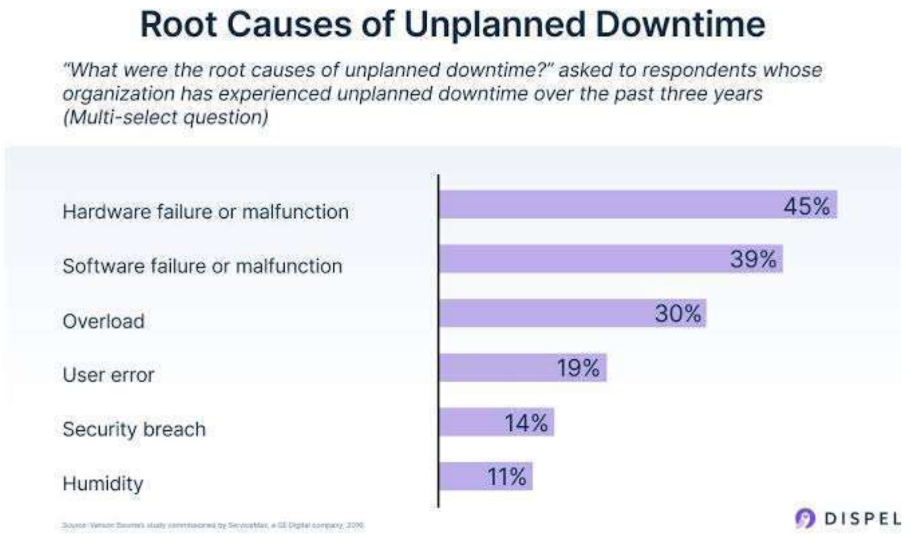
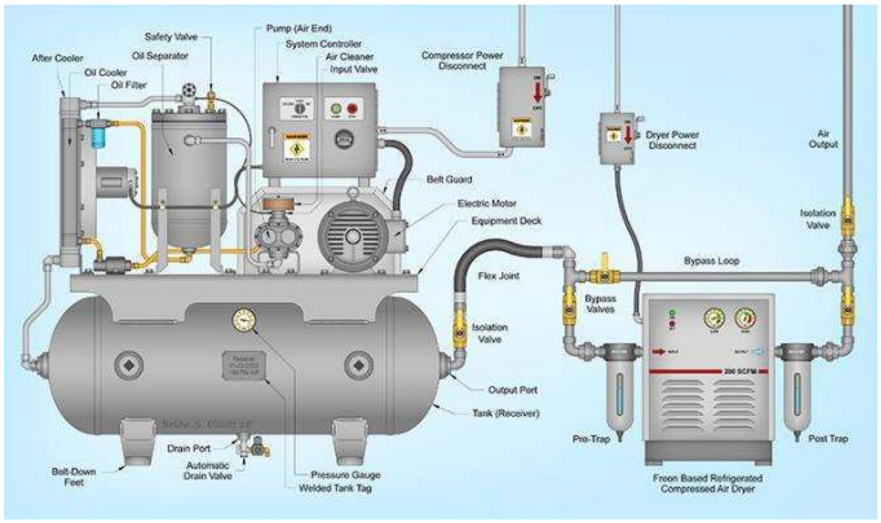
Problem Statement:

One of the big problems in industries is downtime, which can cause significant impacts in the sequential process of the Industry. When heavy equipment breaks down, safety becomes a major concern for workers. This kind of equipment failure not only leads to safety issues but also increases the cost of maintenance and causes delays in production. So, preventing downtime and avoiding equipment failures are crucial for keeping things running smoothly and ensuring the safety of the people working in the industry.





Objective:

- ✓ To develop a Predictive Maintenance System that predicts the occurrence of failure or any abnormal parameter condition in the Equipment.
- ✓ To Ensure the Safety of the workers in the Industry.
- ✓ To reduce the downtime, failure rate ,maintenance cost of the equipment.
- ✓ To Improve Equipment Lifespan .

Proposed Solution:

We developed a Predictive Maintenance System for Air Compressor for the problem statement.

A System which deals with industrial equipment such as Boilers ,Air Compressors,Industrial Bots, Motors, Ventilators etc,…where it continuously monitor's the specified parameters of that equipment to do predictive maintenance using Machine learning techniques.
By implementing predictive maintenance, companies can optimize maintenance schedules, minimize downtime, prevent future failures and reduce maintenance costs.

It will works by

Data from sensors -- Feature Extraction -- AI & ML algorithms -- Alerts and cloud storage

Alternate solutions

1. Predictive Maintenance:

Predictive maintenance is a proactive maintenance strategy that uses data and technology to predict when equipment maintenance is needed. It aims to perform maintenance tasks just in time, right before equipment failure, minimizing downtime and unnecessary maintenance.

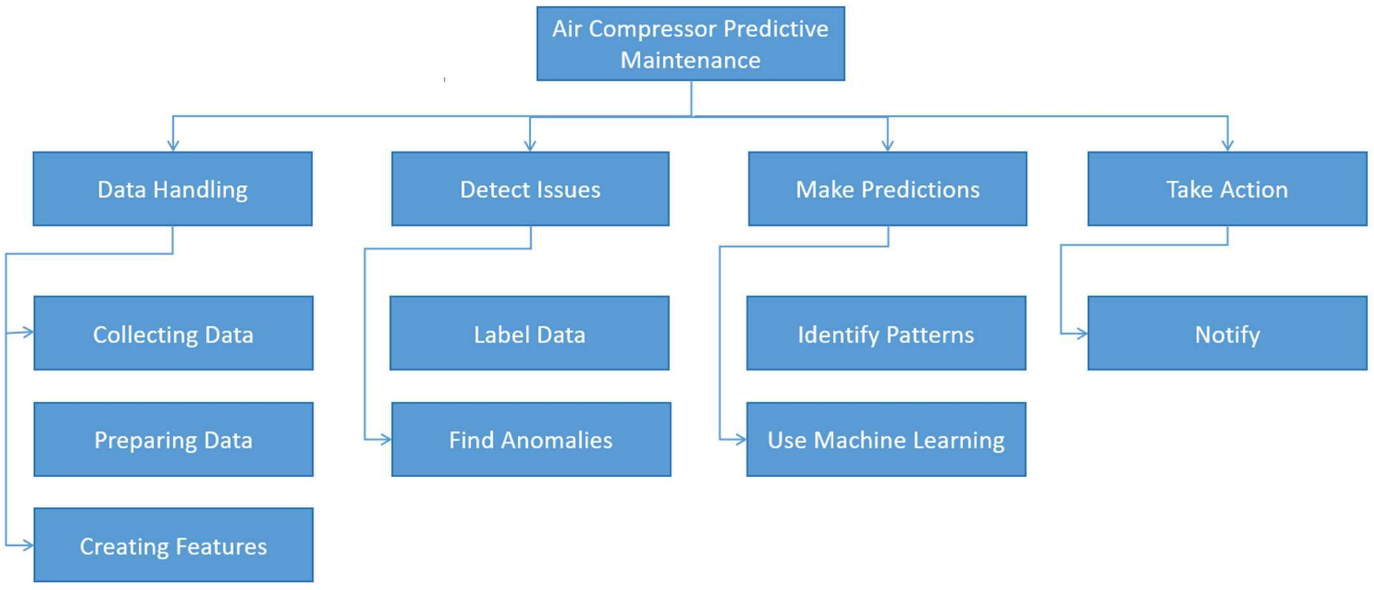
2. Preventive Maintenance:

Preventive maintenance is a proactive maintenance strategy based on scheduled or calendar-based maintenance tasks. It involves performing routine inspections and maintenance activities at predetermined intervals, regardless of the equipment's condition.

3. Reactive Maintenance:

Reactive maintenance, also known as corrective or run-to-failure maintenance, involves repairing or replacing equipment after it has failed. It is a purely reactive approach to maintenance.

Functional Decomposition:



About the Air Compressor :

The compressor that we are chosen to do predictive maintenance is **Two Stage Reciprocating Air Compressor**.

A two-stage reciprocating air compressor is a type of air compressor that consists of two distinct compression stages within a single unit. Each stage compresses the air before it is delivered for various industrial applications.

Key features and functions:

Piston Design:
The compressor uses one or more pistons that move back and forth within cylinders. The pistons are typically powered by an electric motor or an internal combustion engine.

First Stage:
In the first stage, ambient air is drawn into the cylinder and compressed to an intermediate pressure. This compressed air then moves to an intercooler, where it is cooled before entering the second stage.

Intercooler:
The intercooler reduces the temperature of the compressed air, making it more manageable and preparing it for the second stage of compression.

Second Stage: The cooled air from the intercooler is then drawn into the second cylinder for further compression. Here, the air is compressed to the final desired pressure.

Intercooling:
Purpose:

Intercooling between the two stages is essential to reduce the temperature of the air. This prevents excessive heat buildup during compression, which can be detrimental to the compressor's efficiency and performance.

Efficiency:

Cooling the air between stages improves the efficiency of the compression process, allowing for higher pressure ratios without excessive heat.
The two-stage design enables the compressor to achieve higher pressures compared to a single-stage compressor. This is particularly beneficial for applications that require elevated air pressure levels.

Industrial Use:

Two-stage reciprocating air compressors are commonly employed in various industrial applications, including manufacturing, automotive, construction, and other settings where high-pressure compressed air is needed.

Maintenance:

While these compressors provide advantages in terms of pressure output, they may require more maintenance compared to single-stage compressors due to the complexity of the two-stage design.

Specifications:

L.P.Cylinder	Dia: 78 mm	Length: 72 mm
H.P.Cylinder	Dia: 40 mm	Length: 72 mm

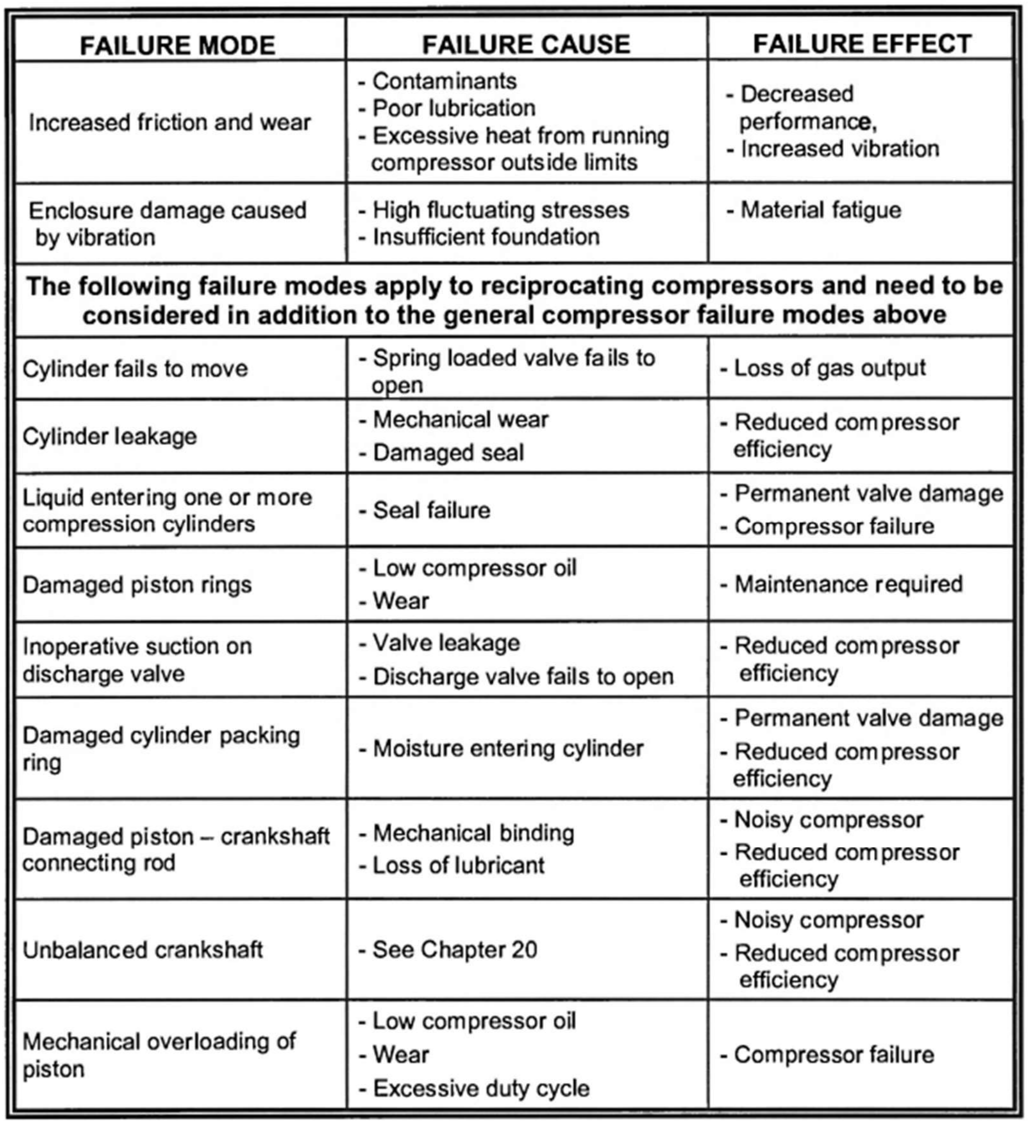
Speed: 950 rpm

Coefficient of Discharge (Cd): 0.6

Parameters:

Out of the parameters listed below , we chosen Temperature, Pressure and Vibration as the inputs to train the model and make the predictions to find the failure rate of the air compressor. As these parameters plays a vital role in the below failure modes of the compressor.

