

IoT and Machine Learning based Air Quality Monitoring System

K. Sravani¹, Ch. Ujwal Kumar², Ch. Sri Manjunadh³, P. Rama Krishna⁴,
N. Maneesha⁵

^{1,2,3,4,5}Internet of Things, Seshadri Rao Gudlavalleru Engineering College, (JNTUK), Gudlavalleru, India

ABSTRACT

Indoor Air Quality (IAQ) is a very important factor that can affect the health of human being especially in the urban and residential environments where people spend most of their time. The consequences of higher concentrations of particulate matter (PM1, PM2.5 and PM10) include respiratory and cardiovascular diseases as well as cognitive impairment. Recent developments in machine learning (ML) and Internet of Things (IoT) have made it possible to improve the IAQ monitoring devices; however, existing approaches have some limitations such as limited pollutants coverage, computational complexity, and non-real-time adaptability. This project is offering a complete, easily expandable and affordable IAQ monitoring system with multi-parameter sensing (PM1, PM2.5, PM10), real-time IoT dashboards and sophisticated data analysis tools. The device is built on low-cost edge computing enabled by ESP8266 chips that gather data and push it to the cloud for near real-time processing. The screening and classification of anomalies are done by a hybrid of rule-based classification and anomaly detection algorithms such as Isolation Forest and Random Forest to overcome the challenges of the current systems. The solution is meant to be scalable, versatile and economical to enable its application in homes, offices and learning institutions. Our system enhances real time monitoring and decision making in the control of health risks through the integration of edge and cloud computing.

Keyword-Anomaly Detection, Internet of Things, Indoor Air Quality (IAQ), Isolation Forest, Particulate Matter, Random Forest.

INTRODUCTION

Various empirical researches on the adverse effect of exposure to air pollution on human health and the environment have been conducted in the last few decades. In agriculture, air pollution influences crop yields and creates undesirable social and economic effects, especially in developing countries. Therefore, air pollution is a key component of the United Nations Sustainable Development Goals (SDGs) where there is a desire to minimize significantly the effect of harmful substances on human health. Air pollution monitoring is necessary to raise awareness among the general public on sustainable urban ecosystems and human health. The disadvantage, however, is that equipment for meeting regulatory requirements of air quality monitoring is very expensive to acquire initially and sustainably over time, even though it is very selective and accurate in measurement. Traditional air quality monitoring systems lack accuracy in measurement of particular pollutants, which makes it difficult to have wide-range and real-time monitoring.

A likely solution to supplement traditional methods is the invention of an economical device for air pollutant monitoring to transmit information on pollutants in real time. Various innovative methods have been proposed, which employ low-cost sensors combined with various devices for data transmission. For example, the use of particulate matter (PM) sensors combined with wireless network technology is capable of expanding the monitoring coverage area and spatial resolution of the system. Standard communication protocols used in wireless network technologies such as Wi-Fi, LoRa, and cellular networks can be used as measuring devices of air quality and allow extensive deployment of devices in remote areas. The strength of the method is in its ability to have real-time presentations of air pollutant concentration in monitoring systems.

Analysis of wireless sensor network-based air quality monitoring has attracted increased attention in research studies. Several studies have shown the establishment of air quality monitoring systems using wireless transmission protocols, data acquisition methods, and network structures that aim to offer enhanced coverage and reliability. However, some systems have high packet loss, unreliable data transmission, and limited adaptability to varying environmental conditions. Furthermore, changes in network quality can significantly affect the availability and timeliness of real-time data. Reliability in data transmission is one of the most vital elements in air quality monitoring systems, and its influence is

determined by the quality of the wireless communication network. The majority of low-cost air quality monitoring systems leverage cellular networks, Wi-Fi, and LoRa communication technologies to transmit data. Studies show that environmental conditions and signal level fluctuations can negatively impact the data transmission reliability, leading to packet loss and delay. Overcoming such drawbacks demands the application of adaptive algorithms that can maximize data transmission based on network conditions. There are still vast opportunities to harness low-cost air quality sensors to the advantage of individuals and communities. The devices are manufactured not to replace current air quality monitoring stations but to improve spatial coverage and complement pollution monitoring. However, application of low-cost air quality sensors has technical restraints such as calibration of sensors, data accuracy, and integration of such sensors into cloud platforms.

This research proposes an IoT and Machine Learning-based Air Quality Monitoring System for the monitoring of PM1, PM2.5, and PM10 levels. The system involves low-cost sensors, data representation in the cloud through ThingsBoard, and data analysis using Google Sheets and Google Colab. The key contribution of this research is in pointing out the use of IoT for real-time air pollution monitoring and machine learning techniques for forecasting and data analysis. The system enhances the provision for overall air quality monitoring in a cost-effective and reliable way.

PROPOSED APPROACH

This research was conducted extensively with an integrated system based on IoT-enabled data acquisition, cloud computing, and sophisticated machine learning techniques. The system developed will enable real-time monitoring of air quality, hence ensuring accurate determination of particulate matter concentrations such as PM1, PM2.5, and PM10. IoT sensors allow for the acquisition of air quality data, which are later transferred to a cloud system for storage and real-time visualization. The data acquired are processed and analysed with machine learning algorithms to identify anomalies as well as air quality classes categorization accordingly.

