MEMS Fault Detection with LoRA Network Data Storage

1st FULCO GIAMMARIO
MAT. 191071

***Abstract*—In this paper the problem of detecting malfuction of a system equipped with MEMS accelerometer is analyzed. Literature offers many solutions to this problem and the low cost of MEMS sensors is the main advantage of the approach. Trying to extend this kind of solution to the case when a LoRa Network is available to store data and provide the possibility to have an insight of malfunctioning server side.**

I. INTRODUCTION

The aim of the project is the development of an embedded system able to exploit soft-realtime accelerometer acquisition and wireless communication, taking advantage of the LoRaWAN network potential. Parameters that are collected by sensors are processed through the ARM standard FFT libraries. The FFT is then analyzed to find possible malfunctioning of the system, if so a message is sent through the LoRa network and an Alarm will trigger. The device will communicate with a gateway, which should collect the messages sent from anywhere under the antenna coverage. To perform this experiment we present a 3D printed shaker system, which will be set in a known a priori state to collect datas and parameters needed to perform the fault detection. Using a simple Matlab script the collected spectrum will be analyzed to find suitable references that will be embedded in the MCU code to process the soft-real time datas.

II. STATE OF THE ART

Using frequency domain analysis looking for malfunctioning and fault is a common technique used for many years. In recent years, application of Machine Learning techniques have become a standard but not a must have. In this section two works will be presented, one that includes machine learning techique and one which is based on classical approach to highlight how frequency domain analysis is a good choice for fault detection. In [1] Faults such as misalignment, rotor cracks and rotor to stator rub, are analyzed, comparing various signal processing tools to evaluate performances and are coupled with Finite Element Models to detect malfunctioning. The comparative study is focused towards detecting the least possible level of the fault induced and the computational time consumed. On top of that, A wavelet zooming procedure for identifying smaller fault details is presented and experimental investigation considering different faults is studied. The work by Ince et. al [2] proposes a fast and accurate motor condition monitoring and early fault-detection system using 1-D convolutional neural networks that has an inherent adaptive design to fuse the feature extraction and classification phases of the motor fault detection into a single learning body. The proposed approach is directly applicable to the raw data (signal), and, thus, eliminates the need for a separate feature extraction algorithm resulting in more efficient systems in terms of both speed and hardware. Experimental results obtained using real motor data demonstrate the effectiveness of the proposed method for real-time motor condition monitoring.

III. BOARD

Before proceeding with the discussion of this paper the hardware specification are provided to have a better insight of the process. The main controller board is an STM32 NUCLEO-L476RG (fig. 1). which embeds Arduino R3-Connector, one USB 2.0, one push-button, a reset button and a couple of LEDs. The choice of the micro-controller unit has been done after taking in consideration the effort needed to perform the FFT algorithm which is relevant and other proposed board such as the STM32L072 has not been able to perform, due to memory limitation.

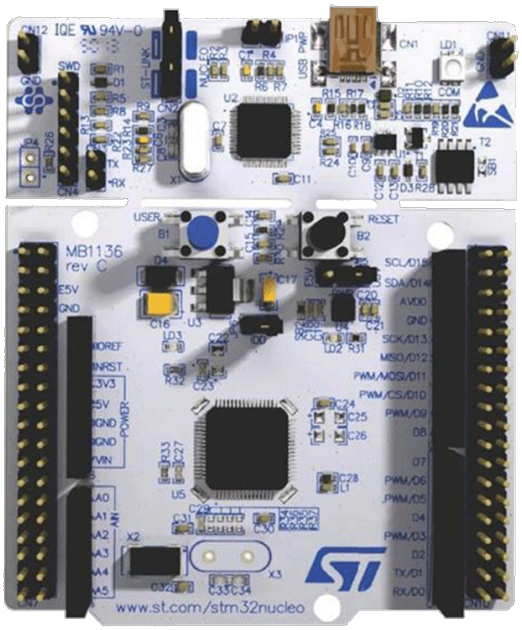

Figure 1: *STM32-NUCLEO-L476RG*

The Sensor board used for collecting the vibration data is a X-NUCLEO-IKS01A1, fig. 2. an integraded board that embed different kinds of Environmental sensors such as a Pressure sensor, a Termomether and a Humidity sensor. Moreover the board contains an IMU which is able to provide acceleration and orientation measurements. The Latter is the one which interests this work so we are gonna give some specification of the accelerometer sensor that is present in the Inertial Measurement unit.