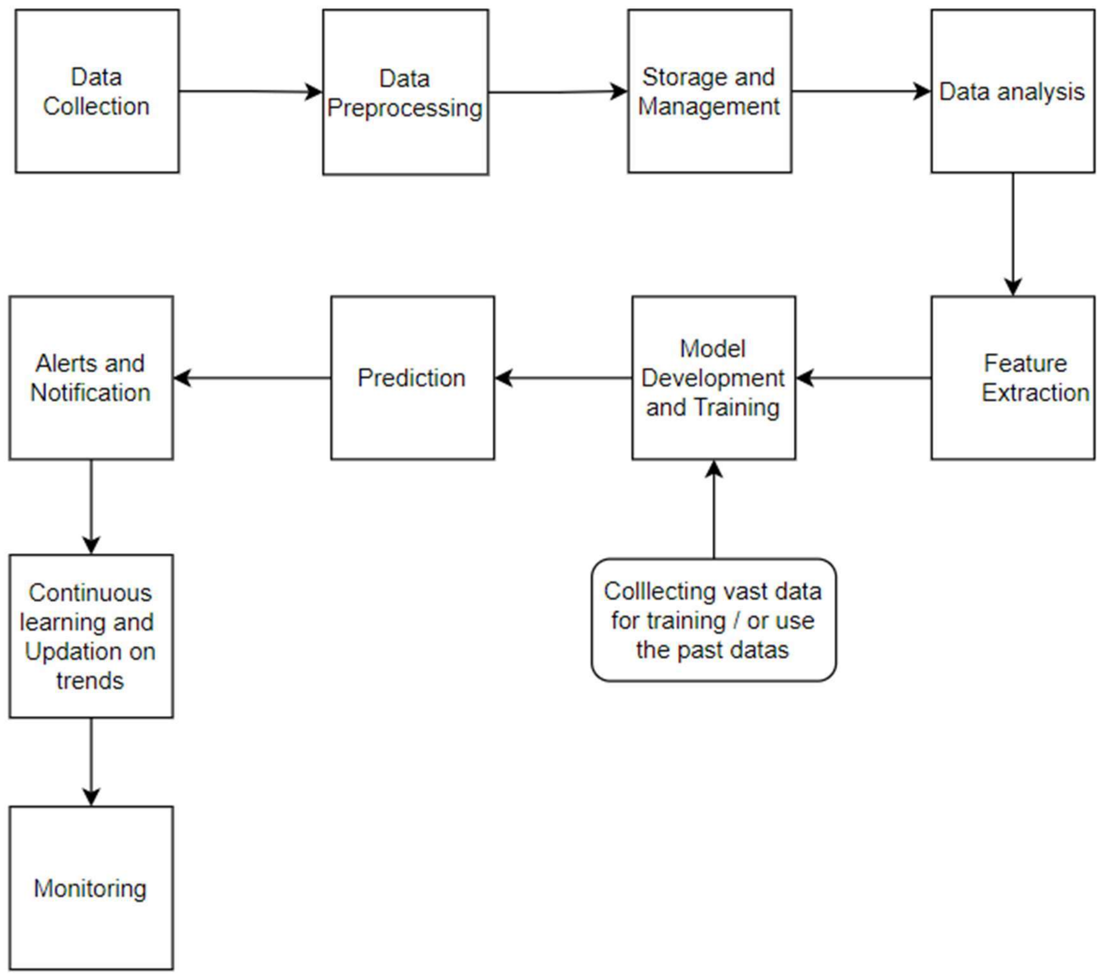
Table 13-1. Compressor Failure Modes (Continued)



FAILURE MODE	FAILURE CAUSE	FAILURE EFFECT
Increased friction and wear	- Contaminants - Poor lubrication - Excessive heat from running compressor outside limits	- Decreased performance, - Increased vibration
Enclosure damage caused by vibration	- High fluctuating stresses - Insufficient foundation	- Material fatigue
The following failure modes apply to reciprocating compressors and need to be considered in addition to the general compressor failure modes above		
Cylinder fails to move	- Spring loaded valve fails to open	- Loss of gas output
Cylinder leakage	- Mechanical wear - Damaged seal	- Reduced compressor efficiency
Liquid entering one or more compression cylinders	- Seal failure	- Permanent valve damage - Compressor failure
Damaged piston rings	- Low compressor oil - Wear	- Maintenance required
Inoperative suction on discharge valve	- Valve leakage - Discharge valve fails to open	- Reduced compressor efficiency
Damaged cylinder packing ring	- Moisture entering cylinder	- Permanent valve damage - Reduced compressor efficiency
Damaged piston – crankshaft connecting rod	- Mechanical binding - Loss of lubricant	- Noisy compressor - Reduced compressor efficiency
Unbalanced crankshaft	- See Chapter 20	- Noisy compressor - Reduced compressor efficiency
Mechanical overloading of piston	- Low compressor oil - Wear - Excessive duty cycle	- Compressor failure

Methodology:

The maintenance system invloves data collection, analysis ,training and prediction with its auxillary processes like data tuning, noise reduction.
Sensors deals with the data collection process and the machine learning techniques to train and predict the output with any cloud system to store the data.


Data Collection
Data Preprocessing
Storage and Management
Data analysis
Feature Extraction
Model Development and Training
Prediction
Alerts and Notification
Continuous learning and Updation on trends
Monitoring
Colllecting vast data for training / or use the past datas

The data collected from sensors sent to the circuit which do the preprocessing task ,then the data's get stored in cloud for further process ,then they are passed to the trained algorithm to predict the output.

Our system continuously monitors the parameters using these sensors ,if any value goes up or down to the given or trained value ,it alerts the personby sending signal to our given interface or to the customized version of the Alarm.

For the further advantage we store the data in the cloud for the future accessAnd too analyzing purposes.

