
**AIR QUALITY MONITORING SYSTEM
PROJECT REPORT**

**21CSE293P – Artificial Intelligence and Industrial Internet of Things
(2021 Regulation)**

**II Year/ IV Semester
Academic Year: 2023 -2024
By**

RISHIRAM B (RA2211026010553)

KEERTHIPAVAN SUGANANAM (RA2211026010561)

SURYA SATHYANARAYANAN (RA2211026010562)

Under the guidance of

Dr. A. ALICE NITHYA

Associate Professor

Department of Computational Intelligence



FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Kancheepuram

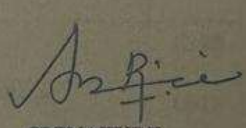
May 2024

BONAFIDE

This is to certify that "21CSE293P – Artificial Intelligence and Industrial Internet of Things" project report titled "AIR QUALITY MONITORING SYSTEM" is the bonafide work of SURYA SATHYANARAYANAN (RA2211026010562), RISHIRAM B(RA2211026010553), KEERTHIPAVAN SUGANANAM(RA2211026010561) who undertook the task of completing the project within the allotted time.


SIGNATURE

Dr. A. Alice Nithya
AIIOT – Course Faculty
Associate Professor
Department of Computational Intelligence
SRM Institute of Science and Technology
Kattankulathur


SIGNATURE

Dr Annie Uthra R
Head of the Department
Professor
Department of Computational Intelligence
SRM Institute of Science and Technology
Kattankulathur

TABLE OF CONTENTS

CHAPTER NO	CONTENTS	PAGE NO
1	INTRODUCTION	4-5
	1.1 Motivation	
	1.2 Objective	
	1.3 Problem Statement	
	1.4 Challenges	
2	LITERATURE SURVEY	6-8
3	REQUIREMENT ANALYSIS	9-10
4	ARCHITECTURE & DESIGN	11
5	IMPLEMENTATION	12-15
6	EXPERIMENT RESULTS & ANALYSIS	16-19
7	CONCLUSION	22
8	REFERENCES	23

1. INTRODUCTION

1.1 Motivation:

The motivation behind developing an IoT-based air quality management system stems from the critical need to address escalating concerns regarding environmental degradation and its impact on public health. With rapid industrialization, urbanization, and increasing vehicular emissions, the quality of the air we breathe has become a pressing global issue. Traditional monitoring methods often lack real-time capabilities and are limited in scope, making it challenging to capture dynamic air quality fluctuations accurately. By harnessing the power of IoT technology, we aim to create a solution that provides continuous, real-time monitoring of air quality parameters, enabling stakeholders to make informed decisions and take proactive measures to mitigate pollution levels.

1.2 Objective:

The primary objective of this project is to design and implement an IoT-based air quality management system that offers reliable, real-time monitoring and management of air quality parameters. Specifically, our goals include:

- Developing a robust hardware setup utilizing NodeMCU, MQ135 gas sensor, and DHT sensor for accurate data collection.
- Integrating cloud-based platforms such as ThingSpeak to store, analyze, and visualize air quality data.
- Providing stakeholders with user-friendly interfaces for accessing real-time air quality information and analytics.
- Facilitating remote monitoring and management of air quality, enabling timely interventions to improve environmental conditions.

1.3 Problem Statement:

The current methods for monitoring air quality often suffer from limitations such as lack of real-time data, limited spatial coverage, and high costs associated with equipment and maintenance. Moreover, existing solutions may not be easily accessible or scalable, particularly in resource-constrained environments. The problem statement revolves around the need for a comprehensive, cost-effective, and scalable air quality management system that addresses these limitations. Specifically, the system should be capable of:

- Collecting real-time data on air pollutants, temperature, and humidity levels.
- Analyzing and visualizing air quality trends and fluctuations over time.
- Providing actionable insights to stakeholders for decision-making and intervention strategies.
- Ensuring scalability and interoperability to accommodate diverse environmental monitoring needs.

1.4 Challenges:

Several challenges need to be addressed in the development and implementation of the IoT-based air quality management system. These challenges include:

Ensuring the accuracy and reliability of sensor data in varying environmental conditions.