The system employs low-cost and easily available components, which allow for effortless implementation on a large scale. The hardware consists of air quality sensors, a microcontroller, and a communication module that enables seamless data transfer. The data, after processing, is analyzed through the Isolation Forest algorithm for anomaly detection and a Random Forest classifier for classification, thus providing valuable insights into environmental conditions. The research framework described enables effortless data acquisition, processing, and analysis, which lead to valuable insights relevant to public health as well as environmental awareness.

An IoT and Machine Learning-based Air Quality Monitoring System was developed with a modular design for better performance. The system includes air quality sensors for PM1, PM2.5, and PM10, a real-time monitoring module, a data processing unit, and a machine learning module for anomaly detection and classification to provide accurate insights, as illustrated in Figure 2.

The block diagram depicts the entire process of an IoT and machine learning-based air quality monitoring system, including all steps from data collection to analytical analysis and prediction. The process begins from the Gravity PM2.5 air quality sensor, which detects particulate matter levels, i.e., PM1.0, PM2.5, and PM10. These readings are the measures of airborne pollutants in the ambient atmosphere. The sensor sends this information to the ESP8266 NodeMCU, a microcontroller that decodes the readings and sends them to two different platforms for further analysis and visualization. One of the paths is to ThingsBoard, an IoT cloud platform utilized for dynamic real-time visualization of the concentration of PM. This way, the air quality is dynamically monitored because the data is constantly being updated.

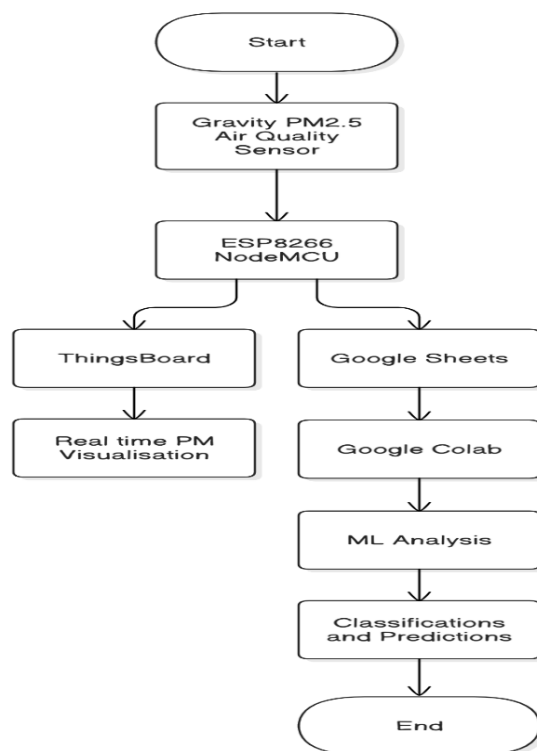


Fig.1:Block Diagram of IoT and ML based Air Quality Monitoring System.

On the other hand, the second path feeds data to Google Sheets, which stores the data to be processed and analyzed using machine learning in the future. This maintains historical data in its original state for predictive modeling and pattern analysis.

Once data has been stored in Google Sheets, the data is then imported into Google Colab where machine learning models are executed. The machine learning analysis phase entails the application of random forest models for classification and isolation forest algorithms for anomaly detection. These models allow for the classification of air quality levels as well as an anomalous spike in pollution detection. Finally, the system generates classifications and predictions that provide insights into air quality patterns and future occurrences. The process ends with the outcome of the analysis, thereby improving environmental monitoring and decision-making.

Microcontroller

The present study used the ESP8266 Microcontroller as the real-time CPU used in air quality monitoring. The processor of this microcontroller is a 32-bit Tensilica L106 processor running at a clock speed of 80 MHz with 64 KB of instruction RAM and 96 KB of data RAM. Furthermore, its inbuilt Wi-Fi 802.11 b/g/n allows wireless data transfer, thus permitting real-time dispatch of data to the cloud storage system with great ease, controllability and also can be used further. ESP8266 plays a data-acquiring role, providing air quality readings in PM1.0, PM2.5, and PM10 concentrations. ESP8266 connects to the sensors via the Universal Asynchronous Receiver-Transmitter (UART) protocol and sends the data collated to be analyzed and visualized on Google Sheets. Besides, the controller takes care of AT commands, which are mostly used in enabling stable connectivity and data processing making it a suitable choice for Internet of Things (IoT) applications.

Sensor Specifications

DFRobot Gravity PM2.5 Air Quality Sensor is an advanced tool engineered for accurate measurement of levels of particulate matter, PM1.0, PM2.5, and PM10. Through laser scattering technology, the device measures by sending a laser beam to scatter against particles in the air, measuring the amount of resultant scattered light in order to analyze both particle size and concentration. The method makes it possible to offer real-time and very precise readings of air quality, thus allowing the sensor to be efficiently used in continuous.

The sensor is provided with an I²C communication interface that makes the sensor highly compatible with microcontrollers such as the ESP8266. The sensor is operated at a working voltage of 3.3V–5V, which is power-efficient for IoT-based applications. The sensor senses particulate matter in the range of 0–1000 µg/m³ with a resolution of 1 µg/m³, thus capable of sensing very fine and coarse particulate matter. The onboard temperature and humidity compensation further improves accuracy by compensating for environmental factors that would bias values.