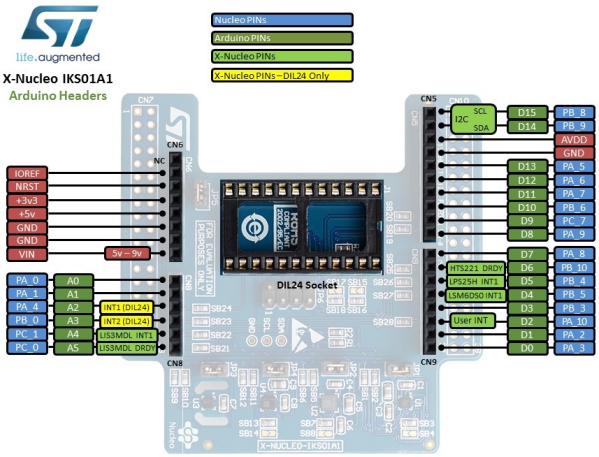

Figure 2: *X-NUCLEO-IKS01A1*

The 3D accelerometer present on the board is the LSM6DS0 gives the user the possibility to choose between different setup such as:

- Full Scale: ∓2g/∓ 4g/∓ 8g
- The Output Data Rate: 19Hz - 962 Hz

The main drawback of this board choice is the impossibility of having acquisition over interrupt: The user needs to read the data when it is ready so, high frequency acquisition arises some problems if the data are to be processed in real time. Although, for this context it is enough since in most Industrial environment application the fault detection needs to be performed after a consistent period of time and not continouosly. Finally, the board which is in charge of transferring data over the LoRA network is the SX1276MB1MAS that embeds 2 antennas (High Frequency and Low Frequency) and is specifically designed to connect and send datas over the LoRA network. The LoRa board is also connected to the MCU via Arduino R3 connector present on the IKS01A1 board, fig 3.

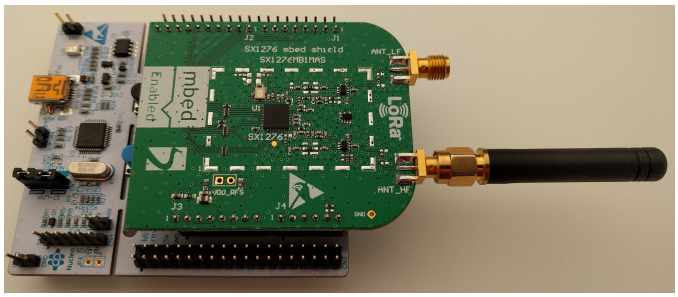

Figure 3: *SX1276MB1MAS*

IV. Firmware Programming

The firmware of the board has been programmed based on the End Node project, found on the STM32CubeExpansion software package: It provides the fundamental libraries and the main software APIs to an user program to be implemented on this board and to transfer data over the LoRA Network. The first challenge has been implementing the FFT libraries in the package to process the acquired data. Firstly, the choice for the numbers of samples to use in the Fourier Transform has been made. Theory give us some limitation, since it has to be a power of 2, while hardware give us other limitations as the maximum ODR of our sensor is 952Hz, The choice has fallen on the maximum ODR available while the number of samples used for the FFT is 1024. With the above choices we fix some boundaries on the resolution and the frequencies available to be inspected: Output samples up to 513 can be analyzed since the FFT provide the a symmetric signal in both negative and positive frequencies. Moreover knowing the Nyquist-Shannon theorem we are bounded to half the sampling frequencies in range so we have a maximum frequency in 476 Hz. As follows from those 2 datas, we can calculate the frequency resolution for our signal which is:

$$\frac{476Hz}{513FFTbins} = 0.927Hz$$

The trade off is clearly between having lower frequency resolution or having lower temporal resolution, this choice has been made to have the better frequency resolution with the available parameters without losing too much in terms of temporal accuracy. The whole firmware designed was structured as follow: after the initialization section, including the system peripherals, system clock, the LoRa network module and the sensors connected, a timer has been set, which purpose involves the triggering of an interrupt useful to wake the system. Indeed, as the program enters the infinite loop, a call of a low power manager function will set the board in stop mode, assuring a very low power consumption, until the timer resets, making the microcontroller execute the callback function. At the end of the execution of this function, the button will be re-initialized and the board will return to the low power state, waiting for the next interrupt. The function which has been rewritten for the purpose of this experiment will enter a reading cycle, acquiring a predefinite number of data from the sensors,1024 as said in previously, before launching the