Optimizing power consumption to prolong the operational lifespan of battery-powered devices.

Designing a scalable and secure cloud infrastructure for data storage and analysis.

Developing user-friendly interfaces that cater to the diverse needs of stakeholders, including policymakers, environmentalists, and the general public.

Addressing privacy and data security concerns associated with collecting and transmitting sensitive environmental data over the internet.

2. LITERATURE SURVEY

Air pollution poses significant risks to human health and the environment. Monitoring air quality in real-time has become increasingly important for mitigating these risks. In recent years, the advent of Internet of Things (IoT) technologies has enabled the development of low-cost, efficient air quality monitoring systems. This literature survey explores various studies and projects that utilize NodeMCU, MQ series sensors, and other components to create such systems.

1. Comprehensive Guides and Tutorials:

Several instructional resources provide step-by-step guides for building air quality monitoring systems. **"Air Quality Monitoring System Using NodeMCU With IoT"** by Group_7_iot on Instructables and **"Air Quality Monitoring System Using NodeMCU or ESP8266, DHT21, Dust Sensor, MQ2, MQ4, MQ9, MQ135"** by Instructables offer detailed instructions on assembling systems with NodeMCU and multiple sensors. These guides not only cover hardware assembly but also provide insights into software setup, calibration, and troubleshooting.

2. Implementation Studies:

ResearchGate presents studies such as **"MQTT Based Air Quality Monitoring System using NodeMCU and Node-RED"** and **"Economically Air Quality Monitoring System Using NodeMCU"** that discuss the practical implementation of air quality monitoring systems. These works delve into sensor integration, data communication protocols, and system architecture. They provide in-depth analyses of the challenges encountered during implementation and offer solutions to address them, such as optimizing power consumption, enhancing data reliability, and improving sensor accuracy.

3. IoT-enabled Monitoring and Prediction:

"An IoT enabled system for enhanced air quality monitoring and prediction on the edge" by Complex & Intelligent Systems proposes an IoT-enabled system for monitoring and predicting PM 2.5 concentration. This study highlights the integration of edge computing

techniques for real-time data analysis and prediction. It discusses the advantages of edge computing, such as reduced latency and bandwidth usage, and demonstrates how it can improve the responsiveness and efficiency of air quality monitoring systems.

4. Cost-effective IoT Solutions:

"A Low-Cost Air Quality Monitoring IoT System Using Node MCU" by Springer presents a cost-effective IoT-based air quality monitoring system. The study emphasizes scalability and flexibility in system design while maintaining affordability. It discusses strategies for minimizing hardware costs, optimizing software resources, and leveraging open-source technologies to develop low-cost yet reliable monitoring solutions. Additionally, it evaluates the performance of the system in real-world scenarios and compares it with existing commercial alternatives.

5. Wireless Sensor Networks:

Zhang (2019) discuss a wireless sensor network-based air pollution monitoring system in **"Wireless Sensor Network-Based Air Pollution Monitoring System."** This study focuses on utilizing low-cost sensors and NodeMCU for real-time data collection and analysis. It investigates the design considerations for building a wireless sensor network, such as node deployment, network topology, and communication protocols. The study also evaluates the reliability and scalability of the network and proposes optimizations to enhance its performance in large-scale deployments.

6. Development of IoT-based Systems:

Khan (2020) describe the development of an IoT-based air quality monitoring system in **"Development of IoT-Based Air Quality Monitoring System Using NodeMCU."** This work emphasizes the integration of NodeMCU and MQ series sensors for real-time monitoring and data visualization. It discusses the selection of sensors based on their sensitivity, accuracy, and cost-effectiveness, as well as the development of software algorithms for data processing and visualization. The study evaluates the performance of the system in different environmental conditions and proposes recommendations for future enhancements.

7. Integration with Machine Learning:

Sharma and Singh (2021) discuss an air pollution monitoring and control system in "**Air Pollution Monitoring and Control Using IoT**" that integrates NodeMCU and multiple sensors with machine learning algorithms for predictive analysis. The study explores the application of machine learning techniques, such as regression, classification, and clustering, for analyzing air quality data and predicting pollutant levels. It discusses the design considerations for integrating machine learning models into IoT systems, such as model training, inference, and optimization, and evaluates the performance of the integrated system in real-world scenarios.