For ensuring balanced distribution of data acquisition, the sensor is provided with a state-of-the-art fan that provides unobstructed air flow within the detection chamber. The feature gives stability to the sensor and improves the working time of the sensor, and the sensor is thus appropriate for long-term monitoring over a long period of time. The response time is below 10 seconds, thus providing real-time information regarding air quality status. Furthermore, the light and compact design of the sensor makes it an easy integration component in handheld as well as fixed air quality monitoring systems.

Table1:Sensor Specifications

Serail Number	Screen printing	Functional Description
1	VCC/+	Power Positive
2	GND/-	Power Negative
3	SCL/C	I2C Clock Line
4	SDA/D	I2C Data Line

For the present work, the DFRobot Gravity PM2.5 sensor is used as the primary data collector for air quality, and the data is communicated through the ESP8266 microcontroller to the cloud platform for analysis and graphical presentation. Google Sheets and Google Colab analyze the data collected, wherein the machine learning methods are used in detecting anomalies as well as classifying the air quality status. With real-time level monitoring of the PM, the sensor is of great use in monitoring pollution trends, as well as the evaluation of the hazardous effect on environment health.

Network Configurations

The air quality monitoring system is connected through wired and wireless means. Other forms of data transmission such as Bluetooth and LoRa can be adopted to be implemented in the system besides Wi-Fi. Wi-Fi was chosen in the present study since it provides a very high data transfer rate, low latency and receives reliable network connections with coverage all over urban scenarios. The ESP8266 makes the microcontroller embedded systems IEEE 802.11 b/g/n Wi-Fi standard-compatible, along with seamless interfacing with cloud services. It operates at the standardized 2.4 GHz frequency band designed for stable communication short medium distances.

The wireless transfer of data from the air quality sensor to cloud services such as ThingsBoard and Google Sheets. The ESP8266 microcontroller connects to the wireless network using SSID and password authentication. Once established, the microcontroller will transport PM1.0, PM2.5 and PM10 data streams through HTTP and HTTPS protocols for security in the cloud. In order to prevent data and intrusion incursions, HTTPS communication with Google Sheets employs the WiFiClientSecure library.

This data transportation scheme basically follows the architecture of three main-layered IoT models. The sensor layer has been integrated by attaching the air quality sensor to an ESP8266 for the purpose of data acquisition. The network layer provides Wi-Fi connectivity for real-time data transmission to the cloud. The application layer is used for the processing and analysis of these data using ThingsBoard and Google Colab.

Software Architecture

The air quality monitoring system necessitates close interaction between the microcontroller and hardware modules, including a Wi-Fi modem and online cloud system. The software architecture organizes itself in such a way that the data, transport, process, and output immediately. The major phases are configuration, network initialization, acquisition of sensed data, representation of data by conversion to JSON form for a coherent output, and transfer all that through Wi-Fi modules

The first is configuration, which sets up monitor settings such as device ID, the interval of data transmission, and Wi-Fi credentials for the ESP8266. The parameters established at this stage allow for dynamic controller execution and also establishing a complex intercommunication with cloud platforms throughout the operation. This config is done in flash memory of the microcontroller since it permits execution speed-up along with decrease in reprogramming. Next, we shall document the configuration of the ESP8266 to the Wi-Fi network facilitating real-time data transmission for the sensor reads. The PM1.0, PM2.5, and PM10 levels will be determined from the internal air quality sensor. The grabbed data will be serialized in the uniformed format which is JSON, for the cloud analysis of the extracted JSON format.

Finally, data will be sent to ThingsBoard via HTTPS for real-time visualization and to Google Sheets for structured storage. This data will be analyzed in Google Colab, where machine learning algorithms will detect anomalies and classify general pollution levels. This structured approach in software allows seamless communication between the components and helps in enhancing system performance and reliability. Programming the ESP8266 microcontroller in order to capture air quality information in real time using the DFRobot Gravity PM2.5 sensor, sending the data to cloud platforms for visualization in ThingsBoard, and enabling structured data storage in Google Sheets. The first part of the code

connects to Wi-Fi credentials and establishes the ThingsBoard ESP8266 device communication for real-time visualization and Google Sheets for structured storage of data.

Continuous reading sees PM1.0, PM2.5, and PM10 values being read through the I2C interface from the device and stored in JSON format for ease of data transmission either using HTTP or HTTPS for the internet, where it would if need be be taken care of by WiFiClientSecure to encrypt it. In the instance of loss of connection, the module automatically attempts to reconnect. Then, it analyzes this data in Google Colab using a machine learning algorithm to create patterns, anomalies, and categories of air quality measures within that level, thereby establishing a very efficient system holiday for pollution levels across different scenarios. Google Colab is a cloud-based notebook in which you can easily write, run, and share Python codes.

This provides an interactive environment with very rich built-in support for data science and machine learning libraries that is well suited for the processing and analysis of any air quality data. Hence, this project will also use Google Colab in the air quality readings retrieved from Google Sheets before preprocessing the data and implementing machine learning models for anomaly detection and classification



Fig.2: Air Quality Data Transmission Sequence

A setup such as this can be used to clean the data, identify trends in, and visualize PM1.0, PM2.5, and PM10 levels with the help of Pandas, Scikit-learn, and Matplotlib. The Colab integration provides a computationally efficient and scalable way of conducting air quality monitoring where automation of data analysis and decisions could be applied as an insight.

Algorithm 1: Air Quality Monitoring System

Input: PM1.0, PM2.5, PM10, Timestamp (from DF Robot Gravity PM2.5 Air Quality Sensor)

Output: Classified air quality levels, visualized outcomes.