FFT procedure which will compute the frequency response of the system and then pass them to the fault detection algorithm. Based on the results given by the algorithm, the system will proceed in sending the message over the LoRa network: in this way we can inform the gateway that there has been a malfunction in the system or else that everything is running smoothly. The LoRa APIs useful to send messages are called and then the board switches back to Low Power Mode, fig. 4. depicts the LoRa firmware flow chart.

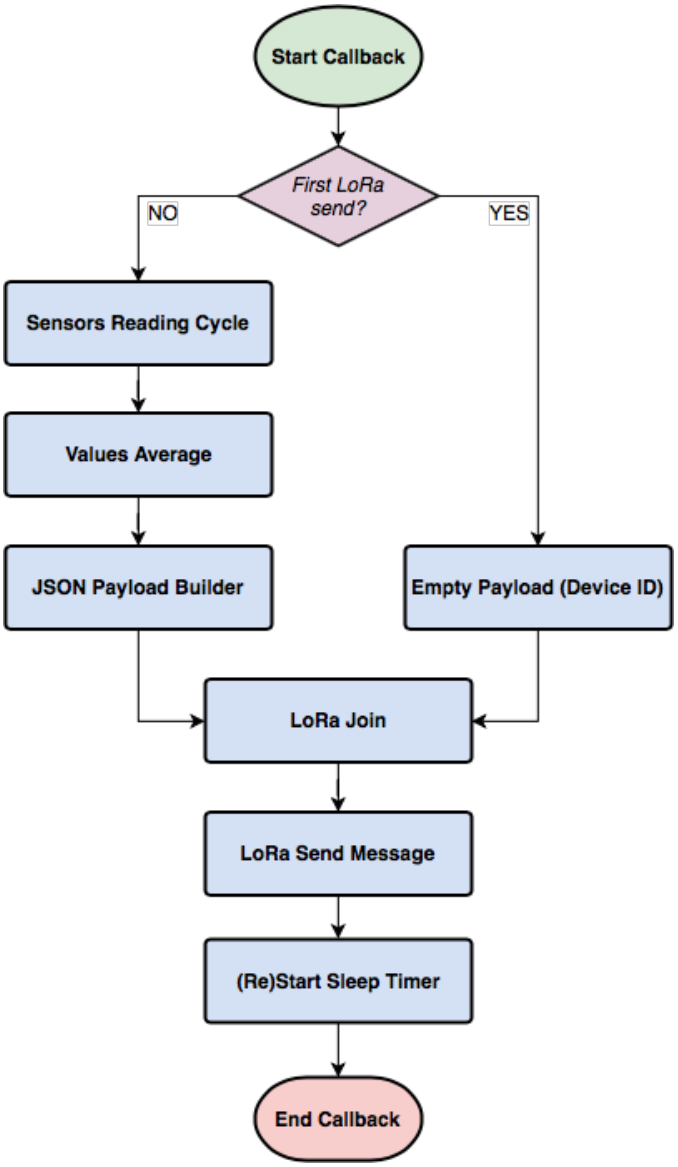

Figure 4: *LoRa Flowchart*

V. LoRaWAN Network Configuration

Due to the encrypted nature of LoRaWAN packets, the device needs to be registred within the network gateway wich will receive the datas. In order to do so the device has to be created inside the LoRa server, using keys that can be found in Commissioning.h file: Device EUI, Application EUI, Application Key, Device Address, Network Session Key and Application Session Key have been set accordingly on the gateway platform. The LoRa Class chosen is the A,

that allows to send packets at any time and limits the receiver to send during pre-scheduled time windows or after a send; however the device has been programmed for unilateral communication, so it'll never wait to receive a packet from the gateway. A parameter chosen in the firmware code is the data rate, that implicitly involves the spreading factor, which influences the time of flight of the frame transmitted: indeed, the data rate chosen, DR 6, refers to the largest possible bandwith available, with this choice we limit the range of transmission, but we gain in terms of payload since we will be able to send 200 bytes of datas with a single message. This setting has been choosen keeping in consideration that possible real application of this experiment will not involve long range transmissions as that should not be needed. Since the device operates under Europe Region, the wireless band associated is the 863-870 MHz frequency band; European region regulations have specified a limit on the maximum number of packets allowed from each LoRa device per day, within the spectrum of this experiment, this limit will not be respected. However, since fault detection is mostly common in industrial application, we don't need to send packets over the network as often as it could be in other applications. In particular, the packets will be send after a specified time has passed which can be tuned based on the desired application.