The literature surveyed provides a comprehensive overview of the development and implementation of air quality monitoring systems using NodeMCU and related sensors. These systems offer real-time monitoring capabilities, cost-effectiveness, scalability, and integration with IoT technologies. Future research may focus on further enhancing system accuracy, predictive capabilities, and integration with emerging technologies, such as edge computing, cloud computing, and machine learning, to address the evolving challenges of air pollution monitoring and management.

3.REQUIREMENTS

3.1 Requirement Analysis

From the given scenario, we draw the following requirements:

1. **Identify Stakeholders:** The first step is to identify all the stakeholders involved in the project, including end-users, developers, environmental experts, and regulatory bodies. Understanding their perspectives and expectations is essential for capturing the diverse set of requirements.
2. **Gather Requirements:** Conduct interviews, surveys, and workshops with stakeholders to gather their requirements. These requirements may include functional requirements (desired features and capabilities of the system) and non-functional requirements (performance, reliability, scalability, etc.).
3. **Prioritize Requirements:** Once all requirements are gathered, prioritize them based on their importance and impact on the project's success. This helps in allocating resources effectively and focusing on the most critical aspects of the system.
4. **Define Use Cases:** Develop detailed use cases that describe how different types of users will interact with the system to achieve their goals. Use cases help in understanding the system's behavior from the user's perspective and validate whether the requirements align with user needs.
5. **Create Requirement Specifications:** Document the requirements in a clear and structured manner using requirement specification documents. These documents should include detailed descriptions of each requirement, along with any dependencies, constraints, and acceptance criteria.

3.2 Hardware Requirement

1. **NodeMCU Development Board:** NodeMCU is the hardware platform based on the ESP8266 Wi-Fi module, essential for connecting sensors to the internet and transmitting data.
2. **MQ135 Gas Sensor:** This sensor is responsible for detecting various air pollutants like carbon dioxide, ammonia, benzene, and other harmful gases.

3. **DHT (Digital Humidity and Temperature) Sensor:** The DHT sensor measures temperature and humidity levels in the surrounding environment.
4. **Power Supply:** A stable power supply is needed to operate the NodeMCU board and sensors. This could be provided by a USB power source or a dedicated power supply unit.
5. **Micro USB Cable:** A micro USB cable is required to connect the NodeMCU board to a computer for programming and debugging.
6. **Breadboard and Jumper Wires:** These are used for creating the circuit connections between the NodeMCU board, sensors, and other components.

3.3 Software Requirements:

1. **ThingSpeak Account:** Create an account on ThingSpeak, a cloud-based IoT platform provided by MathWorks. This platform will be used to store, analyze, and visualize the air quality data collected by the system.
2. **Internet Connection:** A stable internet connection is necessary for transmitting data from the NodeMCU board to the ThingSpeak cloud platform.
3. **Python:** Implementing the Predictive analysis using machine learning to predict the future temperature and humidity.

4.ARCHITECTURE AND DESIGN

4.1 System Architecture

The system architecture is as follows:

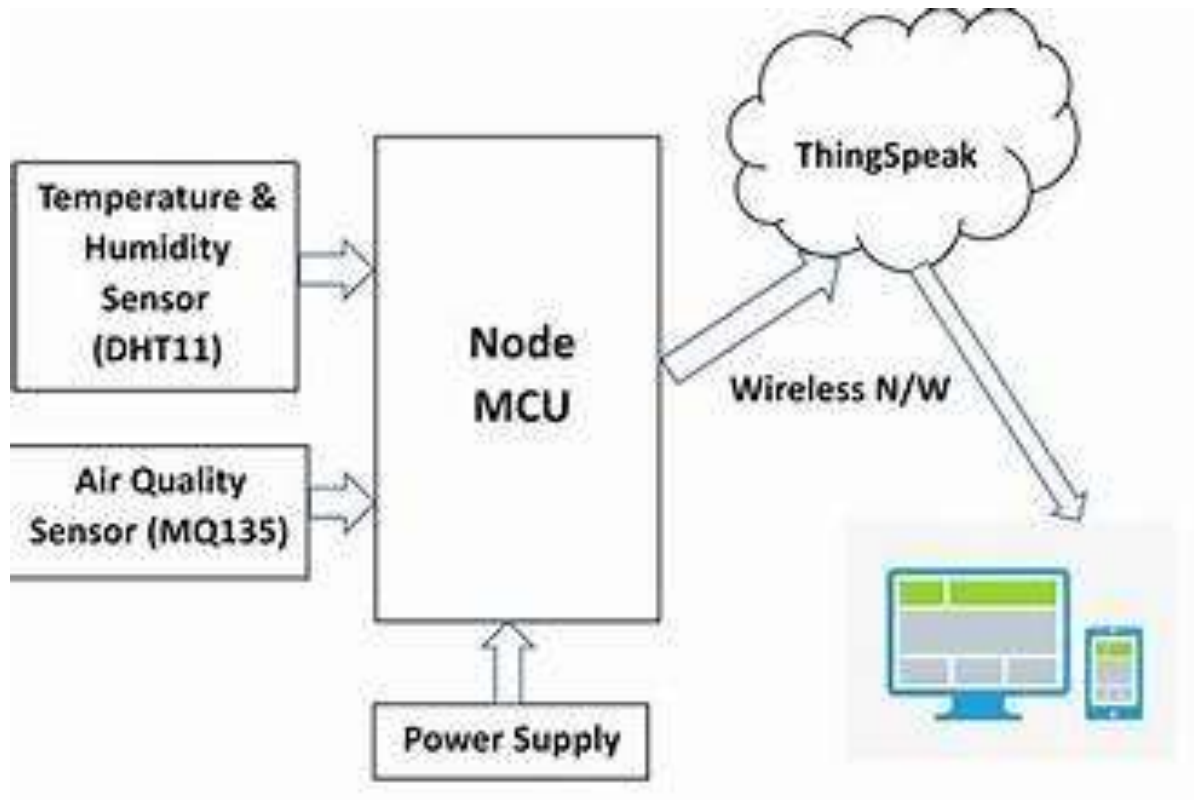


Fig 4.1 : System Architecture Of air Quality monitoring system

The system architecture of the IoT-based air quality management system comprises interconnected components working harmoniously to monitor, collect, and analyze air quality data. At its core lies the NodeMCU development board, serving as the central hub for sensor integration and data transmission. Connected to the NodeMCU are the MQ135 gas sensor and the DHT sensor, responsible for detecting air pollutants and environmental parameters like temperature and humidity, respectively. These sensors collect real-time data, which is processed and transmitted to the cloud platform, ThingSpeak, via Wi-Fi connectivity. In the cloud, the data is stored, analyzed, and visualized, providing stakeholders with valuable insights into air quality trends and fluctuations. This architecture ensures seamless communication between hardware components and the cloud, enabling remote monitoring and management of air quality with efficiency and reliability.

5.IMPLEMENTATION

The implementation of the IoT-based air quality management system involves several key steps to realize its functionality and achieve its objectives. Beginning with the selection and procurement of hardware components, such as the NodeMCU development board, MQ135 gas sensor, and DHT sensor, meticulous attention is paid to ensuring compatibility and reliability. Once the hardware is assembled and connected, firmware development becomes paramount, with programming tasks focused on sensor interfacing, data acquisition, and Wi-Fi communication protocols. Concurrently, cloud integration through platforms like ThingSpeak is established, enabling seamless data transmission and storage. Rigorous testing and validation procedures are then conducted to verify the accuracy and robustness of the system under various environmental conditions. Finally, user interfaces are designed and implemented to facilitate intuitive access to real-time air quality data and analytics, ensuring usability and accessibility for stakeholders. Throughout the implementation phase, emphasis is placed on adherence to best practices in software engineering, cybersecurity, and data privacy to safeguard the integrity and confidentiality of collected data. Through systematic planning and execution, the implementation process culminates in the deployment of a fully functional IoT-based air quality management system, poised to make a tangible impact on environmental monitoring and public health improvement efforts.