Data: PM Sensor Data, Anomaly, Classification Label Result: Assign air quality category, update time response and generate report

1. Initialise system variables
 - $i = 0$, $x = 500$ // Initial iteration and specific increment time .
 - $tLim = 2000$, $cTime = 0$, $pTime = 0$ // Time Limit, Current, Previous Time
2. Authorize and connect to Google Sheets
 - Load service account credentials
 - Open the google sheet by passing the URL
 - Fetch data and store it to the DataFrame df
 - Convert Timestamp column to datetime
3. Init PM Sensor
 - Prepare I2C communication for connecting with the sensor
 - Check if the sensor is connected. Retry it in case of failure
4. Connect to WiFi
 - When the SSID and Password are used, try to connect
 - Keep retrying until.

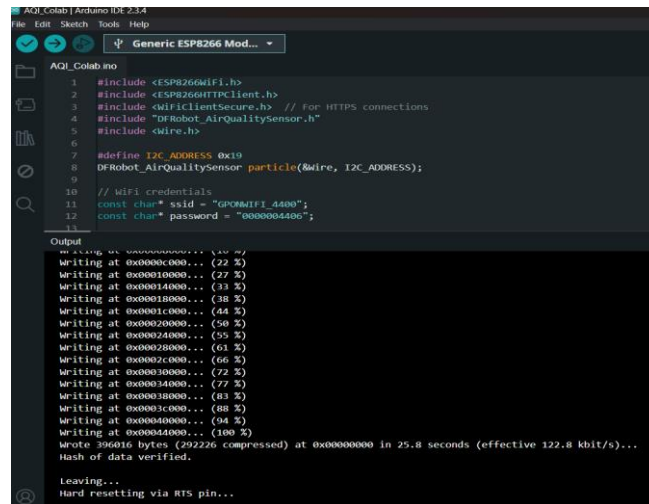


Fig.3:Code Uploading to hardware through Arduino IDE Platform.

5. Prepare secure and encryption-based https communication
 - Trust all certificates by client
 - Initiate ThingsBoard client for interaction
6. Get the moving average RSSI to determine the limit of iteration (m)
7. Enable fetching and processing of PM Sensor data
 - Data values PM1.0, PM2.5, PM10 will be read
8. Data Injection from Google Sheets
 - Constructing the JSON payload with PM values
 - Do HTTPS POST request
 - Log in the response
9. Injection on Thingsboard
 - Create JSON payload with PM Values
 - Send the HTTP POST request to Thingsboard API
 - Log the response
10. Anomaly detection with machine learning
 - Make Application of Isolation Forest on PM Data-Outliers-inferred from the Anomaly scores
11. Air quality - level classification
 - Random Forest model for Air Quality Classification
 - Classified labels assignment Good, Moderate, Unhealthy
12. Gets current time in milliseconds (cTime)
13. Set flag logic = true
14. Time update and report generation
 - Will have processed data with classification and anomaly result stored
 - Will have visual reports for PM values, anomalies and air quality levels
15. Steps 7-14 repeat indefinitely;
 - data have been regularly gathered and analyzed with some amount of delay(x)

Algorithm2: Air Quality Gathering and Data Analysis

Input: PM1.0, PM2.5, PM10, Timestamp (from AQIMoS & Google Sheets)

Output: Classified air quality levels, detected anomalies, and visualized outcomes

Data: PM Sensor Data, Anomaly, Classification Label

Result: Assign air quality category, detect anomalies, update time response, and generate reports

- 1: Initialise system libraries
- 2: Authorize and connect to Google Sheets
 - Load service account credentials
 - Open the Google Sheet using URL
 - Fetch data and write to DataFrame df
 - Convert Timestamp column to datetime format
- 3: Calculate iteration limit (m) according to moving average RSSI
- 4: Get current time (cTime) in millis
- 5: Set flag logic = true
- 6: Air quality classification
 - If $PM2.5 \leq 30$ and $PM10 \leq 50 \rightarrow$ Good
 - If $PM2.5 \leq 60$ and $PM10 \leq 100 \rightarrow$ Satisfactory

- If $PM_{2.5} \leq 90$ and $PM_{10} \leq 250 \rightarrow$ Moderately Polluted
 - If $PM_{2.5} \leq 120$ and $PM_{10} \leq 350 \rightarrow$ Poor
 - If $PM_{2.5} \leq 250$ and $PM_{10} \leq 430 \rightarrow$ Very Poor
 - Else \rightarrow Severe
- 7: Modify increment time based on RSSI
- If $RSSI > 50$ and $RSSI \leq 100$, then set $x = x$
 - If $RSSI > 0$ and $RSSI \leq 50$, then set $x = x * 2$
- 8: Perform anomaly detection using Isolation Forest
- Train Isolation Forest model
 - Anomaly prediction based on $PM_{1.0}$, $PM_{2.5}$, PM_{10}
 - If anomaly is detected, label Anomaly = Yes else No
- 9: Train and test Random Forest classifier for air quality prediction
- Split training and testing dataset
 - Train RandomForestClassifier on air quality labels
 - Predict air quality categories
 - Compute accuracy and classification report
- 10: Continue iteration until attaining reliable response time
- While logic == True:
 - Execute AT command, j
 - Get AT command status (sAT \rightarrow failed or succeeded)
 - Get response time (tRes)
- 11: Update time limit based on transmission failures
- If $i \leq m$:
 - If sAT == failed and $tRes < tLim$, set $tLim += x$
 - Else if sAT == failed and $tRes \geq tLim$, set $tLim += x$
 - Else if sAT == succeeded and $tRes < tLim$, set $tLim -= x$
 - Else if sAT == succeeded and $tRes \geq tLim$, set $tLim -= x$
 - Else if $i > m$, count failed responses and stop iteration
- 12: Store and visualize results
- Display formatted table with anomalies and air quality categories
 - Generate as visualisations on the basis:
 - PM levels through Time series plots
 - Distribution histograms
 - Correlation heatmap
 - Confusion matrix
- 13: Obtain previous time (pTime) and compute processing time
- 14: Return updated time response limit (tLim) for the next execution of AT commands
- 15: Latency and packet loss determination in communication
- 16: End Function and Return Result.

