Acoustic Based Predictive Maintenance

***Hrishikesh Gupta**, CPDM, Smart Manufacturing, ghrishikesh@iisc.ac.in*
***Priya**, CPDM, Smart Manufacturing, priya2024@iisc.ac.in*
***Yogendra Singh Rathore**, CPDM, Smart Manufacturing, yogendrasr@iisc.ac.in*
***Giorgia Bravi**, Exchange Student, OIR, giorgiabravi@iisc.ac.in*
***Pandarasamy Arjunan**, RBCCPS, samy@iisc.ac.in*

Overview

The main objective of this project is to develop an Edge AI-based anomaly detection and prediction system for a 3D printed air blower. The system is designed to monitor the operating conditions of the device, automatically identifying any anomalies or defects through the analysis of the sound produced during operation.

Background and Motivation

Predictive maintenance plays a crucial role in minimizing machine downtime and maintenance costs in smart manufacturing setups. Acoustic signals serve as a non-intrusive, low-cost, and efficient way to detect early signs of machine failure. The need for real-time, on-device (edge) fault detection drives the integration of Edge AI with acoustic analysis. Traditional cloud-based solutions often suffer from latency, bandwidth, and privacy concerns–Edge AI can help overcome these.

Methodology

1. **List of hardware required and their specifications:**

- Arduino Nano BLE Sense 33

- Themisto 12V 4000 RPM DC Motor – 2 Pieces

- 5 mm DC Jack – 1 male and 2 female

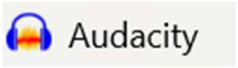
- AC to DC 12V adapter -1 piece

- 3D printer for 3D printed casing

- PLA filament
- Steel Screws, Iron blade

2. List of software used

- Audacity Application for Windows



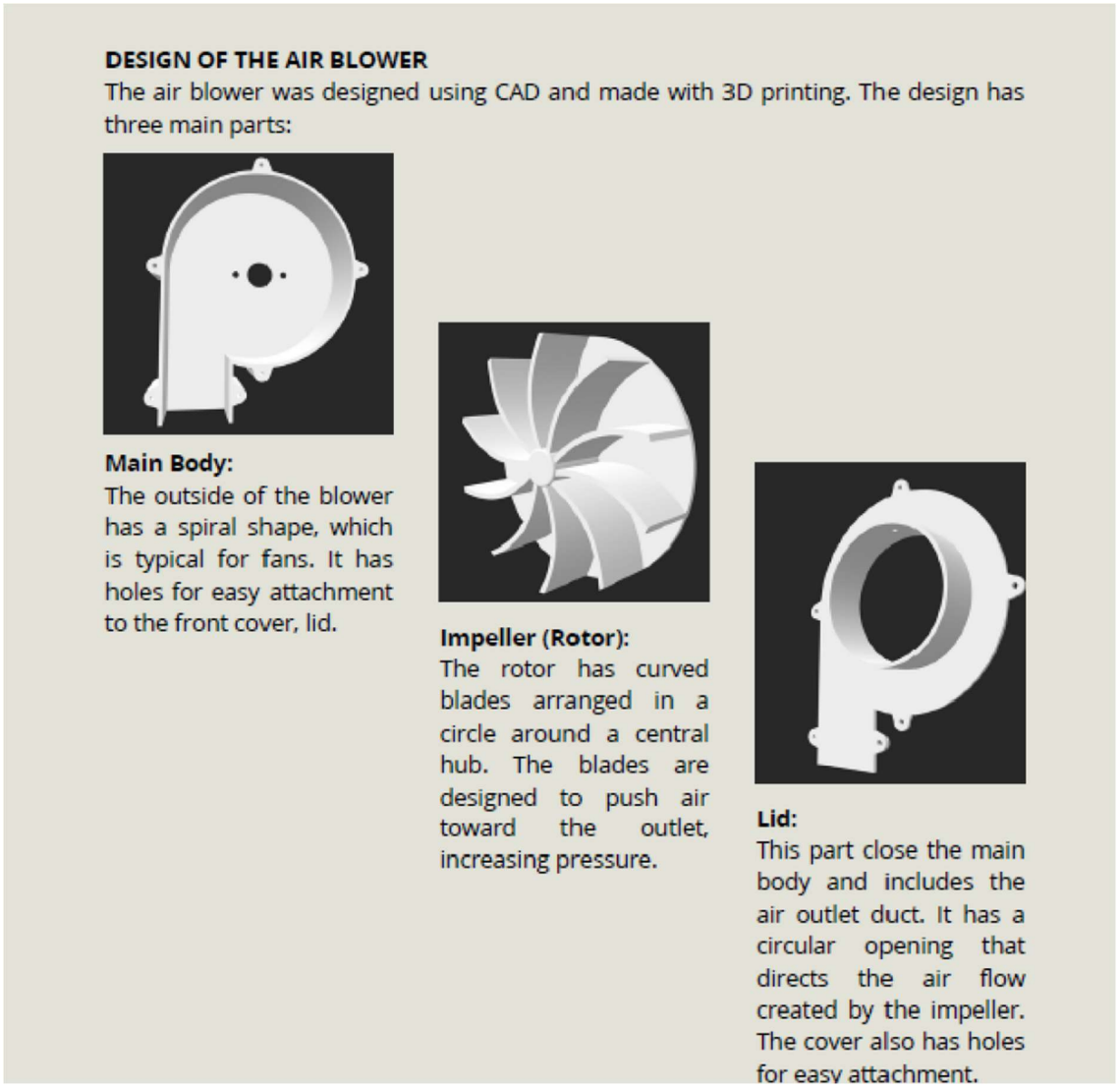
- Edge Impulse

- Arduino IDE

- Command Prompt

DESIGN OF THE AIR BLOWER

To proceed with the development and demonstration of the project, a small-scale air blower was designed using CAD. The model was 3D printed to realistically simulate operating conditions. To enable its functionality, a DC motor was attached to the blower, providing the necessary rotational motion to generate airflow and the characteristic operating sounds required for anomaly detection analysis.



Defects Introduced-

Two types of defects were introduced after the CAD Modelling :

- One is something getting stuck inside the blower
- Other is removing some of the screws from the blower