VI. 3D printed Shaker

Performing suitable tests on the Fault detection algorithm arises the need to have a device which can give at least some kinds of controlled vibration. To face the task, a simple device has been manufactured with the help of a 3D printer and an electric motor. With the help of a screw which will give eccentricity to the spinning axis of the motor it is possible to generate kind of controlled vibrations. Moreover, it is possible to change the natural frequency and amplitude of the system spectrum simply by relaxing or tensing the screw.

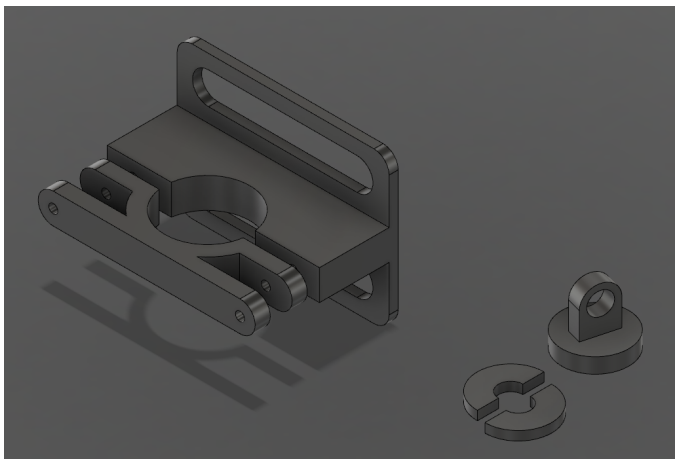

Figure 5: *CAD 3D printed shaker*

As can be seen in fig 5, the 3D printed part of the model is a simple structure to hold the electric motor, and an

accomodation for the screw which will have the responsability to alter the spinning axis. Aside from the case, the shaker has been placed upon a damping base, commonly used to hold MCU in drones applications, that is done to avoid excess movement of the system during the work phase. The Electric motor choosen is a common 30mm diameter DC Motor which works at 12 V as reference. It can spin both clockwise and anti-clockwise depending on the given input current. fig. 6.

Figure 6: *DC Motor*

In fig 7 is shown the constructed system which will be used to perform the experiments for this project.

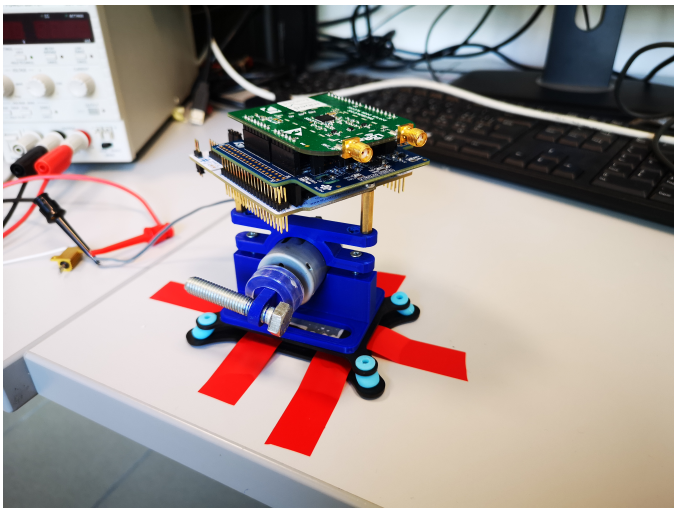

Figure 7: *3D printed Shaker 2*

.

VII. Fault Detection

To address the problem of fault detection a couple of solutions can be taken in consideration. The most common should be the one proposed by Looney in [3] it solves the problem by constructing rectangular windows in the frequencies of interest: This way, it is easy to check if the working frequency of the system has shifted to the side or if the amplitude of the FFT has increased over set bounds, fig. 8 shows the basic principle of this solution.