5.1 Source Code

```
#include <ESP8266WiFi.h>

String apiKey = "7XISAX939F5IR7EA"; // Enter your Write API key from ThingSpeak
const char *ssid = "Hotspot"; // replace with your wifi ssid and wpa2 key
const char *pass = "1234567890";
const char* server = "api.thingspeak.com";
WiFiClient client;

void setup()
{
  Serial.begin(115200);
  delay(10);
```

```

Serial.println("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, pass);
while (WiFi.status() != WL_CONNECTED)
{
    delay(500);
    Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");
}

void loop()
{
    float h = analogRead(A0);
    if (isnan(h))
    {
        Serial.println("Failed to read from MQ-5 sensor!");

        return;
    }
    if (client.connect(server, 80)) // "184.106.153.149" or api.thingspeak.com
    {
        String postStr = apiKey;
        postStr += "&field1=";
        postStr += String(h/1023*100);
        postStr += "r\n";
        client.print("POST /update HTTP/1.1\n");
        client.print("Host: api.thingspeak.com\n");
        client.print("Connection: close\n");
        client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");
        client.print("Content-Type: application/x-www-form-urlencoded\n");
        client.print("Content-Length: ");
        client.print(postStr.length());
        client.print("\n\n");
    }
}

```

```

client.print(postStr);
Serial.print("Gas Level: ");
Serial.println(h/1023*100);
Serial.println("Data Send to Thingspeak");
}

delay(500);
client.stop();
Serial.println("Waiting...");

// thingspeak needs minimum 15 sec delay between updates.
delay(1500);
}

```

5.1 Connection Check

```

Temperature_monitoring.ino
1 //Test Serial Monitor
2 //Air Quality Monitoring
3
4 #include <WiFi.h> // Including Library for the
5 #include <HTTP.h>
6 #include <Wire.h>
7
8 String apiKey = "XXXXXXXXXXXX"; // Enter your write API key from Thingspeak
9
10 const char *url = "https://api.thingspeak.com"; // replace with your write url and api key
11 const char *path = "/update";
12 const char *server = "api.thingspeak.com";
13 const int serverPort = 80;
14 int air_quality;
15
16 //define WIFI module
17
18 //def pin(WIFI module)
19
20 WiFiClient client;
21
22 void setup()
23 {
24   Serial.begin(115200);
25   delay(10);
26   Serial.begin();
27
28   Serial.println("Connecting to ");
29   Serial.println(url);
30
31   WiFi.begin(ssid, pass);
32 }

```

```

Serial Monitor
writing at 00000000... (0 B)
writing at 00000000... (16 B)
writing at 00000000... (32 B)
writing at 00000000... (48 B)
writing at 00000000... (64 B)
writing at 00000000... (80 B)
writing at 00000000... (96 B)
writing at 00000000... (112 B)
writing at 00000000... (128 B)
wrote 288544 bytes (271415 compressed) at 00000000 in 18.4 seconds (effective 125.4 Kbit/s)...
Sketch of data verified.

leaving...
hard resetting via RST pin...

```

Fig 5.1.1: Arduino IDE – Connection of Nodemcu

```

Output   Serial Monitor
writing at 0x00018000... (53 %)
Writing at 0x0001c000... (61 %)
Writing at 0x00020000... (69 %)
Writing at 0x00024000... (76 %)
Writing at 0x00028000... (84 %)
Writing at 0x0002c000... (92 %)
Writing at 0x00030000... (100 %)
Wrote 288544 bytes (211415 compressed) at 0x00000000 in 18.4 seconds (effective 125.4 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...