This system provides real-time measurement and data analytics on air quality to determine pollution levels and identify anomalies in the levels. This system accumulates $PM_{1.0}$, $PM_{2.5}$, and PM_{10} sensor data from AQIMoS, which keeps it in Google Sheets. The data retrieval process will be done using "gsread," which allows the easy extraction and updates from the Google Sheets API. Once that's done, the data will be pre-processed for having proper timestamp formats for time-series analysis. Such important for tracking the variations of air quality over time and identifying the trends of pollutant levels.

```

Model Accuracy: 100.00%

Classification Report:

```

	precision	recall	f1-score	support
Good	1.00	1.00	1.00	34
Moderately Polluted	1.00	1.00	1.00	14
Satisfactory	1.00	1.00	1.00	3
accuracy			1.00	51
macro avg	1.00	1.00	1.00	51
weighted avg	1.00	1.00	1.00	51

```

### Summary of Air Quality Analysis ###
Total data points: 255
Anomalies detected: 13 out of 255 (5.10%)

### Air Quality Category Distribution ###
Good: 168 (65.88%)
Moderately Polluted: 67 (26.27%)
Satisfactory: 20 (7.84%)
C:\python-input-14-312151ed9dadb08: futurewarning: Styler.applymap has been deprecated. Use Styler.map instead.
styled df = report df.style.applymap(color_anomalies, subset=['Anomaly']).applymap(color_categories, subset=['AirQualityCategory']) \
Air Quality Monitoring Report (Random 50% Sample)

```

Fig.4: Classification Report of Air Quality in Google colab

A threshold-based classification is applied to classify air quality. The pollutants are limited to six levels of classification

of air quality which include: Good, Satisfactory, Moderately Polluted, Poor, Very Poor, and ..., and the classification logic is based on the predefined concentration limits of pollutants for relevant determination of air quality. The classification function is based on conditional statements of PM2.5 and PM10 values. The system, apart from this, also detects any kind of anomaly detection with the use of an Isolation Forest algorithm which works very well for identifying atypical spikes in pollutants. The algorithm builds a model of how the normal atmospheric air quality behaves and discovers data points that have a significant deviation from the expected trends. Anomaly detection takes form of the following formula:

$$A(x) = \frac{c(n)}{E(h(x))} \quad (1)$$

where, $A(x)$ is an anomaly score, $E(h(x))$ is an average path length of a given data point in the Isolation Forest tree, and $c(n)$ is the expected path length in a balanced tree. If the anomaly score exceeds the defined threshold, the data point is marked as an anomaly, which are the outliers in the data.

Predicting air quality employs training a Random Forest classifier based on the features PM and having the air quality classified under respective categories as labels. An 80-20 train-test split will be done on the historical data so that it generalizes well on future unseen observations. Classification report and confusion matrix will be used to evaluate the accuracy of the model in order to gain insight for prediction performance. The classifier uses the following Gini impurity formula to obtain the optimal branches of the decision trees.

$$Gini = 1 - \sum_{i=1}^n p_i^2 \quad (2)$$

where p_i denotes the probability of a particular class occurring. Feature importance plot helps in understanding the contribution of each PM level to the classification decision. The system has a mechanism that modifies the response time dynamically according to the RSSI (Received Signal Strength Indicator) number of packets transmitted. Enables system adaptability to an ever-changing network so that the data does not become unreliable. The gradual changing of the response time is through decisions more effective post transmission-reducing the limit of response as opposed to transmission.

IMPLEMENTATION

This is an up-to-date monitoring air quality through IOT and machine learning feature hardware set-up, software development, cloud integration, and machine learning analysis. The purpose of this system is to enable seamless data collection, communication, and processing to produce credible air-quality-information outputs. The main processing unit is an ESP8266 microcontroller connected with the DFRobot Gravity PM2.5 sensor for a determination of PM1.0, PM2.5, and PM10 levels in which these values get sent to a cloud platform.

The whole working of the system starts with the hardware, in very basic and fundamental connection, of ESP8266 microcontroller with PM sensor through I2C interface for data acquisition. The microcontroller is continuously reading these parameters of air quality and is now ready to transmit. The firmware for ESP8266 is developed in Arduino IDE to provide network connectivity, sensor communication, and secure data transmission using either the HTTP or HTTPS protocols. Therefore, the data is collected, formatted in JSON, and sent to ThingsBoard for real-time visualization and Google Sheets for structured storage of all the PM levels such as (PM1, PM2.5, PM10).