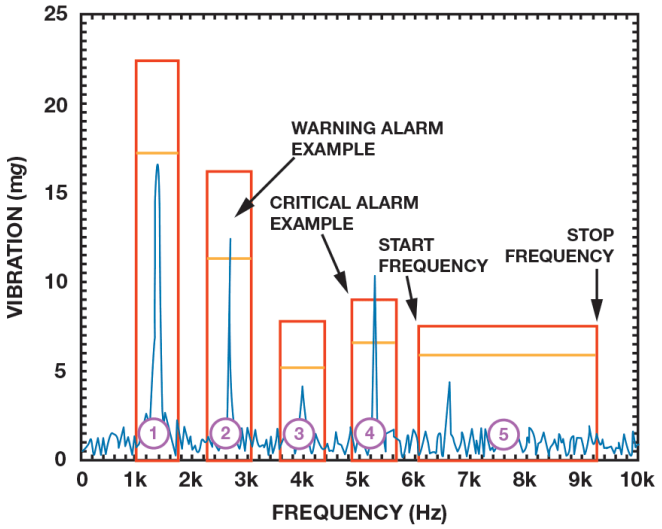

Figure 8: *RWA Algorithm*

Another possible solution is to restrict even more the windows: finding the maximum of the FFT peaks and constructing square or ellyptic windows around a statistically significant mean value for the optimal working conditions is a good way to address the problem fig. 9. In this case the amplitude of the windows is given by the std. deviation of the peaks. This solution can be a good alternative in cases were working condition will slightly differ from nominal ones. For example, tool wear, in particular the tool tip radius increase due to wearing is a well known case in which the frequency response amplitude will slightly decrease or increase depending on the depth of cut with respect to optimal working conditions [4].

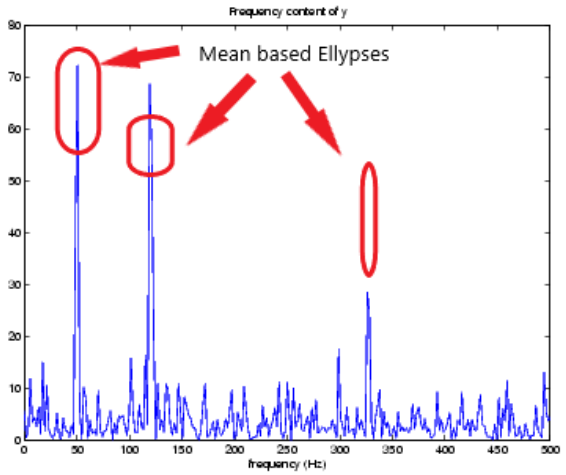

Figure 9: *MVA Algorithm*

In this work, there is an attempt to provide both the algorithm to have a comparation between the two solutions,

for the sake of simplicity let's name the 2 algorithm as, Rectangual Window Algorithm (RWA) the first one explained in the paragraph and Mean Value based Algorithm(MVA) the latter. The RWA windows' height are constructed based on the maximum sample available in the dataset, this will create the smaller windows that will give the warning threshold. That to take in account the worst case scenario, while the width is chosen based on a trial and error process. On top of that, the bigger windows that will signal possible critical situation is constructed choosing, arbitrary, a value increased by 20% of the previous warning windows. The ellypses for the MVA algorithm on the other hand are constructed taking in consideration relevant statistical parameters of the available dataset: The height and width are given by 3 times the standard deviation, and the center is the mean value of the maximum of peaks. It's good to notice that since 3 times the std. deviation implies that 99% of the samples will fall inside the ellipses it is not necessary to have warning thresholds as in the previous case.

VIII. Simulation

Finding the reference for the Fault detection algorithm is the first step for the Simulation of this algorithm, once we choose the reference condition for the screw on the motor, we can start acquiring data, in this experiment, we fixed the input voltage and current given to the motor and the position of the screw to get N=105 data acquisition for the 3 axis of the accelerometer. Voltage of the motor has been fixed at 8 V. The data have been read from the serial port of the STM Nucleo and with a script passed to Matlab. Various parameters have been then calculated: The mean value for the peaks position and amplitude together with the std. deviation. In this paper, a plot of the RMS signal that results from the acquisition for the x axis has been provided in fig. 10 The RMS formula is well known:

$$RMS = \sqrt{\frac{1}{N}\sum_{k=1}^{N} x_k^2}$$

This experiment will be focused on the first 3 harmonic present in the spectrum.