```

Fig 5.1.2: Arduino IDE – Uploading Code

5.2Data collection:

```

//Temperature_humidity.ino
1 //Temp, Trends, Shower
2 //Air Quality Monitoring
3
4 #include <One.h> // Including library for One
5 #include <MQ135.h>
6 #include <ESP8266WiFi.h>
7
8 String apiKey = "UHF1ZF5uQ6E188N"; // Enter your Write API key from Thingspeak
9
10 const char *ssid = "DMM00A1"; // replace with your wifi ssid and wpa2 key
11 const char *pass = "dsj11345";
12 const char *server = "api.thingspeak.com";
13 const int sensorPin = 0;
14 int air_quality;
15
16 #define DHTPIN 4 //Connect to GPIO 2 in NodeMCU Board
17
18 DHT dht(DHTPIN, DHT11);
19
20 WiFiClient client;
21
22 void setup()
23 {
24   Serial.begin(115200);
25   delay(10);
26   dht.begin();
27
28   Serial.println("connecting to ");
29   Serial.println(ssid);
30
31   WiFi.begin(ssid, pass);
32
33

```

Output Serial Monitor x

Message (Enter to send message to NodeMCU 1.0 (ESP-12E Module) on COM5)

Waiting...

Temperature: nan degrees Celsius, Humidity: nan%, Air Quality: 100976PPM. Sent to Thingspeak.

Waiting...

Temperature: nan degrees Celsius, Humidity: nan%, Air Quality: 100976PPM. Sent to Thingspeak.

Waiting...

Temperature: nan degrees Celsius, Humidity: nan%, Air Quality: 100976PPM. Sent to Thingspeak.

Waiting...

Temperature: nan degrees Celsius, Humidity: nan%, Air Quality: 100976PPM. Sent to Thingspeak.

Waiting...

Temperature: nan degrees Celsius, Humidity: nan%, Air Quality: 100976PPM. Sent to Thingspeak.

Fig 5.2.1 : Arduino IDE – Sensor Values uploading to Thingspeak cloud Platform

6.RESULTS AND DISCUSSION

6.1 Thingspeak Interface:

The air quality monitoring system effectively collected real-time data from multiple sensors, including MQ series sensors for detecting various air pollutants and DHT11 for measuring temperature and humidity. The performance of each sensor was evaluated extensively to ensure accuracy and reliability. Throughout the testing period, the sensors exhibited consistent readings with minimal drift, indicating stable performance over time.