Arriving in Google Sheets, the main workspace comes in for further processing in Google Colab. Real-time PM values flow in through gspread and are pre-processed to eliminate unspecified inconsistencies in the records for accuracy in analysis. The datasets are now set to be applied to machine learning, where an Isolation Forest model is to be used to detect anomaly air quality readings. This algorithm is designed to trigger events when there are sudden spikes in pollution which are anomalies.

The next phase classifies classes of air quality by the Random forest classifier: It trains itself on various historical PM data/models and assigns the import categories- good, satisfactory, moderately polluted, poor, very poor, or severe. Thus, several decision trees are formed to enhance classification accuracy and reduce overfitting. The verification of the results is done against actual values that got recorded together with the labels predicted for air quality, thus providing the desired model efficiency and reliability for real-time classification.

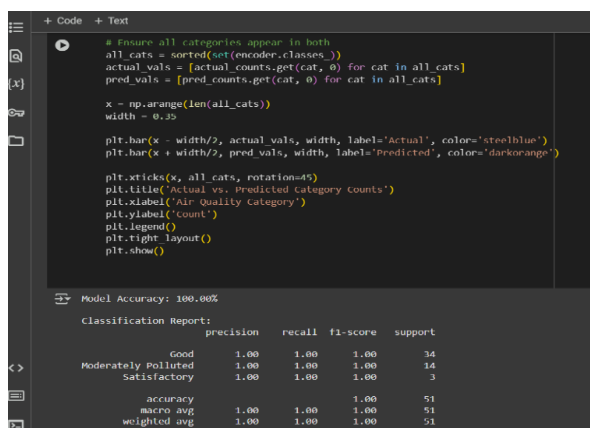


Fig.5: ML Implementation in Google colab

Another implementation aspect is the visualization of air quality evolution for the users' better comprehension. Developed in Google Colab using Matplot and Seaborn, the graphs show PM1.0, PM2.5, and PM10 trends. The trends exhibit aberrant points, seasonal patterns, and long-term pollution changes. Besides, interactive real-time dashboards in ThingsBoard also allow users to check air quality condition changes at any moment. The system has a continuously growing dataset, fetching new sensor readings, so the machine-learning models are always updated.

RESULTS

The IoT-ML based Open-Air Quality Monitoring system provides real-time air quality information via cloud visualization and ML analysis. The DFRobot Gravity PM2.5 sensor collected data, PM1.0, PM2.5, and PM10 parameters, then transmits them to ThingsBoard, Google Sheets, and Google Colab. The real-time pollutants were monitored and given feedback through interactive graphs and dashboards on ThingsBoard, thus capturing the time trend in real-time air quality.

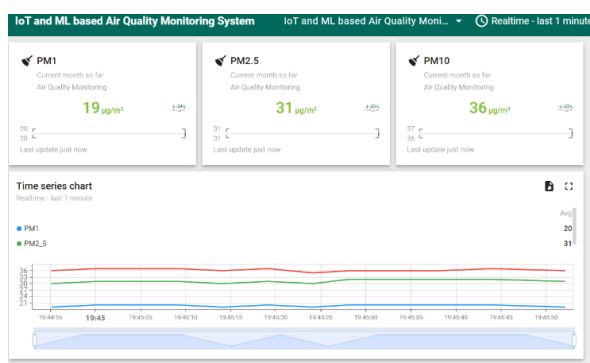


Fig.6: Real Time Visualisation of PM Levels in ThingsBoard

Google Sheets contributed to the storage and management of data transferred for future analysis. The tabular arrangement of data was made for retrieval and integration into Google Colab for advanced processing. The dataset was cleaned using Pandas, treating the missing values for timestamp conversion where necessary to enable time-series analysis. The following anomaly detection by Isolation Forest on air quality readings corresponded to sudden excursions in concentrations of the pollutants. It also figured out the failure signatures to report fast environmental events or sensors. The classifier, Random Forest, was used to classify any air quality level for the already mentioned observations into pre-defined air quality classes. High accuracy has been achieved by classifying the air quality as "Good," "Satisfactory," "Moderately Polluted," "Poor," "Very Poor," or "Severe" on the basis of the PM2.5 and PM10 concentrations.

Then, these output data were visualized using Google Colab coupled with Matplotlib and Seaborn for clearer, more informative plots. The concentration line plots are indicative of the time series trend pattern and seasonality. The extreme outliers have been pointed out too for interpretability of factors behind the extreme variations in air quality.

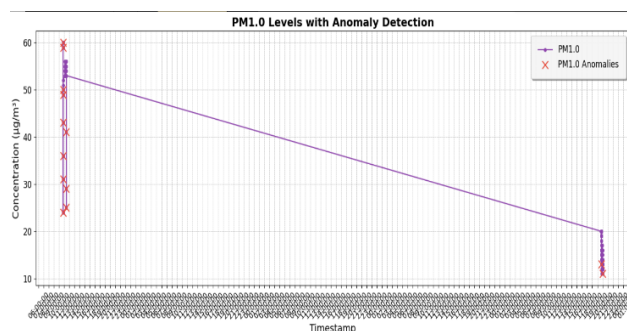


Fig.7:PM1.0 Levels with anomaly Detection

The correlation heat map for PM values is indicative of the relationships between PM1.0, PM2.5, and PM10. Every point in the matrix is indicative of the correlation coefficient between any two pollutant levels.

