLC host;
- Modules with RS485 port can be use as remote I/O;
- AI, AO supports 6 kinds of signal types: [4,20]mA, [1,5]V, [0,20]mA, [0,5]V, [0,10]V, [-10,10]V;
- Expansion modules with Ethernet port and RS458 port, can be remote IO unit by distributed installation.

APPENDIX-C: CIRCUIT DRAWINGS




3
Code
18
Read
1
N
48
19200,N,8,2 RTU
Protocol
2
Port
//Network 4 CHANNEL 2 X
MODR
En
Out
M3
1
Slave
Rxd
V3
3
Code
19
Read
1
N
48
19200,N,8,2 RTU
Protocol
2
Port
//Network 5 CHANNEL 3Y
MODR
En
Out
M4
1
Slave
Rxd
V4
3
Code
20
Read
1
N
48
File
PGB:Analog Read Block
Project name
Bielomatic Vib Read
Designer
Oshan
User name
Oshan
Date
1/22/2021 7:13:38 PM
Company
Atlas Axillia (Pvt) Ltd - Peliyagoda
Page
3 OF 4



File	PGB:Analog Read Block	Project name	Bielomatic Vib Read		
Designer	Oshan	User name	Oshan		
Date	1/22/2021 7:13:38 PM	Company	Atlas Axillia (Pvt) Ltd - Peliyagoda	Page	4 OF 4

APPENDIX-D: CODE

Condition monitoring and Maintenance Scheduling

```
#!/usr/bin/env python
# coding: utf-8

# In[583]:
import os
import glob
import csv
import pandas as pd
from pandas import DataFrame
import numpy as np
from sklearn import preprocessing
from sklearn import model_selection
import joblib
import seaborn as sns
sns.set(color_codes=True)
import matplotlib.pyplot as plt
#%matplotlib inline
from numpy.random import seed
import scipy.io
from scipy.fftpack import fft
import glob
import time
time_start = time.perf_counter

# # Detect latest csv files in directory
# In[584]:
path = "C:/Users/Atlas Printing/Predictive Maintenance Model/ATLAS
preprocessed data/"
most_recent = max(glob.iglob("C:/Users/Atlas Printing/Predictive
Maintenance Model/ATLAS preprocessed data/*.csv"),
key=os.path.getmtime)
```

```
print(most_recent)
# In[585]:
data_from_csv = pd.read_csv(most_recent)
#data_from_csv.head()
# In[586]:
index_merged_data = data_from_csv.index
number_of_rows_merged_data = len(index_merged_data)
print(number_of_rows_merged_data)
# In[587]:
#print(data_from_csv.info())
# In[588]:
col_list_merged_data = ["Date & Time", "Bearing 1_X", "Bearing 1_Y",
"Bearing 2_X", "Bearing 2_Y", "Bearing 3_X", "Bearing 3_Y", "Bearing
4_X", "Bearing 4_Y"]
df_merged_data = pd.read_csv(most_recent,
usecols=col_list_merged_data)
#df_merged_data.head()
# # Plot all data
# In[589]:
df_bearing_all_graph = DataFrame(data_from_csv, columns=['Date &
Time', 'Bearing 1_X', 'Bearing 1_Y','Bearing 2_X', 'Bearing
2_Y','Bearing 3_X', 'Bearing 3_Y','Bearing 4_X', 'Bearing 4_Y'])
#df_bearing_all_graph.plot(x ='Date & Time', figsize = (24,12))

# # Bearing 1
# In[590]:
#bearing 1 data from all csv
df_bearing_1_graph = DataFrame(data_from_csv, columns=['Date &
Time', 'Bearing 1_X', 'Bearing 1_Y'])
#df_bearing_1_graph.plot(x ='Date & Time', figsize = (24,12), color
= ['green','blue'])
```

```
# # Bearing 2

# In[591]:
#bearing 2 data from all csv
df_bearing_2_graph = DataFrame(data_from_csv, columns=['Date &
Time', 'Bearing 2_X', 'Bearing 2_Y'])
#df_bearing_2_graph.plot(x ='Date & Time', figsize = (24,12), color
= ['green','blue'])

# # Bearing 3
# In[592]:
#bearing 3 data from all csv
df_bearing_3_graph = DataFrame(data_from_csv, columns=['Date &
Time', 'Bearing 3_X', 'Bearing 3_Y'])
#df_bearing_3_graph.plot(x ='Date & Time', figsize = (24,12), color
= ['green','blue'])

# # Bearing 4

# In[593]:
#bearing 4 data from all csv
df_bearing_4_graph = DataFrame(data_from_csv, columns=['Date &
Time', 'Bearing 4_X', 'Bearing 4_Y'])
#df_bearing_4_graph.plot(x ='Date & Time', figsize = (24,12), color
= ['green','blue'])

# # Bearings from dataframe to csv
# In[594]:
#bearing 1 create dataframe file
Date_time_bearing_1 = data_from_csv["Date & Time"]
vibration_bearing_1_X = data_from_csv["Bearing 1_X"]
vibration_bearing_1_Y = data_from_csv["Bearing 1_Y"]
dict = {'Date & Time': Date_time_bearing_1, 'Bearing 1 X axis':
vibration_bearing_1_X, 'Bearing 1 Y axis': vibration_bearing_1_Y}
```

```
df_bearing_1 = pd.DataFrame(dict)
print("created bearing_1 dataframe")


#bearing 2 create dataframe file
Date_time_bearing_2 = data_from_csv["Date & Time"]
vibration_bearing_2_X = data_from_csv["Bearing 2_X"]
vibration_bearing_2_Y = data_from_csv["Bearing 2_Y"]
dict = {'Date & Time': Date_time_bearing_2, 'Bearing 2 X axis':
vibration_bearing_2_X, 'Bearing 2 Y axis': vibration_bearing_2_Y}
df_bearing_2 = pd.DataFrame(dict)