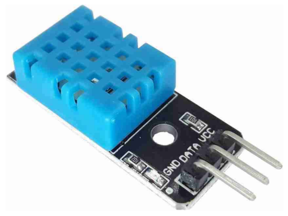
Components Required:

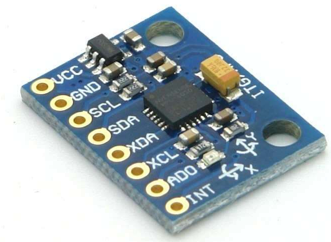
Sensors:

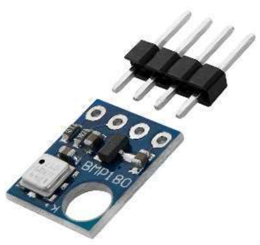
- ✓ DHT11 (Temperature Sensor).
- ✓ BMP180 Pressure Sensor
- ✓ MPU6050 Accelerometer

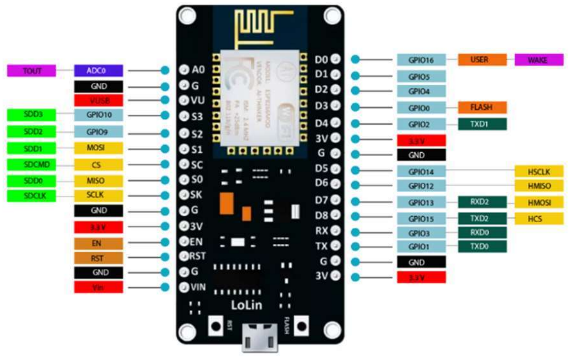
Controller for prototype

ESP8266

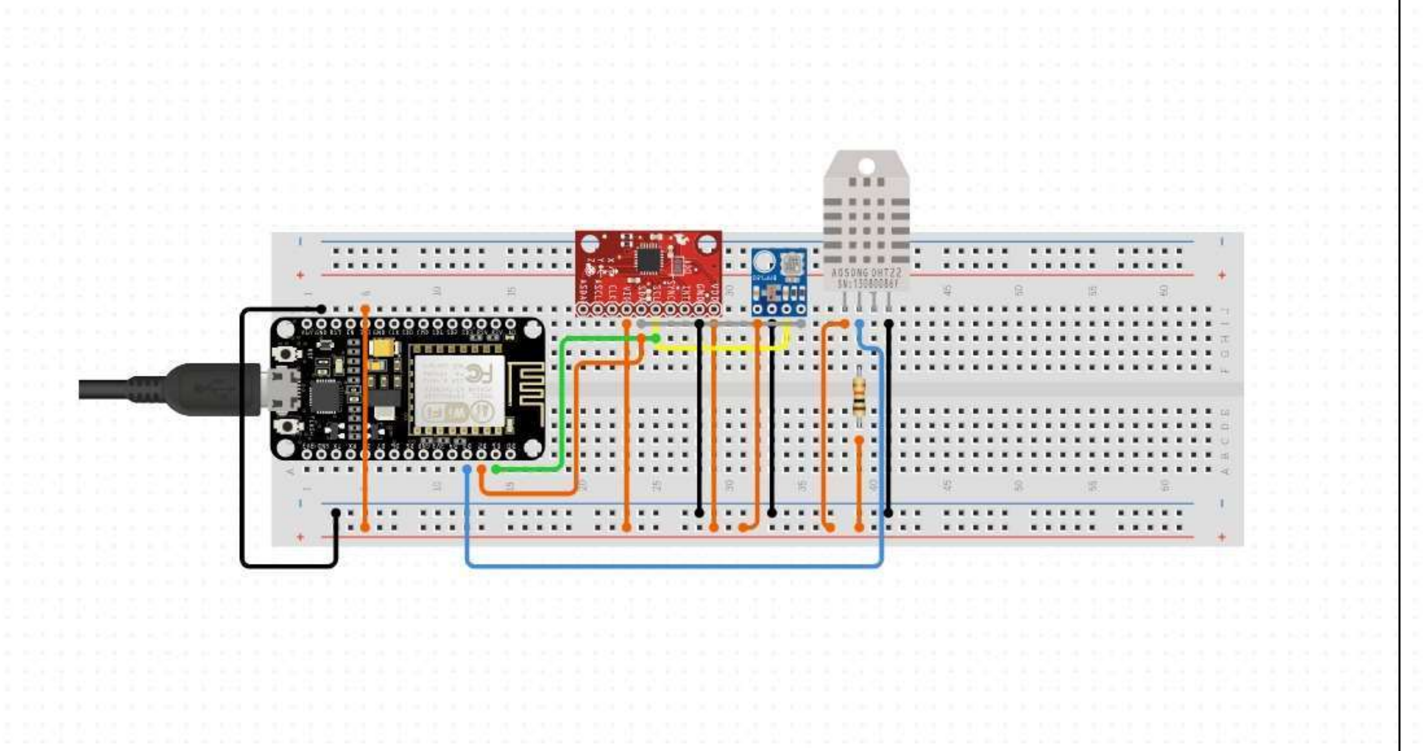




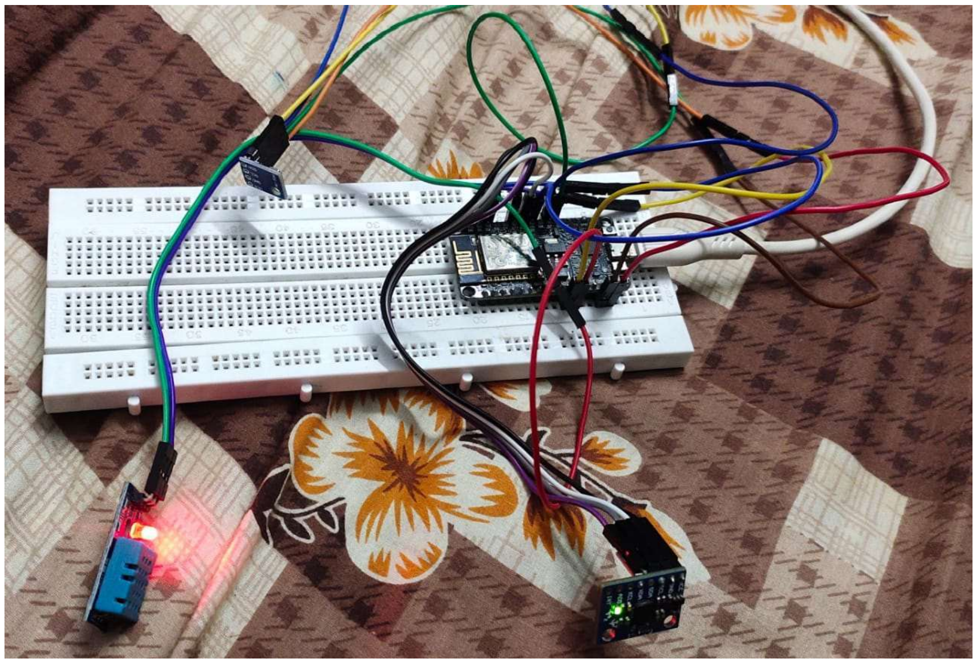




Circuit Diagram:

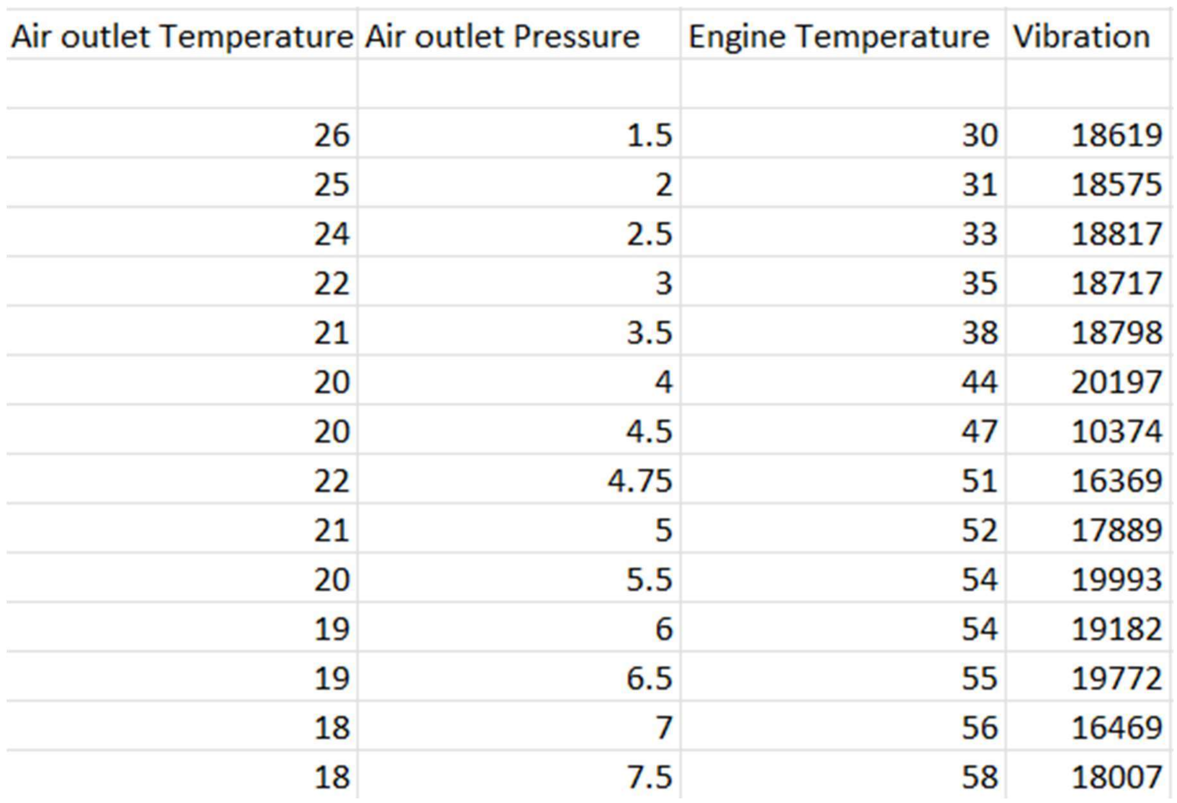


Hardware Setup:



Dataset:

We collected the real time dataset from the two stage air compressor. where it comprises of Outlet Air Pressure , Temperature, Engine Temperature and Vibration of Air Compressor.



Air outlet Temperature	Air outlet Pressure	Engine Temperature	Vibration
26	1.5	30	18619
25	2	31	18575
24	2.5	33	18817
22	3	35	18717
21	3.5	38	18798
20	4	44	20197
20	4.5	47	10374
22	4.75	51	16369
21	5	52	17889
20	5.5	54	19993
19	6	54	19182
19	6.5	55	19772
18	7	56	16469
18	7.5	58	18007

Tools and Technology:

Tools:

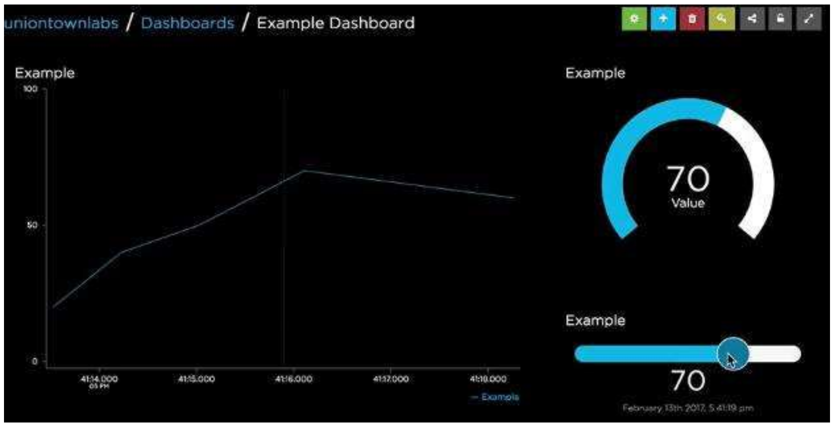
- Sensors
- Data Acquisition System
- Microcontroller
- Data Visualisation System

Technology:

- ML
- Cloud computing

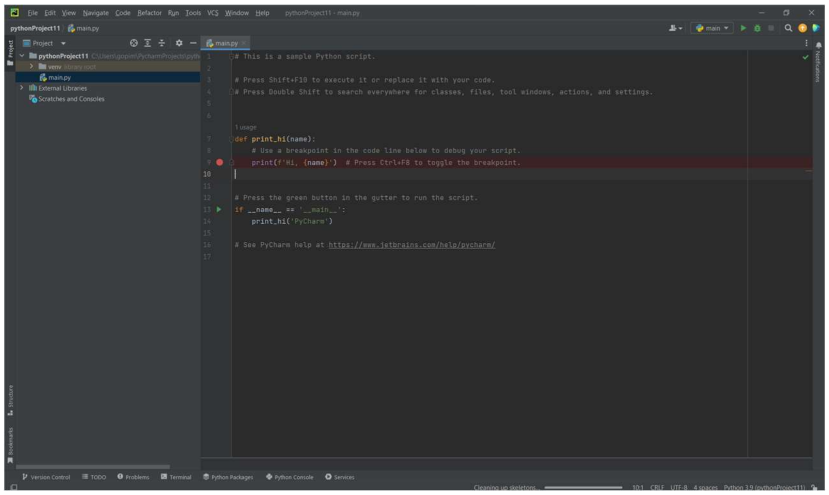
Data Visualisation System(Cloud):