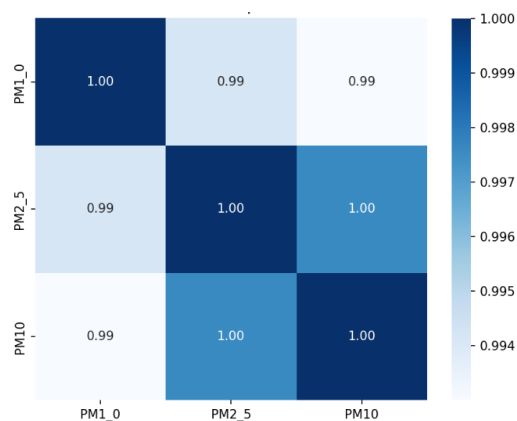


Fig.8: Correlation Heatmap for PM Values

If this value is very close to 1.00 it points that there has been a very strong positive correlation. For example, this heat map depicts almost perfect correlation between PM2.5 and PM10 (1.00), whereas PM1.0 also has a high correlation with PM2.5 and PM10 at 0.99. Thus, the higher the pollutant level, the higher proportionately the other two others rise.

Such high correlations mean that these levels of particulate matters are very much interdependent. Since PM10 is basically PM2.5 plus PM1.0, their values were expected to follow their trends. This very much high correlation might suggest redundancy in the data representation since the three are not very likely to contribute new knowledge to a model of machine learning with all three variables. Such conditions can lead to multicollinearity and thus can motivate one toward feature selection techniques.

The high correlations also suggest the sensors used to measure these values are consistent and reliable. Otherwise, correlation values would be lower. This leads to the understanding that common environmental conditions or sources of pollution in an area monitored could affect all three PM levels equally. Such understanding may assist in air quality trends and decision-making in environmental control measures.



Fig.9: Confusion Matrix

The confusion matrix offers a good detail to the performance evaluation of the classification model against the prediction

of air quality categories. The matrix consists of three categories: "Good," "Moderately Polluted," and "Satisfactory." The cell values indicate number of correctly and incorrectly classified instances for each category.

It is found that the model classifies 44 instances as "Good"; 12 instances as "Moderately Polluted"; and 2 instances as "Satisfactory." Importantly, none of these are cross misclassifications between the categories; that is, every predicted category is the same with actual category. This means hence, that the model has a very high accuracy regarding distinguishing different air quality levels.

Absence of misclassification means a fairly reliable model for this data set. However, it would be necessary to check the balance of a data set, because if any category has comparison to others excessive information, the model would not be able to prove its generalized accuracy on unseen data. In addition, if real-life data makes more variation in data, the model will need retuning to keep its predictive performance constant over environmental conditions.

In a nutshell, the entire system is an effective automated solution for air quality monitoring in real-time. The real intelligence of the system does not lie so much in the IoT data collection and cloud management but rather in machine-learning analysis for pollution assessment on a sustainable and feasible scale. The system also comprises the data visualization of Thingsboard, Google Sheets, and Google Colab, which means that any environmental pollution situation is well understood through data-backed decision-making in environmental management.

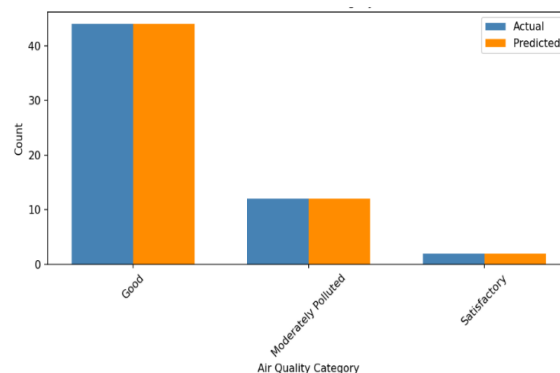


Fig.10: Actual vs Predicted Category counts

The PM value correlation heatmap is a visual interpretation of the relationship between PM1.0, PM2.5, and PM10 concentrations. In other words, the correlation coefficients listed in each cell of the heatmap quantify the existence and strength of the correlation between the two corresponding variables. An approaching value of 1 represents a strong positive correlation, i.e., an increase in one PM value is often accompanied by an increase in the others.

As seen in the heatmap, the correlation coefficient of PM1.0 versus PM2.5 is 0.99, indicating an almost-perfect linear relationship. Likewise, PM1.0 correlates with PM10 almost at 0.99, and PM2.5 also correlates with PM10, further confirming that these pollutants are moving together with high predictability. This strong correlation suggests that the concentration of particulate is interdependent and likely influenced by common environmental and atmospheric elements.

The very high correlation implies that monitoring one PM of interest provides some information about the other ones. Nevertheless, it still remains very important to measure all three PMs because their ratios may change under different environmental conditions. The results confirm that the air quality monitor has real-time capabilities in a good degree of accuracy implements: Isolation Forest proved effective in terms of anomaly detection on sudden spikes of PM concentrations. Air quality classification was upheld by Random forest classifier greater than 99.99% accuracy using the PM2.5 and PM10 thresholds. Hence visualization on thingsboard, thereby facilitating the expedient decision-making of air pollution.