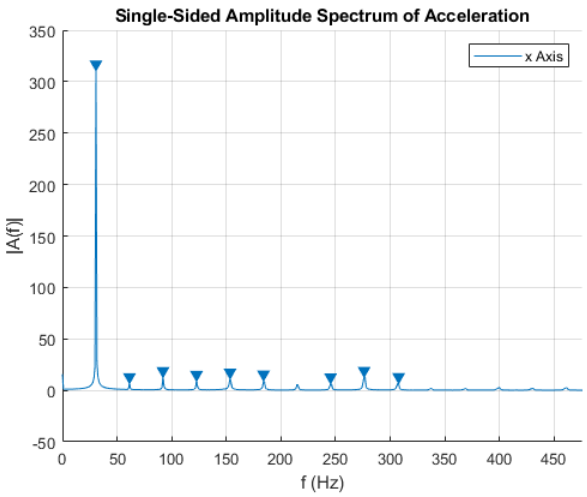

Figure 10: *X axis RMS*

Next step is to define the parameters for the previously defined Fault Detection Algorithm. For the first one RWA, we will construct the windows directly in the Matlab software and hardcoding the parameters to reconstruct them in the STM32 firmware. For the second, we will simply have the mean value and the std. dev of the peaks as parameters which will be used to filter the incoming real time datas.

Something to keep in mind before proceeding is that due to the good stability in frequency given by the power supplier and the poor frequency resolution, in this experiment we got that std. deviation for the peaks position is zero for each signal. To bypass this problem we will assume that the std. deviation for the frequency axis is 1. To give an example on how the algorithms are constructed, the rectangular windows and the ellipses for the 2 algorithms used for the x axis accelerometers are plotted in fig. 11 - 12

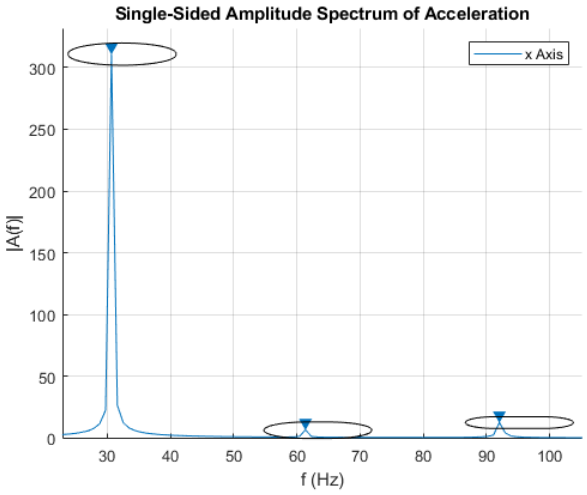

Figure 11: *MVA Algorithm ellipses*

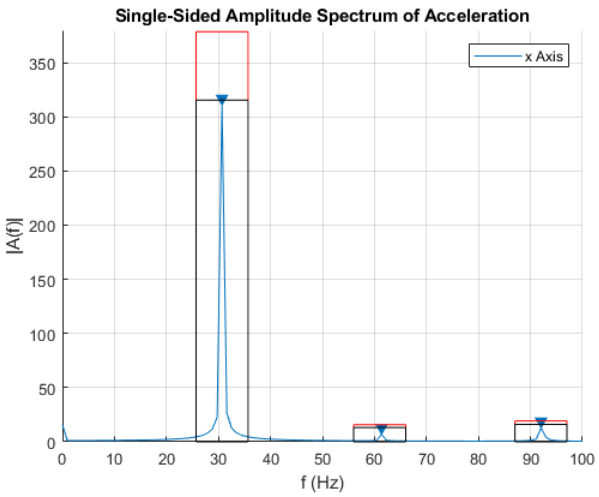

Figure 12: *Rectangular Windows*

Now we will show how the algorithms behave when data acquisition is done in condition different from the reference. We will have some simple simulation setting the input Voltage to 15V and 8V (reference Voltage) in 2 diffent positions of the screw, one is the mentioned reference position which has low axis eccentricity, the second position is opposite to the first one and has a more eccentric spinning axis. As before, for the sake of simplicity we will show some Matlab graph just for the x axis acceleration data fig. 13 - 15