Fig. Defects Introduced

Data collection

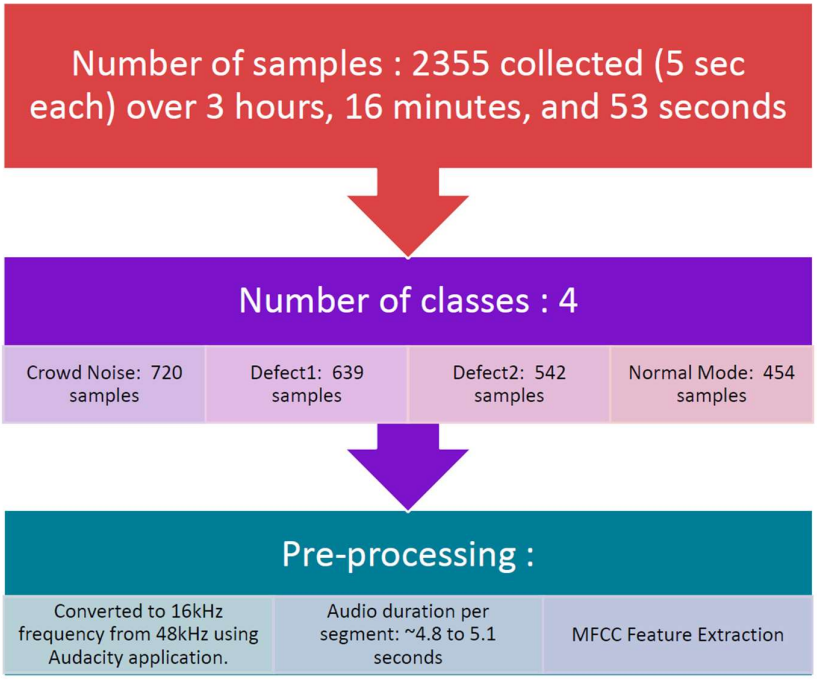


Fig. Data Collection

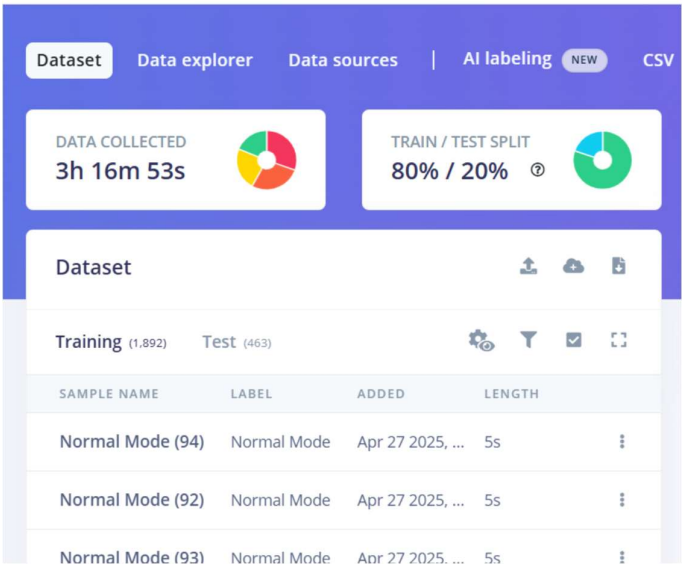


Fig. Data Collection (Edge Impulse)

Data Preprocessing using Audacity