print("created bearing_2 dataframe")

#bearing 3 create dataframe file
Date_time_bearing_3 = data_from_csv["Date & Time"]
vibration_bearing_3_X = data_from_csv["Bearing 3_X"]
vibration_bearing_3_Y = data_from_csv["Bearing 3_Y"]
dict = {'Date & Time': Date_time_bearing_3, 'Bearing 3 X axis':
vibration_bearing_3_X, 'Bearing 3 Y axis': vibration_bearing_3_Y}
df_bearing_3 = pd.DataFrame(dict)
print("created bearing_3 dataframe")

#bearing 4 create dataframe file
Date_time_bearing_4 = data_from_csv["Date & Time"]
vibration_bearing_4_X = data_from_csv["Bearing 4_X"]
vibration_bearing_4_Y = data_from_csv["Bearing 4_Y"]
dict = {'Date & Time': Date_time_bearing_4, 'Bearing 4 X axis':
vibration_bearing_4_X, 'Bearing 4 Y axis': vibration_bearing_4_Y}
df_bearing_4 = pd.DataFrame(dict)
print("created bearing_4 dataframe")
```

```
#convert into csv each
df_bearing_1.to_csv(r'C:\Users\Atlas Printing\Predictive Maintenance
Model\ATLAS samples\Bearing 1 ATLAS data.csv', index = False, header
= True)
print("created bearing_1 csv file !!")
df_bearing_2.to_csv(r'C:\Users\Atlas Printing\Predictive Maintenance
Model\ATLAS samples\Bearing 2 ATLAS data.csv', index = False, header
= True)
print("created bearing_2 csv file !!")
df_bearing_3.to_csv(r'C:\Users\Atlas Printing\Predictive Maintenance
Model\ATLAS samples\Bearing 3 ATLAS data.csv', index = False, header
= True)
print("created bearing_3 csv file !!")
df_bearing_4.to_csv(r'C:\Predictive Maintenance Model\FYP\ATLAS
samples\Bearing 4 ATLAS data.csv', index = False, header = True)
print("created bearing_4 csv file !!")

# # Select Train & Test by index no

# In[595]:
index_num = int((2/3)*number_of_rows_merged_data)
            #index num selecting
train_date_time = df_merged_data[ : index_num]
print("Date and Time Classified")
train_b_1_end = index_num
            #define dataset b 1
train_data_b_1 = df_bearing_1[  : train_b_1_end]
print("Bearing 1 train data classified")

#bearing 2
            #index num selecting
train_b_2_end = index_num

            #define dataset b 2
```

```
train_data_b_2 = df_bearing_2[  : train_b_2_end]
print("Bearing 2 train data classified")

#bearing 3
            #index num selecting
train_b_3_end = index_num



            #define dataset b 1
train_data_b_3 = df_bearing_3[  : train_b_3_end]
print("Bearing 3 train data classified")

#bearing 4
            #index num selecting
train_b_4_end = index_num

            #define dataset b 1
train_data_b_4 = df_bearing_4[  : train_b_4_end]
print("Bearing 4 train data classified")

# In[596]:
col_list_train = ["Bearing 1_X", "Bearing 1_Y", "Bearing 2_X",
"Bearing 2_Y", "Bearing 3_X", "Bearing 3_Y", "Bearing 4_X", "Bearing
4_Y"]
df_train = pd.read_csv(most_recent, usecols=col_list_train)

train_b_all_end = index_num

dataset_train_b_all = df_train[ :train_b_all_end]
print("Bearing all test, train data classified")
#dataset_train_b_all.head()
```

```
# In[597]:
#get Date & time from train datasets
col_list_train_date = ["Date & Time"]
df_train_date = pd.read_csv(most_recent,
usecols=col_list_train_date)
dataset_train_date_time = df_train_date[:train_b_all_end]
dataset_train_date_time =
pd.to_datetime(dataset_train_date_time.stack()).unstack()
#dataset_train_date_time.head()

# # Preprocess data

# In[598]:
scaler = preprocessing.MinMaxScaler()

#bearing  train
train_data_b_all_scaled =
pd.DataFrame(scaler.fit_transform(dataset_train_b_all),

columns=dataset_train_b_all.columns,

index=dataset_train_b_all.index)

#Random shuffle training data
train_data_b_all_scaled.sample(frac=1)
# # PCA  configuration

# In[599]:
from sklearn.decomposition import PCA
pca = PCA(n_components=2, svd_solver= 'full')
X_train_PCA = pca.fit_transform(np.array(train_data_b_all_scaled))
X_train_PCA = pd.DataFrame(X_train_PCA)
X_train_PCA.index = train_data_b_all_scaled.index
```

```
# # Defining functions related with PCA

# # Calculate the covariance matrix

# In[600]:
def cov_matrix(data, verbose=False):
    covariance_matrix = np.cov(data, rowvar=False)
    if is_pos_def(covariance_matrix):
        inv_covariance_matrix = np.linalg.inv(covariance_matrix)
        if is_pos_def(inv_covariance_matrix):
            return covariance_matrix, inv_covariance_matrix
        else:
            print("Error: Inverse of Covariance Matrix is not
positive definite!")