We used Adafruit cloud in order to store and represent the data .It is Adafruit IO is a cloud service for IoT projects, allowing users to store, retrieve, and visualize data from their devices. It features data feeds, customizable dashboards, and libraries for popular IoT platforms. Users can connect devices, send data to feeds, and create interactive dashboards for monitoring. Advanced features include triggers for automation. Adafruit IO Plus offers additional benefits, and the platform emphasizes security and privacy.


uniontownlabs / Dashboards / Example Dashboard
Example
Example
70
Value
Example
70
February 13th 2017, 5:41:19 pm
— Example

Data Acquisition System and ML training Platform:

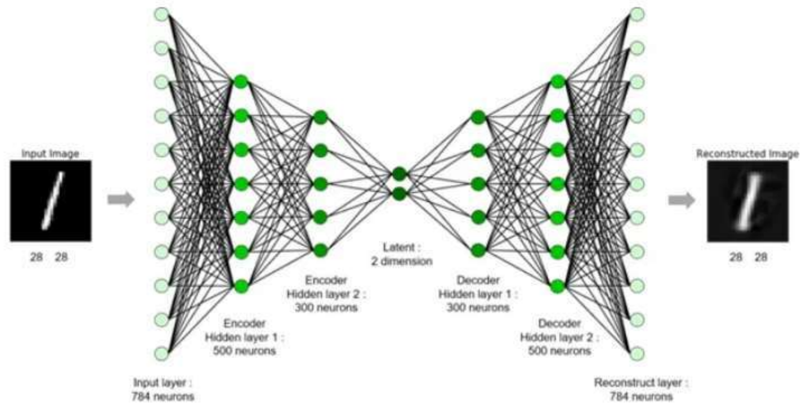
Pycharm IDE is the platform that we used to collect data , classify , make predictions and then write it into the IOT Arduino cloud. Where PyCharm is a popular integrated development environment (IDE) for Python development. It is developed by JetBrains and is available in two editions: PyCharm Community Edition (free) and PyCharm Professional Edition (commercial, with additional features).

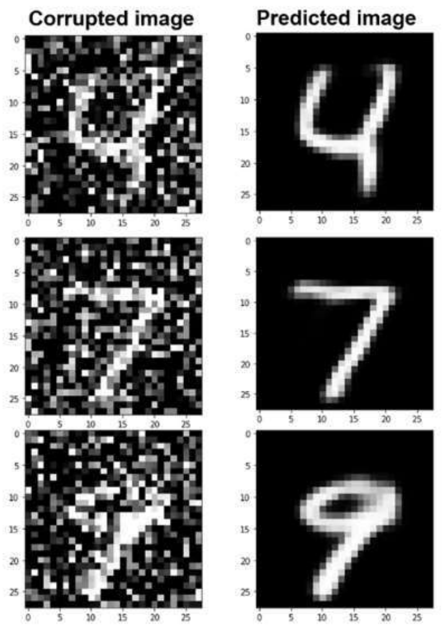


```
# This is a sample Python script.

# Press Shift+F10 to execute it or replace it with your code.
# Press Double Shift to search everywhere for classes, files, tool windows, actions, and settings.


def print_hi(name):
    # Use a breakpoint in the code line below to debug your script.
    print(f'Hi, {name}')  # Press Ctrl+F8 to toggle the breakpoint.


# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    print_hi('PyCharm')

# See PyCharm help at https://www.jetbrains.com/help/pycharm/
```

Algorithm :

We used **Auto Encoders** here to classify and predict the data. Autoencoders are a type of artificial neural network used for unsupervised learning and dimensionality reduction. They consist of an encoder network that compresses input data into a lower-dimensional representation (encoding) and a decoder network that reconstructs the original input from this encoding. Autoencoders are widely employed for tasks such as feature learning, data denoising, and anomaly detection. They are characterized by their ability to learn efficient representations of data, capturing essential features and reducing redundancy.


Input Image
28 28
Input layer : 784 neurons
Encoder Hidden layer 1 : 500 neurons
Encoder Hidden layer 2 : 300 neurons
Latent : 2 dimension
Decoder Hidden layer 1 : 300 neurons
Decoder Hidden layer 2 : 500 neurons
Reconstruct layer : 784 neurons
Reconstructed Image
28 28



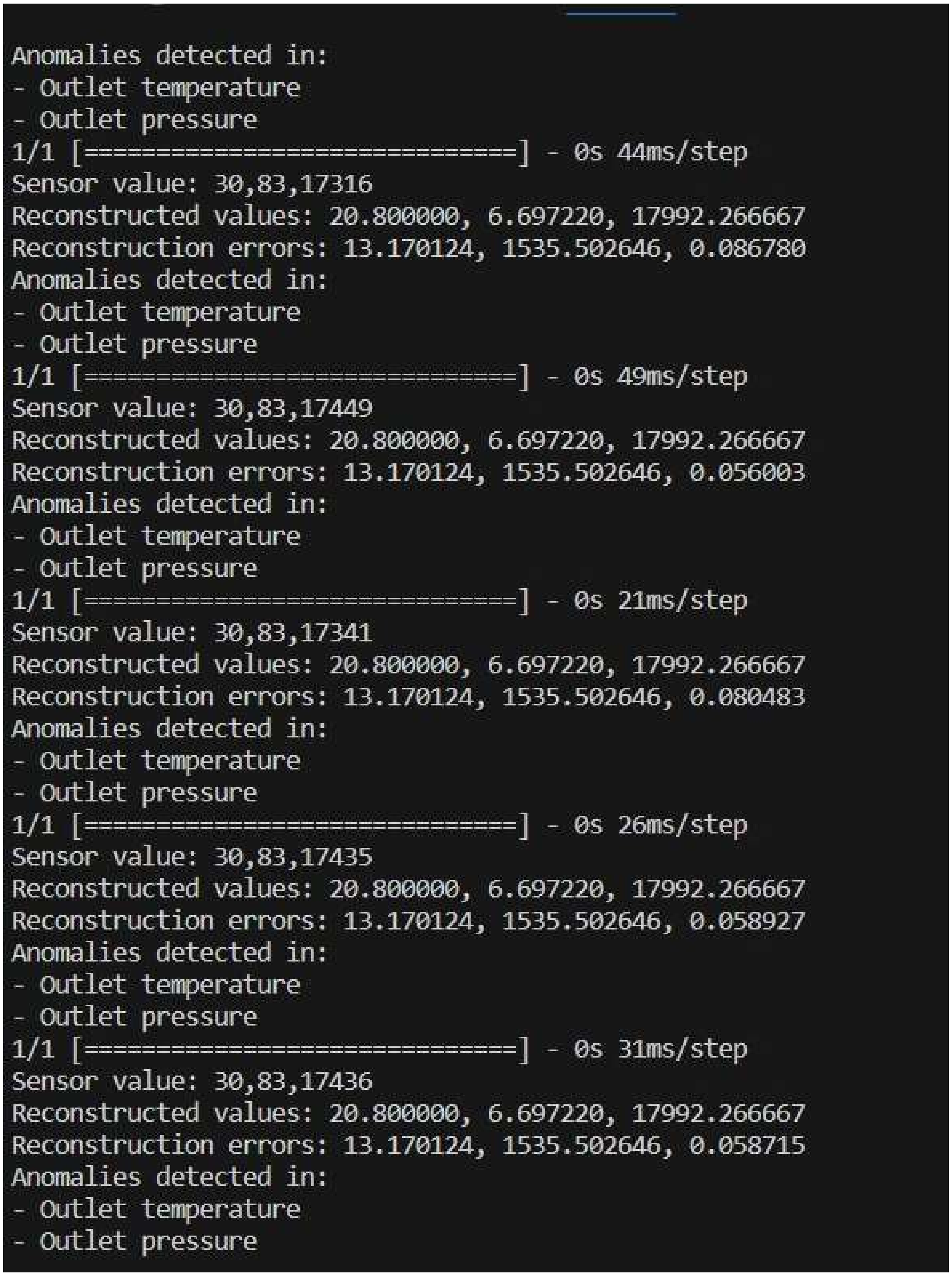
It is a type of artificial neural network used to learn efficient codings of unlabeled data (unsupervised learning). An autoencoder learns two functions: an encoding function that transforms the input data, and a decoding function that recreates the input data from the encoded representation.