- Converted to 16kHz frequency from 48kHz using Audacity application.
- Audio duration per segment: ~4.8 to 5.1 seconds

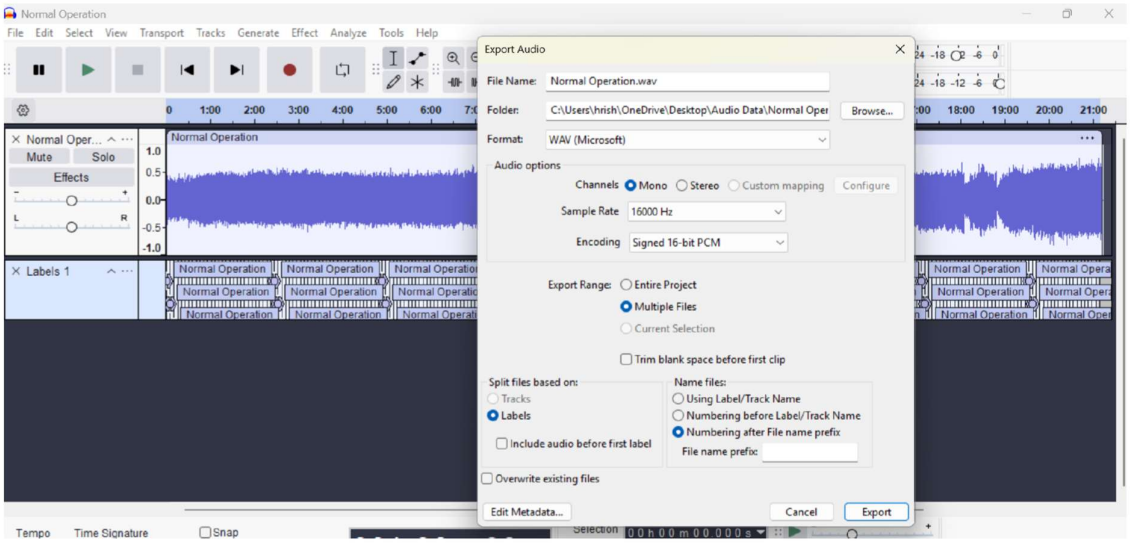
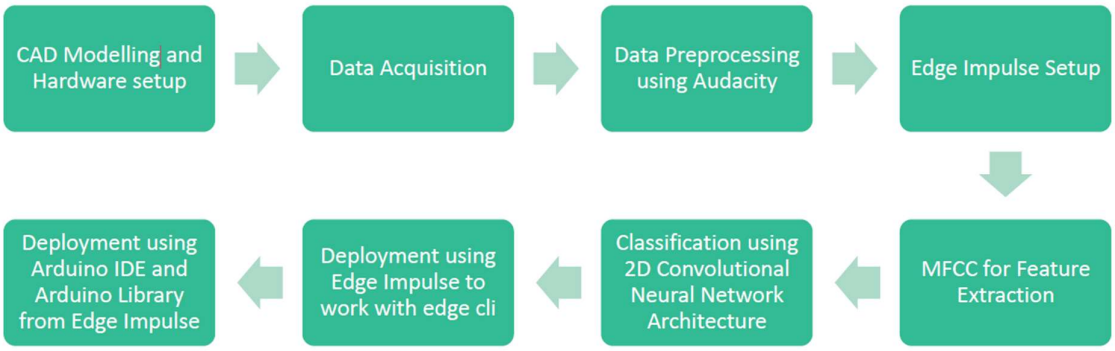