    else:
        print("Error: Covariance Matrix is not positive definite!")

# # Calculate the Mahalanobis distance

# In[601]:
def Mahalanobis_Dist(inv_cov_matrix, mean_distr, data,
verbose=False):
    inv_covariance_matrix = inv_cov_matrix
    vars_mean = mean_distr
    diff = data - vars_mean
    md = []
    for i in range(len(diff)):

md.append(np.sqrt(diff[i].dot(inv_covariance_matrix).dot(diff[i])))
    return md
```

```
# # Detecting Mahalanobis distance outliers

# In[602]:
def MD_detect_Outliers(dist, extreme=False, verbose=False):
    k = 3. if extreme else 2.
    threshold = np.mean(dist) * k
    outliers = []
    for i in range(len(dist)):
        if dist[i] >= threshold:
            outliers.append(i)
    return np.array(outliers)

# # Threshold value calculation

# In[603]:
def MD_threshold(dist, extreme=False, verbose=False):
    k = 3. if extreme else 2.
    threshold = np.mean(dist) * k
    return threshold

# # Check if matrix is positive definite

# In[604]:
def is_pos_def(A):
    if np.allclose(A, A.T):
        try:
            np.linalg.cholesky(A)
            return True
        except np.linalg.LinAlgError:
            return False
    else:
        return False
```

```
# # Define train/test set
# In[605]:
data_train = np.array(X_train_PCA.values)

# # Calculate the covariance matrix and its inverse
# In[606]:
cov_matrix, inv_cov_matrix  = cov_matrix(data_train)

# # Mahalanobis distance to datapoints

# In[607]:
mean_distr = data_train.mean(axis=0)

# In[608]:
dist_train = Mahalanobis_Dist(inv_cov_matrix, mean_distr,
data_train, verbose=True)
threshold = MD_threshold(dist_train, extreme = True)

# # Threshold value for flagging an anomaly

# In[609]:
#plt.figure()
#sns.distplot(np.square(dist_train),
             #bins = 29,
             #kde= False);
#plt.rcParams["figure.figsize"] = (24, 12)
#plt.xlim([0.0,15])

# # Visualize the Mahalanobis distance

# In[610]:
#plt.figure()
#sns.distplot(dist_train,
             #bins = 29,
```

```
             #kde= True,
            #color = 'green');
#plt.xlim([0.0,5])
#plt.rcParams["figure.figsize"] = (24, 12)
#plt.xlabel('Mahalanobis dist')

# # Flag the anomaly

# In[611]:
anomaly_train = pd.DataFrame()
anomaly_train['Mob dist']= dist_train
anomaly_train['Thresh'] = threshold
anomaly_train['Anomaly'] = anomaly_train['Mob dist'] >
anomaly_train['Thresh']
anomaly_train.index = X_train_PCA.index
anomaly = pd.DataFrame()

# In[612]:
anom_train_processed = anomaly_train
anom_train_processed['Date & Time']  = dataset_train_date_time
anom_train_processed = anom_train_processed[['Date & Time', 'Mob
dist', 'Thresh', 'Anomaly']]
#anom_train_processed.head()

# In[613]:
anomaly_alldata = pd.concat([anomaly_train, anomaly])
#anomaly_alldata.info()

# # Visualize PCA model on train data
# In[614]:
anom_train_processed.to_csv(r'C:\Users\Atlas Printing\Predictive
Maintenance Model\ATLAS anomaly\Anomaly_distance_train_data.csv',
index = False, header = True)
```

```
# In[615]:
df_anomaly_alldata = DataFrame(anom_train_processed, columns=['Mob
dist', 'Thresh'])
#df_anomaly_alldata.plot(logy=True, figsize = (24,12), ylim = [1e-
1,1e3], color = ['green','red'])

# # Raw Flag
# In[616]:
x_flag = anom_train_processed.index
y_flag = anom_train_processed["Anomaly"]
#plt.rcParams["figure.figsize"] = (24, 12)
#plt.plot(x_flag, y_flag)
# # Smoothing
# In[617]:
#Smoothing by exponentially weighted window functions and calculate
the exponentially weighted average
com_val = 4.5
smooth_anomaly_mob_dis = df_anomaly_alldata ["Mob dist"].ewm(com=
com_val).mean()
smooth_anomaly_thresh = df_anomaly_alldata ["Thresh"]
smooth_anomaly_anomal_val = smooth_anomaly_mob_dis >