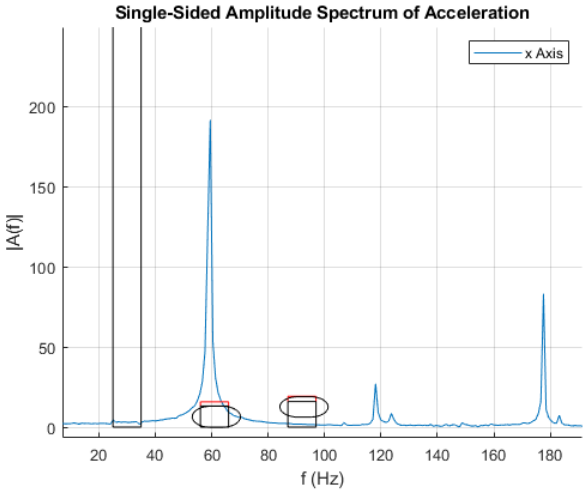

Figure 13: *Reference position, 15V*

The first data acquisition, taken by raising the Voltage to 15V behaves just as expected, since the velocity of the motor is higher but the spinning axis has low exentricity the higher frequency harmonics are more pronounced while the first one is clearly lower. Moreover, the peaks are in general in higher frequency which is also explained by the higher rotating speed of the DC motor. Both the algorithms will give positive response since the all the peaks are out of the ellipses and the rectangular windows. Following this, the screw has been loosen to give the orbit some eccentricity. In the graphs below we can see what happens with same Voltage as reference (8V). What the plot shows that the harmonics have higher amplitude this due to the eccentricity of the spinning axis which will cause the system to shake harder than before. It's good to notice that restoring the initial Voltage, re-established the position of the harmonics, a deduction that can be made is that loosening the screw will only harm the amplitude of the harmonics, while changing the imput Voltage will modify their position. Both algorithm will detect a fault in this case.

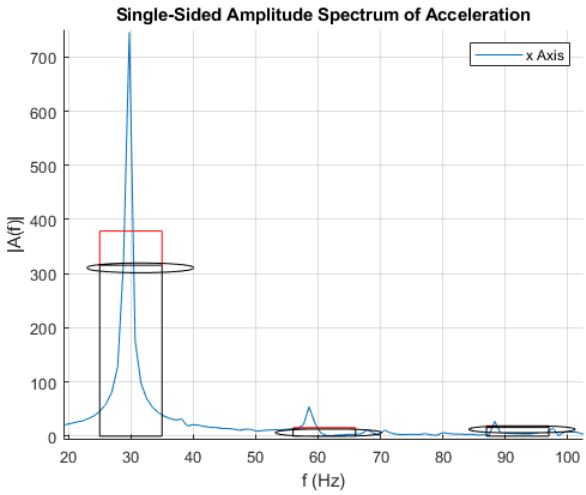

Figure 14: *Eccentric Axis, 8V*

Raising the voltage again to 15V will be the next study case we propose for this experiment. Again speeding up the motor velocity will shift the peaks of harmonics to the right, and shifting the axis will increase the amplitude of vibrations. In this case too the 2 algorithms will have similar performances, detecting problems in all the available harmonics.

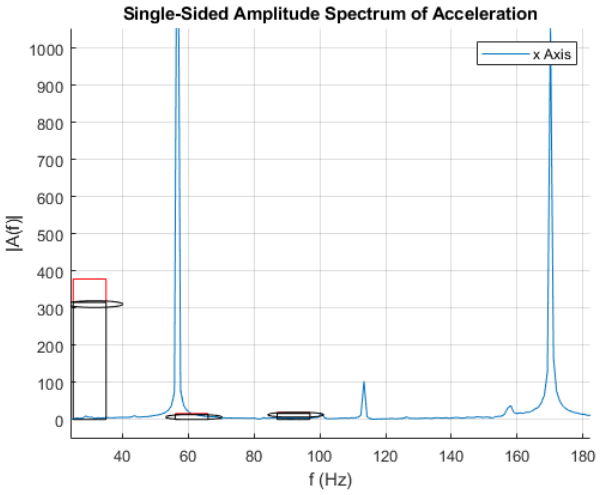

Figure 15: *Eccentric axis, 15V*

To figure out a case where the two algorithms behave differently we will try to fix the input voltage and balancing

even more the spinning axis to see how the ellipses will work way differently than rectangular windows. In fig. 16 is shown how in this case the algorithms will work in different ways. The first harmonic will be rendered as a problem from the MVA point of view, while RWA will just pass it a normal behaviour. Considering different applications, this can be a pro or a cons.