Fig. Data Preprocessing using Audacity

Methodology



Feature Extraction

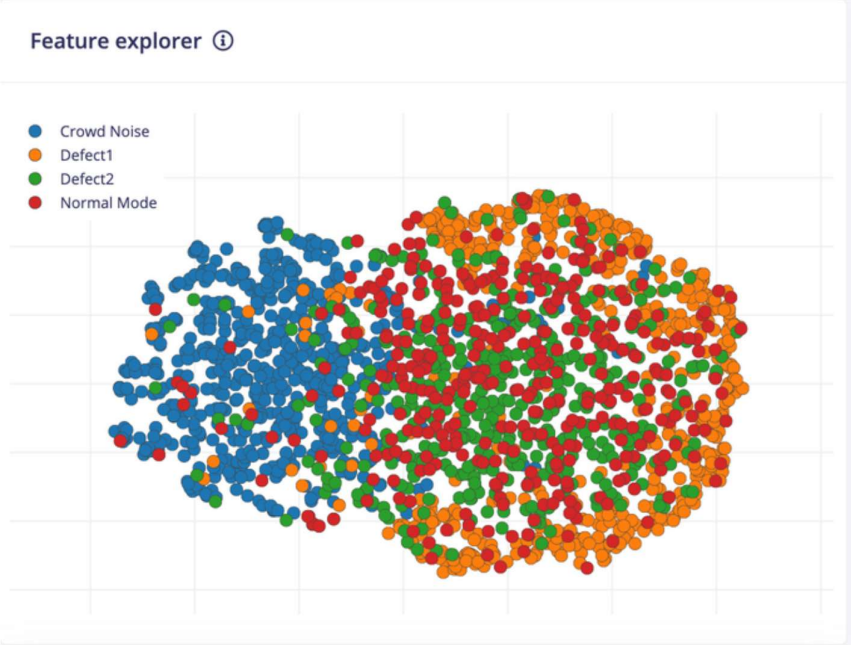
MFCC (Mel Frequency Cepstral Coefficients): Feature extraction technique used to represent the spectral content of an audio signal. It captures the most relevant sound characteristics. In our project, 25 MFCC coefficients are extracted from each audio segment to reduce the dimensionality and keep key information.

- Number of Coefficients: 25
- Frame Length: 0.02 s
- Frame Stride: 0.02 s
- Filter Number: 50
- FFT Length: 512
- Normalization Window Size: 101
- Low Frequency: 20 Hz
- High Frequency: 8000 Hz
- Pre-emphasis Coefficient: 0.98

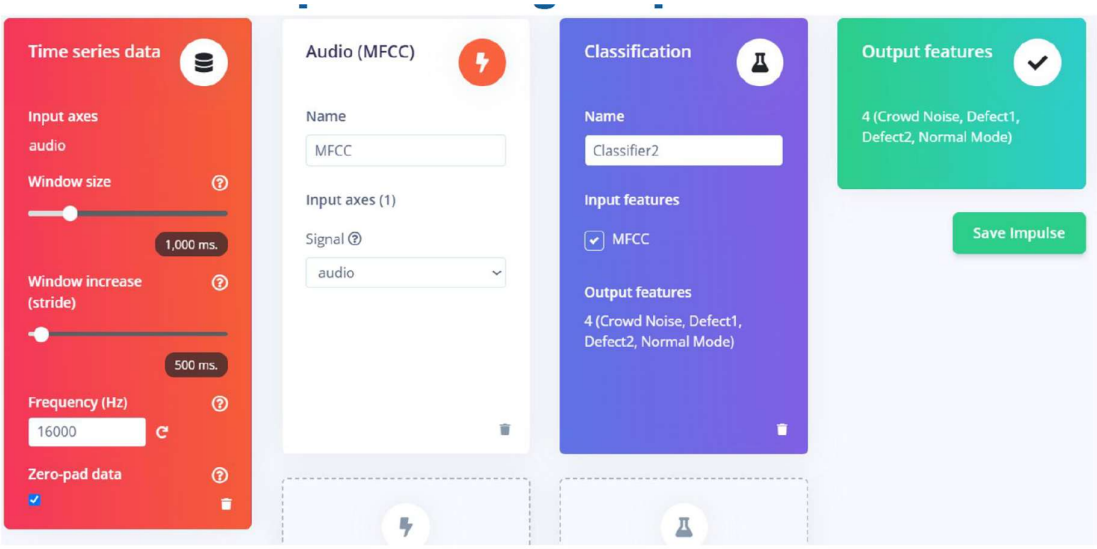
Windows and Segmentation: The audio data is segmented into temporal windows to analyze changes in the sound over time. Each window has a frame length of 0.02 seconds and a frame stride of 0.02 seconds. This segmentation allows detailed analysis of the audio signal, as each window is processed individually to extract features.