smooth_anomaly_thresh
smooth_anomaly_date_time = anom_train_processed["Date & Time"]
smooth_anomaly_time_stamp = pd.to_datetime(smooth_anomaly_date_time)
smooth_anomaly_dataframe = pd.DataFrame()
anomaly_frame_layout = {
                        "Date & Time" : smooth_anomaly_time_stamp,
                        "Thresh" : smooth_anomaly_thresh,
                        "Mob dist" : smooth_anomaly_mob_dis,
                        "Anomaly" : smooth_anomaly_anomal_val
```

```
                        }
smooth_anomaly_dataframe = pd.concat(anomaly_frame_layout, axis = 1)
smooth_anomaly_dataframe.set_index('Date & Time')
smooth_anomaly_dataframe.to_csv(r'C:\Users\Atlas Printing\Predictive
Maintenance Model\ATLAS
anomaly\Anomaly_distance_smoothed_trained_data.csv', index = False,
header = True)

# In[618]:
#smooth_anomaly_dataframe.head()

# In[619]:
df_smooth = DataFrame(smooth_anomaly_dataframe, columns=['Mob dist',
'Thresh'])
#df_smooth.plot(logy=True, figsize = (24,12), ylim = [1e-1,1e3],
color = ['green','red'])

# # Smoothed flag

# In[620]:
x_df_smooth_flag = smooth_anomaly_dataframe.index
y_df_smooth_flag = smooth_anomaly_dataframe['Anomaly']
#plt.rcParams["figure.figsize"] = (24, 12)
#plt.plot(x_df_smooth_flag, y_df_smooth_flag)

# In[621]:
visualize_flag_time = data_from_csv['Date & Time']
visualize_flag_anomaly =
smooth_anomaly_dataframe['Anomaly'].astype(int)
dict = {'Date & Time': visualize_flag_time, 'Anomaly':
visualize_flag_anomaly}
visualize_flag = pd.DataFrame(dict)
#visualize_flag.head()
```

```
# In[622]:
#visualize_flag.plot(x ='Date & Time', figsize = (12,3), color =
['blue'])

# # Test part
# In[623]:
#load test data
path = "C:/Users/Atlas Printing/Predictive Maintenance Model/ATLAS
test data/"
test_most_recent = max(glob.iglob("C:/Users/Atlas
Printing/Predictive Maintenance Model/ATLAS test data/*.csv"),
key=os.path.getmtime)

#print(test_most_recent)

# In[624]:
test_data_from_csv = pd.read_csv(test_most_recent)
#test_data_from_csv.head()

# In[625]:
df_test_data = DataFrame(test_data_from_csv, columns=['Bearing 1_X',
'Bearing 1_Y', 'Bearing 2_X', 'Bearing 2_Y', 'Bearing 3_X', 'Bearing
3_Y', 'Bearing 4_X', 'Bearing 4_Y'])
#df_test_data.info()

# In[626]:
#bearing  test
df_test_data = pd.DataFrame(scaler.fit_transform(df_test_data),
                                        columns=df_test_data.columns,
                                        index=df_test_data.index)
# In[627]:
X_test_PCA = pca.transform(np.array(df_test_data))
X_test_PCA = pd.DataFrame(X_test_PCA)
X_test_PCA.index = df_test_data.index
```

```
# In[628]:
data_test = np.array(X_test_PCA.values)
dist_test = Mahalanobis_Dist(inv_cov_matrix, mean_distr, data_test,
verbose=True)
anomaly = pd.DataFrame()

anomaly['Mob dist']= dist_test
anomaly['Thresh'] = threshold

anomaly['Anomaly'] = anomaly['Mob dist'] > anomaly['Thresh']
anomaly.index = X_test_PCA.index
#anomaly.head()

# In[629]:
anom_test_processed = anomaly
anom_test_processed['Date & Time']  = test_data_from_csv['Date &
Time']
anom_test_processed = anom_test_processed[['Date & Time', 'Mob
dist', 'Thresh', 'Anomaly']]
anom_test_processed.head()

# In[630]:
test_data_from_csv

# In[631]:
anom_test_processed.to_csv(r'C:\Users\Atlas Printing\Predictive
Maintenance Model\ATLAS anomaly\Anomaly_distance_test_data.csv',
index = False, header = True)

# In[632]:
#anom_test_processed.info()

# In[633]:
```

```
df_anomaly_test_data = DataFrame(anom_test_processed, columns=['Mob
dist', 'Thresh'])
#df_anomaly_test_data.plot(logy=True, figsize = (24,12), ylim = [1e-
1,1e3], color = ['green','red'])

# In[634]:
com_val = 4.5
smooth_anomaly_mob_dis_test = df_anomaly_test_data["Mob