Python code for Algorithm:

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models
import serial
import time

def load_data(file_path, columns):
    data = pd.read_csv(file_path)
    sensor_data = data[columns].values
    return sensor_data

def normalize_data(data):
    data_mean = np.mean(data, axis=0)
    data_std = np.std(data, axis=0)
    normalized_data = (data - data_mean) / data_std
    return normalized_data, data_mean, data_std

def build_autoencoder(input_dim):
    encoder = models.Sequential([
        layers.Dense(64, activation='relu', input_shape=(input_dim,)),
        layers.Dense(32, activation='relu'),
        layers.Dense(16, activation='relu'),
        layers.Dense(8, activation='relu')
    ])

    decoder = models.Sequential([
        layers.Dense(16, activation='relu', input_shape=(8,)),
        layers.Dense(32, activation='relu'),
        layers.Dense(64, activation='relu'),
        layers.Dense(input_dim, activation='sigmoid')
    ])

    autoencoder = models.Sequential([encoder, decoder])
    autoencoder.compile(optimizer='adam', loss='mse')
    return autoencoder

def train_autoencoder(model, data, epochs, batch_size):
    model.fit(data, data, epochs=epochs, batch_size=batch_size)

def calculate_threshold(model, data):
    reconstructed_data = model.predict(data)
    mse = np.mean(np.square(data - reconstructed_data), axis=1)
    mean_error = np.mean(mse)
    std_error = np.std(mse)
    threshold = 2*(mean_error + (2 * std_error))  # Adjust the factor (e.g., 2) based on your requirements
    return threshold

def reconstruction(test_instance):
    normalized_test_instance = (test_instance - data_mean) / data_std
    reconstructed_test_instance = autoencoder.predict(normalized_test_instance.reshape(1, -1))
    denormalized_reconstructed_instance = (reconstructed_test_instance * data_std) + data_mean
    reconstruction_errors = np.mean(np.square(normalized_test_instance - reconstructed_test_instance), axis=0)
    reconstructed_values = denormalized_reconstructed_instance.flatten()
    formatted_values = ', '.join(['{:.6f}'.format(val) for val in reconstructed_values])
    formatted_errors = ', '.join(['{:.6f}'.format(err) for err in reconstruction_errors])
    return formatted_values, formatted_errors

file_path = r"C:\Users\aakas\OneDrive\Desktop\Copy of CDP_Dataset(1).csv"
selected_columns = ['outlet_temperature','outlet_pressure','vibration']
epochs = 1000
batch_size = 32
sensor_data = load_data(file_path, selected_columns)
normalized_data, data_mean, data_std = normalize_data(sensor_data)
input_dim = normalized_data.shape[1]
autoencoder = build_autoencoder(input_dim)
train_autoencoder(autoencoder, normalized_data, epochs, batch_size)

threshold = calculate_threshold(autoencoder, normalized_data)
print("Threshold:", threshold)

while True:

    ser = serial.Serial('COM5', 115200)  # Replace 'COMX' with your port
    time.sleep(2)  # Allow time for serial connection to initialize
    sensor_data = str(ser.readline().strip(), 'utf-8', errors='ignore')
    sensor_values = list(map(float, sensor_data.split(',')))
    sensor_array = np.array(sensor_values)
    test_instance = sensor_array
    reconstructed_values, reconstruction_errors = reconstruction(test_instance)
    print("Sensor value:",sensor_data)
    print("Reconstructed values:", reconstructed_values)
    print("Reconstruction errors:", reconstruction_errors)
    reconstruction_errors = np.array([float(err) for err in reconstruction_errors.split(', ')])
    anomalies = reconstruction_errors > threshold
    anomalous_features = np.where(anomalies)[0]
    if len(anomalous_features) > 0:
        print("Anomalies detected in:")
        for idx in anomalous_features:
            if idx == 0:
                print("- Outlet temperature")
            elif idx == 1:
                print("- Outlet pressure")
            else:
                print("- Vibration")
    else:
        print("No anomalies detected")