Training Set Information:

- **Data in Training Set:** 2h 38m 11s
- **Classes:** 4 (Crowd Noise, Defect1, Defect2, Normal Mode)
- **Training Windows:** 17,027



Model development:



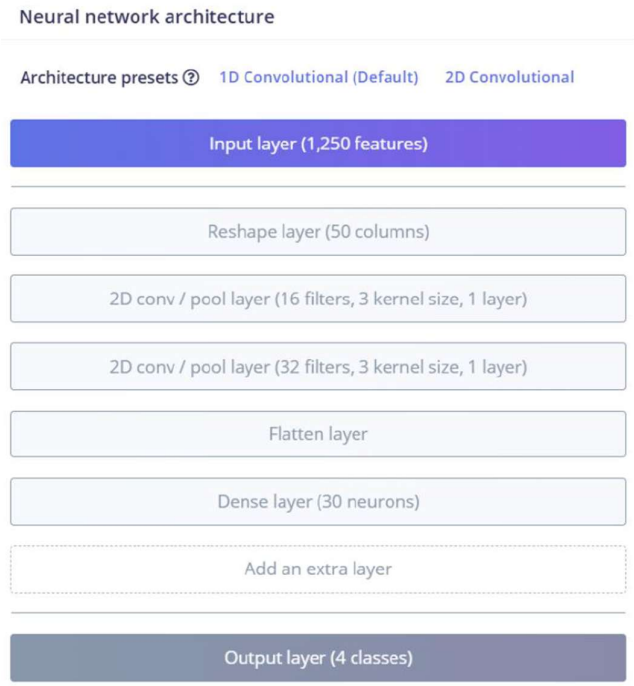
An impulse has been created in the edge impulse . It uses time series data with window size of 1000 ms and stride of 500 ms. MFCC has been used for feature extraction and then it has been classified into four categories : Crowd Noise, Defect1, Defect2, Normal Mode.

The neural network architecture is designed to process the 1,250 MFCC features extracted from audio data. The model consists of a reshape layer followed by two 2D convolutional layers with increasing filter sizes, a flattening layer, and a fully connected dense layer. The final output layer classifies the input into one of four classes: Crowd Noise, Defect1, Defect2, or Normal Mode. This architecture allows for extracting and analysing the relevant information from the audio features and classifying the blower's operating conditions in real-time.

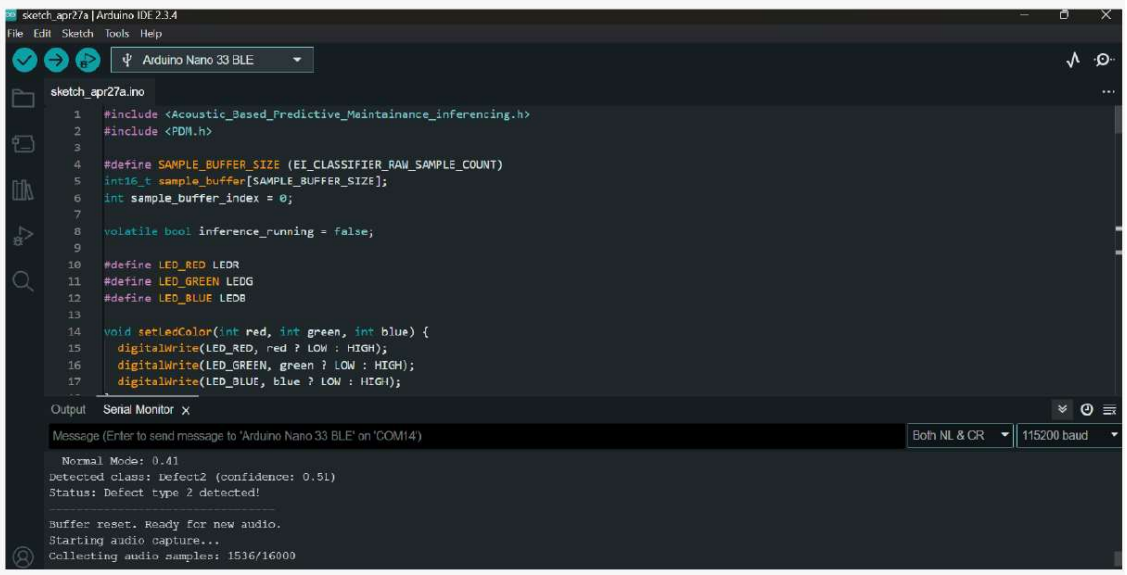
Number of epochs: 30

Learning Rate: 0.0005

Processor used: CPU



Further we have used Arduino IDE and Arduino Library from Edge Impulse to Deploy the model which can be executed just from a power supply.