dist"].ewm(com= com_val).mean()
smooth_anomaly_thresh = anom_test_processed["Thresh"]
smooth_anomaly_anomal_val_test = smooth_anomaly_mob_dis_test >
smooth_anomaly_thresh
smooth_anomaly_date_time_test = anom_test_processed["Date & Time"]
smooth_anomaly_time_stamp_test =
pd.to_datetime(smooth_anomaly_date_time_test)
smooth_anomaly_dataframe_test = pd.DataFrame()
anomaly_frame_layout = {
                        "Date & Time" :
smooth_anomaly_time_stamp_test,
                        "Thresh" : smooth_anomaly_thresh,
                        "Mob dist" : smooth_anomaly_mob_dis_test,
                        "Anomaly" : smooth_anomaly_anomal_val_test
                        }
smooth_anomaly_dataframe_test = pd.concat(anomaly_frame_layout, axis
= 1)
smooth_anomaly_dataframe_test.set_index('Date & Time')
smooth_anomaly_dataframe_test.to_csv(r'C:\Users\Atlas
Printing\Predictive Maintenance Model\ATLAS
anomaly\Anomaly_test_distance_smoothed.csv', index = False, header =
True)
# In[635]:
df_smooth_test = DataFrame(smooth_anomaly_dataframe_test,
columns=['Mob dist', 'Thresh'])
```

```
#df_smooth_test.plot(logy=True, figsize = (24,12), ylim = [1e-
1,1e3], color = ['green','red'])

# In[636]:
x_df_smooth_flag_test = smooth_anomaly_dataframe_test.index
y_df_smooth_flag_test = smooth_anomaly_dataframe_test['Anomaly']
plt.rcParams["figure.figsize"] = (24, 12)
#plt.plot(x_df_smooth_flag_test, y_df_smooth_flag_test)

# In[637]:
visualize_flag_time_test = test_data_from_csv['Date & Time']
visualize_flag_anomaly_test =
smooth_anomaly_dataframe_test['Anomaly'].astype(int)
dict = {'Date & Time': visualize_flag_time_test, 'Anomaly':
visualize_flag_anomaly_test}
visualize_flag_test = pd.DataFrame(dict)
#visualize_flag_test.head()

# In[638]:
#visualize_flag_test.plot(x ='Date & Time', figsize = (12,3), color
= ['blue'])

# In[639]:
visualize_test_txt_test = smooth_anomaly_dataframe_test
#visualize_test_txt_test

# In[640]:
edge_detected_anomaly_test =
visualize_test_txt_test[visualize_test_txt_test.Anomaly.diff().filln
a(False)]
#edge_detected_anomaly_test
anom_col_test = visualize_test_txt_test['Anomaly']

if(len(edge_detected_anomaly_test) == 0):
```

```
    if(anom_col_test[0] == True):
        print("Rising flag detected")
        test_date_flag_list = visualize_test_txt_test['Date & Time']
#Banana
        test_date_flag = test_date_flag_list[0]
        last_date_flag = test_date_flag_list.max()
        flag_val = 1
    else:
        print("Just Normal...")
        flag_val = 0
else:
    print("Normal edge logging")

# In[641]:
rising_data_test = pd.DataFrame()
falling_data_test = pd.DataFrame()

rising_data_test =
edge_detected_anomaly_test[edge_detected_anomaly_test['Anomaly']]
rising_data_df_test = DataFrame(rising_data_test)
len_rising_test = len(rising_data_df_test)
rising_data_df_test

# In[642]:
falling_data_test =
edge_detected_anomaly_test[~edge_detected_anomaly_test['Anomaly']]
falling_data_df_test = DataFrame(falling_data_test)
len_falling_test = len(falling_data_df_test)
falling_data_df_test

# In[643]:
if(len(rising_data_df_test) > 0):
    last_day_rising_test = rising_data_df_test['Date & Time'].max()
```

```
    last_day_max_falling_test = falling_data_df_test['Date &
Time'].max()
    last_day_test = visualize_test_txt_test['Date & Time'].max()
    print("Rising latest point = " + str(last_day_rising_test))
    test_date_flag = last_day_rising_test
    flag_val = 1
    if last_day_max_falling_test > last_day_rising_test:
        print("Falling latest point = " +
str(last_day_max_falling_test))
        last_day_falling_test = last_day_max_falling_test
    else:
        print("Only rising!!")