    ser.close()
```

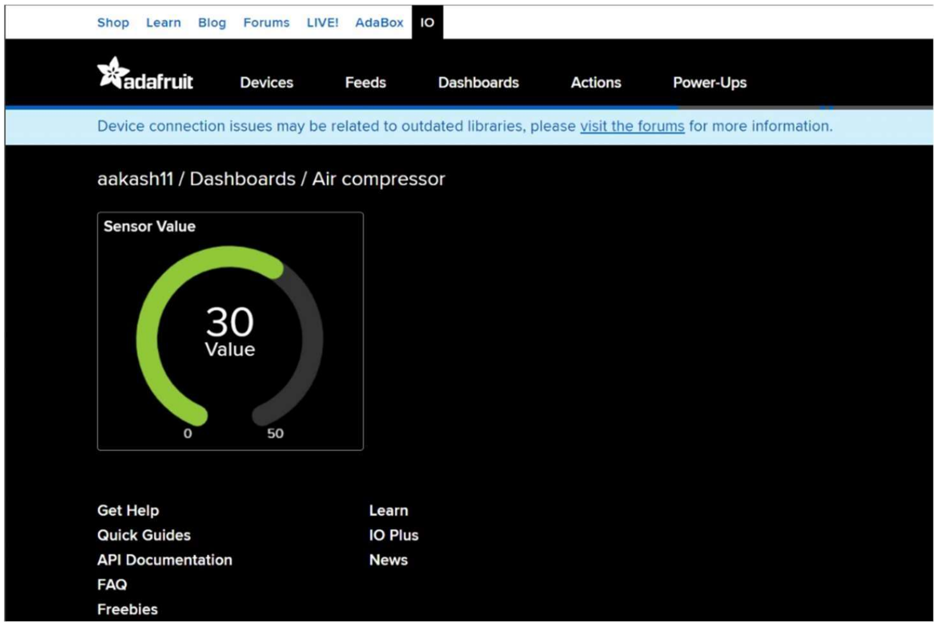
Output:

Python Terminal:



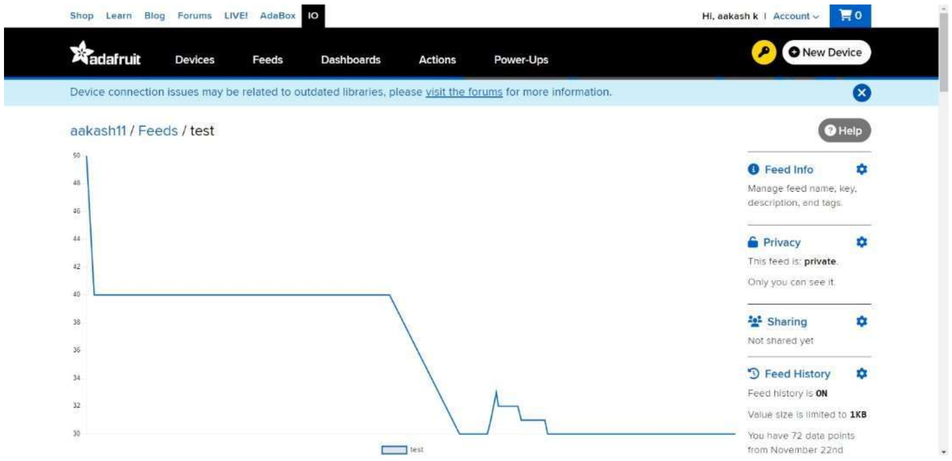
```
Anomalies detected in:
- Outlet temperature
- Outlet pressure
1/1 [==============================] - 0s 44ms/step
Sensor value: 30,83,17316
Reconstructed values: 20.800000, 6.697220, 17992.266667
Reconstruction errors: 13.170124, 1535.502646, 0.086780
Anomalies detected in:
- Outlet temperature
- Outlet pressure
1/1 [==============================] - 0s 49ms/step
Sensor value: 30,83,17449
Reconstructed values: 20.800000, 6.697220, 17992.266667
Reconstruction errors: 13.170124, 1535.502646, 0.056003
Anomalies detected in:
- Outlet temperature
- Outlet pressure
1/1 [==============================] - 0s 21ms/step
Sensor value: 30,83,17341
Reconstructed values: 20.800000, 6.697220, 17992.266667
Reconstruction errors: 13.170124, 1535.502646, 0.080483
Anomalies detected in:
- Outlet temperature
- Outlet pressure
1/1 [==============================] - 0s 26ms/step
Sensor value: 30,83,17435
Reconstructed values: 20.800000, 6.697220, 17992.266667
Reconstruction errors: 13.170124, 1535.502646, 0.058927
Anomalies detected in:
- Outlet temperature
- Outlet pressure
1/1 [==============================] - 0s 31ms/step
Sensor value: 30,83,17436
Reconstructed values: 20.800000, 6.697220, 17992.266667
Reconstruction errors: 13.170124, 1535.502646, 0.058715
Anomalies detected in:
- Outlet temperature
- Outlet pressure
```

Cloud:



Device connection issues may be related to outdated libraries, please visit the forums for more information.

aakash11 / Dashboards / Air compressor



Device connection issues may be related to outdated libraries, please visit the forums for more information.

aakash11 / Feeds / test

Feed Info

Manage feed name, key, description, and tags.

Privacy

This feed is: **private**.

Only you can see it.

Sharing

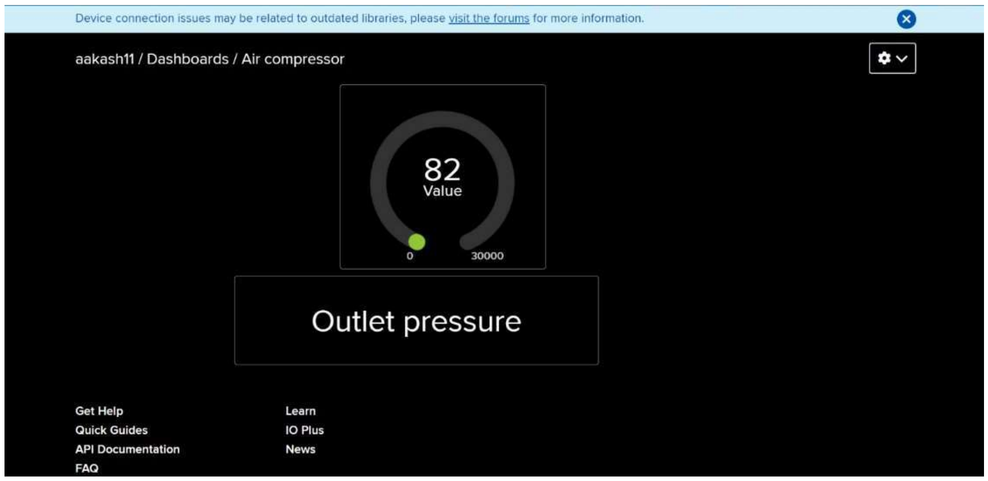
Not shared yet

Feed History

Feed history is **ON**

Value size is limited to **1KB**

You have 72 data points from November 22nd


Device connection issues may be related to outdated libraries, please visit the forums for more information.

aakash11 / Dashboards / Air compressor