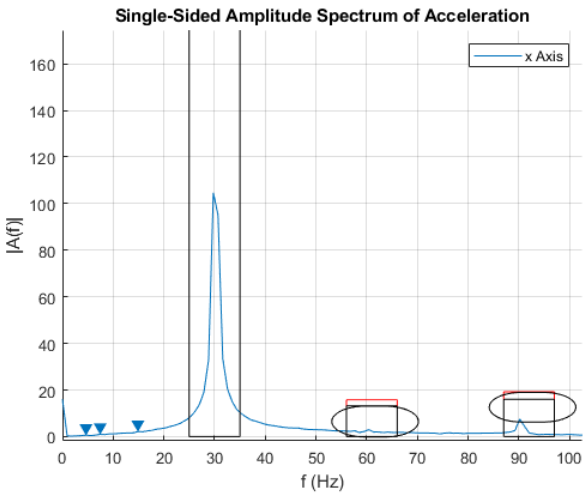

Figure 16: *Lower Eccentricity axis, 8V*

IX. ThingsBoard

To conclude this experiment, we will give the user a way to proper manage fault detection alarms got from the MCU on the server side of LoRa Network. As before, for simplicity, we will show the board screenshot just for the x axis, just to give an insight, without overloading the number of images. ThingsBoard is a simple instrument that will read data from the JSON packet that the MCU delivers over the LoRa Network. NodeRED will take the burden to parse the packet from the gateway and send important data to ThingsBoard, once the board is set-up on ThingsBoard platform, various Dashboard will be available to give a visual interpretation of all kinds of data. In this case, data will be shown real time on a Chart that will report over time changes of the interested source. The chart will have three levels, Off Alaram, Warning and Crtitical, representing the 3 possible state of the system over time, fig. 17.

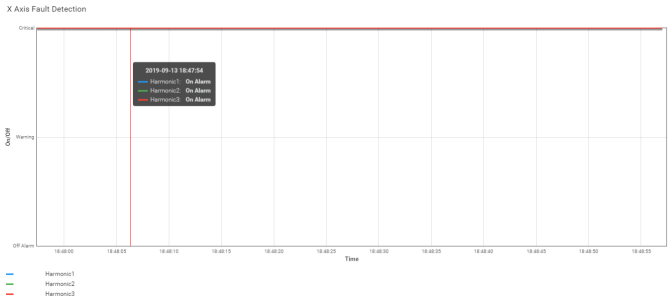

Figure 17: *Dashboard for Fault detection*

X. Conclusion and Future Development

From the results of our experiment we can presume that both the algorithms will work pretty well in most cases. Although maybe the second one is preferrable in case where the possibility of having lower peaks in the Fourier Transform can be a symptom of malfunctioning. To further expand this experiment, working with more datas would be necessary, to construct more reliable windows and maybe using a real world case as model to have a comparison between datas and reality. Another future development which could be useful, is to have some of the FFT bins sent to the server via LoRa Network, so the server can further elaborate the data and maybe use them to send replies to the MCU which can act as a controller for the system in use.

References

[1] Harish Chandra, A.S.Sekhar Fault detection in rotor bearing systems using time frequency techniques. MSSP, 2016.

[2] T. Ince ; S. Kiranyaz ; L. Eren ; M. Askar, M. Gabbouj Real-Time Motor Fault Detection by 1-D Convolutional Neural Networks IEEE Transactions on Industrial Electronics, 2016.

[3] M. Looney. *An Introduction to MEMS Vibration Monitoring* analog.com/en/analog-dialogue/articles/intro-to-mems-vibration-monitoring.html

[4] Mehmet Alper Ince. *Effects of Cutting Tool Parameters on Vibration* , ICMMR, 2016