        last_day_falling_test = last_day_test_test
elif(len(rising_data_df_test) == 0 or len(falling_data_df_test) ==
0):
    print("banana")
    if(anom_col_test[0] == True):
        print("Rising flag detected")
        test_date_flag_list = visualize_test_txt_test['Date & Time']
#Banana
        test_date_flag = test_date_flag_list[0]
        flag_val = 1
    else:
        print("Just Normal...")
        flag_val = 0
else:
    print("Normal edge logging")

# In[644]:
if len_falling_test == 0 and len_rising_test == 1:
    print("There is only rising....")
    rising_only_datetime_test = last_day_rising_test
    time_duartion = last_day_falling_test -
rising_only_datetime_test
```

```
    print("Time duration : "+ str(time_duartion))
elif len_falling_test > 0 and len_rising_test >= 1:
    print("There is flagging periods...")
    time_duartion = last_day_falling_test - last_day_rising_test
    print("Time duration : "+ str(time_duartion))
elif len_falling_test == 0 and len_rising_test == 0 and
anom_col_test[0] == True:
    print("Rising Flag Detected in special manner")
    time_duration = last_date_flag - test_date_flag
time_duration_hrs = time_duration / np.timedelta64(1, 'h')
#print(time_duration_hrs)



# In[645]:
if(time_duration_hrs >= 1.5):
    print("Flag triggered!!!!!!")
    final_val_triggered = True
else:
    final_val_triggered = False

# In[646]:
#save flagged time in csv temp
data = {'Date & Time': [test_date_flag]}
flag_date_df = pd.DataFrame(data)
flag_date_df.to_csv(r'C:\Users\Atlas Printing\Predictive Maintenance
Model\ATLAS flag date\Flag_Time.csv', index = False, header = True)
# In[647]:
only_rising = False
only_falling = False
both_rising = False
special_rising = False
# In[648]:
if (final_val_triggered == True):
    if len_falling_test == 0 and len_rising_test == 1:
```

```
        print("Only rising point detected")
        only_rising_df = rising_data_df_test.iloc[-1]
        df_o_rising = int(only_rising_df.toframe().T.index.values)
        only_rising_df1 = df_bearing_1_graph[df_o_rising_df :]
        only_rising_df2 = df_bearing_2_graph[df_o_rising_df :]
        only_rising_df3 = df_bearing_3_graph[df_o_rising_df :]
        only_rising_df4 = df_bearing_4_graph[df_o_rising_df :]
        only_rising = True
        only_falling = False
        both_rising = False
        special_rising = False
    elif len_falling_test > 0 and len_rising >= 1:
        print("Flagging period with rising and falling and
detected")
        both_rising = rising_data_df_test.iloc[-1]
        df_b_rising = int(both_rising.to_frame().T.index.values)
        both_falling = falling_data_df.iloc[-1]
        df_b_falling = int(both_falling.to_frame().T.index.values)
        both_falling_df1 = df_bearing_1_graph[df_b_rising :
df_b_falling]
        both_falling_df2 = df_bearing_2_graph[df_b_rising :
df_b_falling]
        both_falling_df3 = df_bearing_3_graph[df_b_rising :
df_b_falling]
        both_falling_df4 = df_bearing_4_graph[df_b_rising :
df_b_falling]
        only_rising = False
        only_falling = False
        both_rising = True
        special_rising = False
    elif len_falling_test == 0 and len_rising_test == 0 and
anom_col_test[0] == True:
        print("Rising Flag Detected in special manner")
        special_rising =anom_test_processed.iloc[0]
```

```
        df_s_rising = int(special_rising.to_frame().T.index.values)
        special_falling = anom_test_processed.iloc[-1]
        df_s_falling =
int(special_falling.to_frame().T.index.values)
        special_falling_df1 = df_bearing_1_graph[df_s_rising :
df_s_falling]
        special_falling_df2 = df_bearing_2_graph[df_s_rising :
df_s_falling]
        special_falling_df3 = df_bearing_3_graph[df_s_rising :
df_s_falling]
        special_falling_df4 = df_bearing_4_graph[df_s_rising :
df_s_falling]
        only_rising = False
        only_falling = False
        both_rising = False
        special_rising = True
    elif len_falling_test >= 0 and len_rising_test == 0 :
        print("Only falling point detected")
        only_falling_df = falling_data_df_test.iloc[-1]
        df_o_falling = int(only_falling_df.toframe().T.index.values)
        only_falling_df1 = df_bearing_1_graph[df_o_falling_df :]
        only_falling_df2 = df_bearing_2_graph[df_o_falling_df :]