Model compression:

The model underwent quantization (from float32 to int8), significantly reducing its size and improving performance while preserving accuracy.

Model characteristics:

- **Size Before and After Compression :**
  - Before (float32): ~383.3 KB
  - After (int8): ~121.1 KB→Compression ratio: ~3.16× smaller

Performance (Latency / RAM / Flash / Accuracy) :

- Latency improved by nearly 7.3× (from 2563 ms →350 ms).
- RAM usage reduced by more than 72%.
- Flash memory usage cut by ~68%.
- Accuracy slightly increased after quantization (from 99.64% to 99.71%).

Quantized (int8)

Selected ✔

	MFCC	CLASSIFIER2	TOTAL
LATENCY	258 ms.	92 ms.	**350 ms.**
RAM	15.4K	27.6K	**27.6K**
FLASH	-	121.1K	**-**
ACCURACY			**99.71%**

Unoptimized (float32)

Select

	MFCC	CLASSIFIER2	TOTAL
LATENCY	258 ms.	2,305 ms.	**2,563 ms.**
RAM	15.4K	100.3K	**100.3K**
FLASH	-	383.3K	**-**
ACCURACY			**99.64%**

Model deployment

Model was deployed in two ways:

1. Direct flashing on Arduino Nano BLE Sense 33 using Edge Impulse.
2. Using Arduino Library deployed using Edge Impulse and Arduino IDE to write a program so that the model doesn't require a terminal to show the output. Instead, it uses LED blinking to display the classes.

Result

Result on training data -

ACCURACY 99.4%

LOSS 0.05

Confusion matrix (validation set)

	CROWD NOISE	DEFECT1	DEFECT2	NORMAL MODE
CROWD NOISE	100%	0%	0%	0%
DEFECT1	0.3%	99.5%	0.2%	0%
DEFECT2	0.4%	0.5%	98.4%	0.8%
NORMAL MOD	0.2%	0%	0.3%	99.5%
F1 SCORE	1.00	1.00	0.99	0.99

Metrics (validation set)

METRIC	VALUE
Area under ROC Curve	1.00
Weighted average Precision	0.99
Weighted average Recall	0.99
Weighted average F1 score	0.99

Result on testing data-

ACCURACY 99.64%

Metrics for Classifier2

METRIC	VALUE
Area under ROC Curve	1.00
Weighted average Precision	1.00
Weighted average Recall	1.00
Weighted average F1 score	1.00

Confusion matrix

	CROWD NOI	DEFECT1	DEFECT2	NORMAL M	UNCERTAIN
CROWD NOISE	100%	0%	0%	0%	0%
DEFECT1	0.1%	99.9%	0%	0%	0%
DEFECT2	0.3%	0%	99.2%	0.3%	0.2%
NORMAL MOD	0.1%	0%	0%	99.3%	0.6%
F1 SCORE	1.00	1.00	1.00	0.99	

Prototype and Demo

- https://studio.edgeimpulse.com/studio/661135

- https://github.com/Hrishi2921/Acoustic_based_Predictive_Maintenance.git

- https://drive.google.com/file/d/1_4VNXr-3AqF0h8gI1namtyJOWFjnBIu6/view?usp=sharing

Challenges and Workarounds

Challenge 1: CAD modelling required a lot of calculations and hands – on. Later it got worked out after visualizing a lot of online prebuilt designs.

Challenge 2: The pre-defined neural network architecture was not giving good accuracy (only 75%). We tried using MFE which gave good accuracy during training and testing but later during prediction it also failed. Then we defined our own layers in the neural network architecture after feature extraction using MFCC. This in turn produced a good model with great results. We had to train the model 4 times to get to this accuracy (99.4 %)

Challenge 3: The code implementation for LED using Arduino IDE was a new learning. It taught us how to use Arduino library and tweak our own functions for the development board. It also enabled us to learn how to make the BLE Sense execute without any terminal and just with a power supply such as a Powerbank!!