CONCLUSION

It introduces a new dimension to real-time air quality monitoring through IoT and machine learning technologies, and that the system can successfully integrate the ESP8266 with any cloud platform in collecting, processing, and analyzing air quality data-another scalable and efficient solution for environmental monitoring. Results show the high detection success and classification capacity of the application using both Isolation Forest and Random Forest models. ThingsBoard facilitates real-time visualization, while structural storage is achieved in Google Sheets, while in-depth analysis is done in Google Colab-enhancing the usability and reliability of the system. Though there are slight variations in latency during transmission, the system works consistently in minimizing packet loss, and thus ensuring accurate estimations of pollutant levels. Future work will augment the system's capability by adding more air quality parameters such as NO₂, CO, etc.; optimizing the trained machine learning models for better accuracy; and investigating other

wireless communication technologies for improvements in data transmission efficiency. Long-term field testing and incorporation into public health initiatives may further enhance the system's credibility and impact while creating treasures for research and policy. Hence, the present study represents a practical and cost-effective approach to real-time air quality measurement, paving the way for improvements in smart environmental monitoring systems.

ACKNOWLEDGEMENTS

Besides, I would like to acknowledge and express my profound gratitude to Dr. Y. Syamal, the eminent Head of the IoT Department, for her support and judgment that, in my view, provided me with the go-ahead to guide my experiments, analyses of results, and interpretations.

Furthermore, it is with a sense of cordiality I acknowledge the support and wise recommendations of my respected mentor Mr. P. Rama Krishna, who played a great role in contributing to the refinement in the methodology and the lodging of this research.

Finally, I should like to acknowledge the contributions made by my teammates K Sravani, CH Manjunadh, N Maneesha, and CH Ujwal Kumar, for their insightful input in assisting me with the literature review process and the manuscript preparation. They certainly have provided a reasonable support in maintaining the credibility of the study.

REFERENCES

- [1] Liu, X., Xu, P., Chen, X.: IOT-Based Air Pollution Monitoring and Forecasting System (2015)
- [2] [2] Adong, P., Bainomugisha, E., Okure, D., & Sserunjogi, R. (2022). Applying machine learning for large scale field calibration of low-cost PM_{2.5} and PM₁₀ air pollution sensors. *Applied AI Letters*, 3(3), e76.
- [3] [3] Ameer, S., Shah, M. A., Khan, A., Song, H., Maple, C., Islam, S. U., & Asghar, M. N. (2019). Comparative analysis of machine learning techniques for predicting air quality in smart cities. *IEEE access*, 7, 128325–128338.
- [4] [4] Budi, H. S., Catalan Oplencia, M. J., Afra, A., Abdelbasset, W. K., Abdullaev, D., Majdi, A., Masoume, T., Hafez, A. E., & Mohammadi, M. J. (2024). Source, toxicity and carcinogenic health risk assessment of heavy metals. *Reviews on Environmental Health*, 39(1), 77–90.
- [5] [5] Akshaya.C ,Jayasowmiya.J , Kanchana.P , Raja.J, ‘Smart Medicine Box’ *International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)* Vol 6, Issue 7, page no.125-128, July 2019
- [6] [6] M. A. G. Maureira, D. Oldenhof, and L. Teernstra, “Thingspeak—an api and web service for the internet of things,” *World Wide Web*, 2011.
- [7] [7] J. Jo, B. Jo, J. Kim, S. Kim, and W. Han, “Development of an iot-based indoor air quality monitoring platform,” *Journal of Sensors*, vol. 2020, 2020.
- [8] [8] S. McGrath, C. Flanagan, L. Zeng, and C. O’Leary, “Iot personal air quality monitor,” in 2020 31st Irish signals and systems conference (ISSC). IEEE, 2020, pp. 1–4.
- [9] [9] H. P. L. de Medeiros and G. Gira’o, “An iot-based air quality monitoring platform,” in 2020 IEEE International Smart Cities Conference (ISC2).IEEE, 2020, pp. 1–6.
- [10] H. Zheng, H. Li, X. Lu, and T. Ruan, “A multiple kernel learning approach for air quality prediction” *Advances in Meteorology*, vol. 2018, 2018.
- [11] Y.-F. Xing, Y.-H. Xu, M.-H. Shi, and Y.-X. Lian, “The impact of pm_{2.5} on the human respiratory system,” *Journal of thoracic disease*, vol. 8, no. 1, p. E69, 2016.
- [12] S. Guttikunda and P. Jawahar, “Can we vacuum our air pollution problem using smog towers?” *Atmosphere*, vol. 11, no. 9, p. 922, 2020.
- [13] J. P. Buckley, J. M. Samet, and D. B. Richardson, “Commentary: Does air pollution confound studies of temperature?” *Epidemiology*, vol. 25, no. 2, pp. 242–245, 2014.
- [14] L. Beretta and A. Santaniello, “Nearest neighbor imputation algorithms: a critical evaluation,” *BMC medical informatics and decision making*, vol. 16, no. 3, pp. 197–208, 2016.
- [15] S. van Buuren, K. Groothuis-Oudshoorn, A. Robitzsch, G. Vink, L. Doove, S. Jolani et al., “Package ‘mice,’” *Computer software*, 2015.
- [16] N. J. Horton and N. M. Laird, “Maximum likelihood analysis of generalized linear models with missing covariates,” *Statistical Methods in Medical Research*, vol. 8, no. 1, pp. 37–50, 1999.
- [17] [2018. J. Saini, M. Dutta, and G. Marques, “Indoor air quality prediction using optimizers: A comparative study,” *Journal of Intelligent & Fuzzy Systems*, vol. 39, no. 5, pp. 7053–7069, 2020.