        only_falling_df3 = df_bearing_3_graph[df_o_falling_df :]
        only_falling_df4 = df_bearing_4_graph[df_o_falling_df :]
        only_rising = False
        only_falling = True
        both_rising = False
        special_rising = False
    elif len_falling_test == 0 and len_rising_test == 0:
        print("No failure alert")
        only_rising = False
        only_falling = False
        both_rising = False
        special_rising = False
```

```
else:
    print("No failure alert")
# In[649]:
if only_rising == True and both_rising == False and special_rising
== False and only_falling == False:
    val_1_x = float((only_rising_df1.max()).to_frame().T['Bearing
1_X'])
    val_2_x = float((only_rising_df2.max()).to_frame().T['Bearing
2_X'])
    val_3_x = float((only_rising_df3.max()).to_frame().T['Bearing
3_X'])
    val_4_x = float((only_rising_df4.max()).to_frame().T['Bearing
4_X'])
    val_1_y = float((only_rising_df1.max()).to_frame().T['Bearing
1_Y'])
    val_2_y = float((only_rising_df2.max()).to_frame().T['Bearing
2_Y'])
    val_3_y = float((only_rising_df3.max()).to_frame().T['Bearing
3_Y'])
    val_4_y = float((only_rising_df4.max()).to_frame().T['Bearing
4_Y'])

elif only_rising == False and both_rising == True and special_rising
== False and only_falling == False:
    val_1_x = float((both_falling_df1.max()).to_frame().T['Bearing
1_X'])
    val_2_x = float((both_falling_df2.max()).to_frame().T['Bearing
2_X'])
    val_3_x = float((both_falling_df3.max()).to_frame().T['Bearing
3_X'])
    val_4_x = float((both_falling_df4.max()).to_frame().T['Bearing
4_X'])
```

```
    val_1_y = float((both_falling_df1.max()).to_frame().T['Bearing
1_Y'])
    val_2_y = float((both_falling_df2.max()).to_frame().T['Bearing
2_Y'])
    val_3_y = float((both_falling_df3.max()).to_frame().T['Bearing
3_Y'])
    val_4_y = float((both_falling_df4.max()).to_frame().T['Bearing
4_Y'])
elif only_rising == False and both_rising == False and
special_rising == True and only_falling == False:
    val_1_x =
float((special_falling_df1.max()).to_frame().T['Bearing 1_X'])
    val_2_x =
float((special_falling_df2.max()).to_frame().T['Bearing 2_X'])
    val_3_x =
float((special_falling_df3.max()).to_frame().T['Bearing 3_X'])
    val_4_x =
float((special_falling_df4.max()).to_frame().T['Bearing 4_X'])
    val_1_y =
float((special_falling_df1.max()).to_frame().T['Bearing 1_Y'])
    val_2_y =
float((special_falling_df2.max()).to_frame().T['Bearing 2_Y'])
    val_3_y =
float((special_falling_df3.max()).to_frame().T['Bearing 3_Y'])
    val_4_y =
float((special_falling_df4.max()).to_frame().T['Bearing 4_Y'])
elif only_rising == False and both_rising == False and
special_rising == False and only_falling == True:
    val_1_x = float((only_falling_df1.max()).to_frame().T['Bearing
1_X'])
    val_2_x = float((only_falling_df2.max()).to_frame().T['Bearing
2_X'])
    val_3_x = float((only_falling_df3.max()).to_frame().T['Bearing
3_X'])
```

```
    val_4_x = float((only_falling_df4.max()).to_frame().T['Bearing
4_X'])
    val_1_y = float((only_falling_df1.max()).to_frame().T['Bearing
1_Y'])
    val_2_y = float((only_falling_df2.max()).to_frame().T['Bearing
2_Y'])
    val_3_y = float((only_falling_df3.max()).to_frame().T['Bearing
3_Y'])
    val_4_y = float((only_falling_df4.max()).to_frame().T['Bearing
4_Y'])

# In[650]:
thred_val = float(0.1355)

# In[651]:
avg_1 = float((val_1_x + val_1_y)/2)
avg_2 = float((val_2_x + val_2_y)/2)
avg_3 = float((val_3_x + val_3_y)/2)
avg_4 = float((val_4_x + val_4_y)/2)

fail_1 = False
fail_2 = False
fail_3 = False
fail_4 = False

# In[652]:
if avg_1 >= thred_val:
    print("Bearing 1 going to be fail")
    fail_1 = True
if avg_2 >= thred_val:
    print("Bearing 2 going to be fail")
    fail_2 = True
if avg_3 >= thred_val:
    print("Bearing 3 going to be fail")
```

```
    fail_3 = True
if avg_4 >= thred_val:
    print("Bearing 4 going to be fail")
    fail_4 = True
if avg_1 < thred_val and avg_2 < thred_val and avg_3 < thred_val and
avg_4 < thred_val:
    print("No failure detected")
    fail_1 = False
    fail_2 = False
    fail_3 = False
    fail_4 = False

# In[653]:
#load maintenance csv
path = "C:/Users/Atlas Printing/Predictive Maintenance Model/ATLAS
maintenance"
most_recent_maintenance = max(glob.iglob("C:/Users/Atlas
Printing/Predictive Maintenance Model/ATLAS maintenance/*.csv"),
key=os.path.getmtime)
data_from_csv_maintenance = pd.read_csv(most_recent_maintenance)
# In[654]:
str_repair =""

# In[655]:
if fail_1 == True:
    str_repair = str_repair + "1 "
if fail_2 == True:
    str_repair = str_repair + "2 "
if fail_3 == True:
    str_repair = str_repair + "3 "
if fail_4 == True:
    str_repair = str_repair + "4 "
```

```
# In[656]:
str_repair
# In[657]:
data_from_csv_maintenance
# In[658]:
latest_datetime_maintenance =
data_from_csv_maintenance['Date'].max()
# In[659]:
selected_index =
data_from_csv_maintenance[data_from_csv_maintenance['Date'] ==
latest_datetime_maintenance].index[0]

# In[660]:
latest_datetime_maintenance_datetime =
pd.to_datetime(latest_datetime_maintenance)

# In[661]:
#data_from_csv_maintenance=
latest_datetime_maintenance[latest_datetime_maintenance ==
latest_datetime_maintenance]
# In[662]:
data_from_csv_maintenance['Fault Bearing / s'].loc[selected_index] =
str(str_repair)
# In[663]:
data_from_csv_maintenance
```

Data acquisition

```
import glob
import os
import time
import pandas as pd
import datetime
from pymodbus.client.sync import ModbusTcpClient
```

```
import numpy as np
import csv


PLC_IP = "192.168.1.112"
client = ModbusTcpClient(PLC_IP)
date_time_file_name =
datetime.datetime.now().strftime('%Y_%m_%d_%H_%M_%S')
date_time_file_name_str = str(date_time_file_name) + '.csv'
date_time_file_name_loc = "r'C:/Users/Atlas Printing/Predictive
Maintenance Model/ATLAS test data/Test Data/'" +
date_time_file_name_str


file_df = pd.DataFrame(
    columns=['Date & Time', 'Bearing 1_X', 'Bearing 1_Y', 'Bearing
2_X', 'Bearing 2_Y', 'Bearing 3_X', 'Bearing 3_Y'])


file_df.to_csv(
    r'C:\Users\Atlas Printing\Predictive Maintenance Model\ATLAS
test data\Test Data\read_data.csv', index=False,
    header=True)
for x in range(720):  # 2h
    date_time = datetime.datetime.now()


    reading_1 = client.read_holding_registers(512, 1, unit=1)
    reading_2 = client.read_holding_registers(513, 1, unit=1)
    reading_3 = client.read_holding_registers(514, 1, unit=1)
    reading_4 = client.read_holding_registers(515, 1, unit=1)
    reading_5 = client.read_holding_registers(516, 1, unit=1)
    reading_6 = client.read_holding_registers(517, 1, unit=1)
    result_1_br = str(reading_1.registers)
    result_2_br = str(reading_2.registers)
    result_3_br = str(reading_3.registers)
    result_4_br = str(reading_4.registers)
    result_5_br = str(reading_5.registers)
```

```
    result_6_br = str(reading_6.registers)


    result_1 = result_1_br.strip("[]")
    result_2 = result_2_br.strip("[]")
    result_3 = result_3_br.strip("[]")
    result_4 = result_4_br.strip("[]")
    result_5 = result_5_br.strip("[]")
    result_6 = result_6_br.strip("[]")


    element = pd.DataFrame({"Date & Time": [date_time],
                            "Bearing 1_X": [result_1],
                            "Bearing 1_Y": [result_2],
                            "Bearing 2_X": [result_3],
                            "Bearing 2_Y": [result_4],
                            "Bearing 3_X": [result_5],
                            "Bearing 3_Y": [result_6],
                            })
    # print(element)
    element.to_csv(r'C:\Users\Atlas
Printing\Desktop\TCP\TCP\element_temp.csv', index=False,
header=True)
    most_recent_file_df = max(
        glob.iglob("C:/Users/Atlas Printing/Predictive Maintenance
Model/ATLAS test data/Test Data/*.csv"),
        key=os.path.getmtime)
    file_df = pd.read_csv(most_recent_file_df)


    element_series = pd.Series(element.iloc[0])
    file_df.loc[x] = element_series
    print(file_df)
    # print(date_time_file_name_str)
    file_df.to_csv(
        r'C:\Users\Atlas Printing\Predictive Maintenance Model\ATLAS
test data\Test Data\read_data.csv', index=False,
```

```
        header=True)


    most_recent_real_time = max(
        glob.iglob("C:/Users/Atlas Printing/Predictive Maintenance
Model/ATLAS test data/Test Data/*.csv"),
        key=os.path.getmtime)


    real_time_df = pd.read_csv(most_recent_real_time)
    real_time_np = np.asarray(real_time_df)
    real_time_df.to_csv(date_time_file_name_str, index=False,
header=True)
    # print(element)
    time.sleep(10)
else:
    print (2h period data logging complete")
```