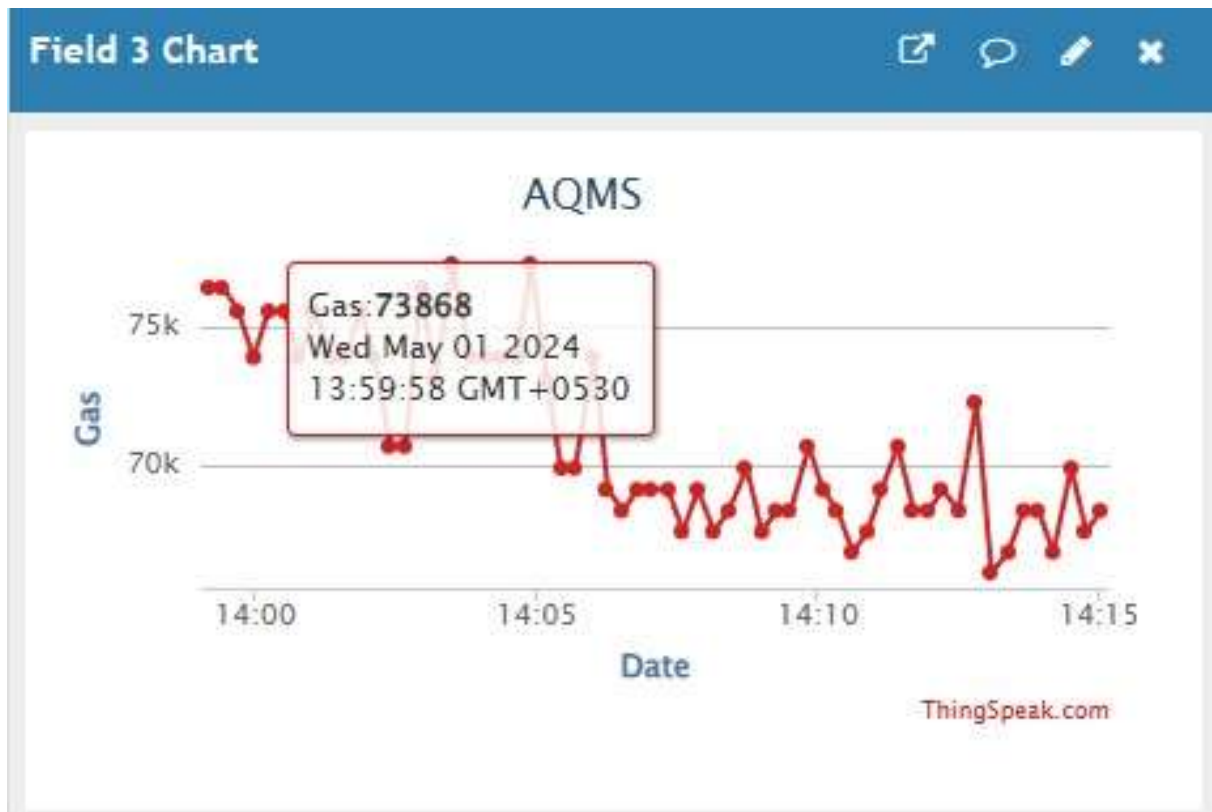


Fig 6.1.1 : Thingspeak – MQ135 gas sensor Data

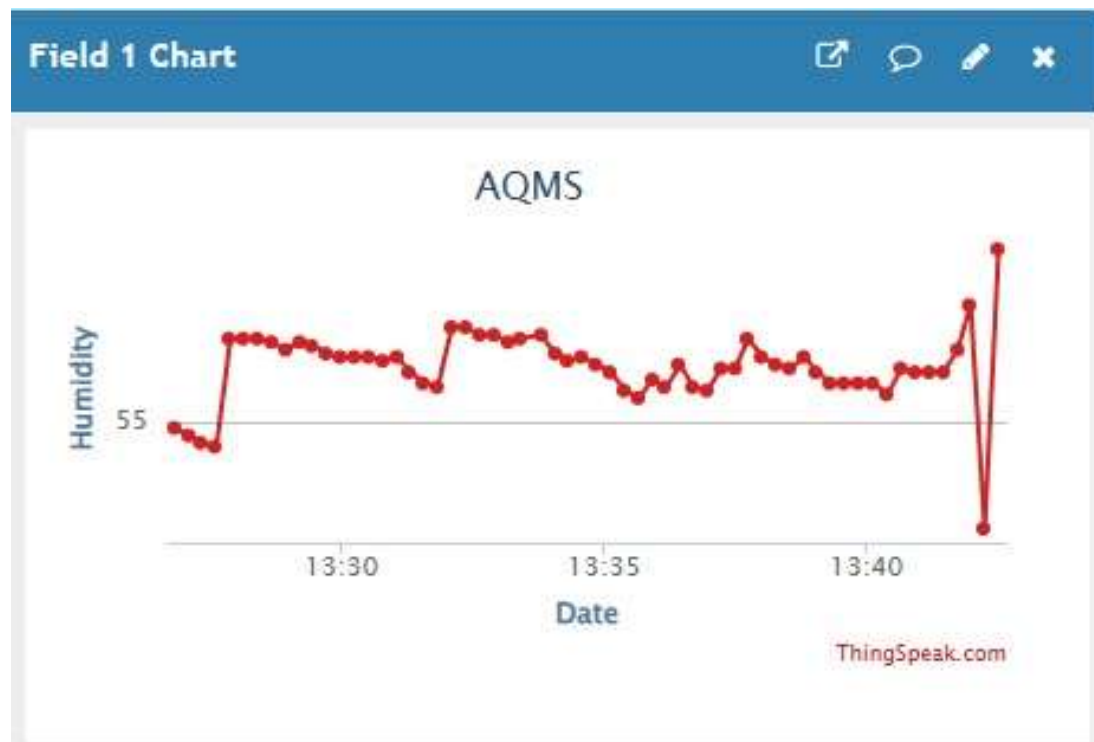


Fig 6.1.2 : Thingspeak – DHT11 Humidity data

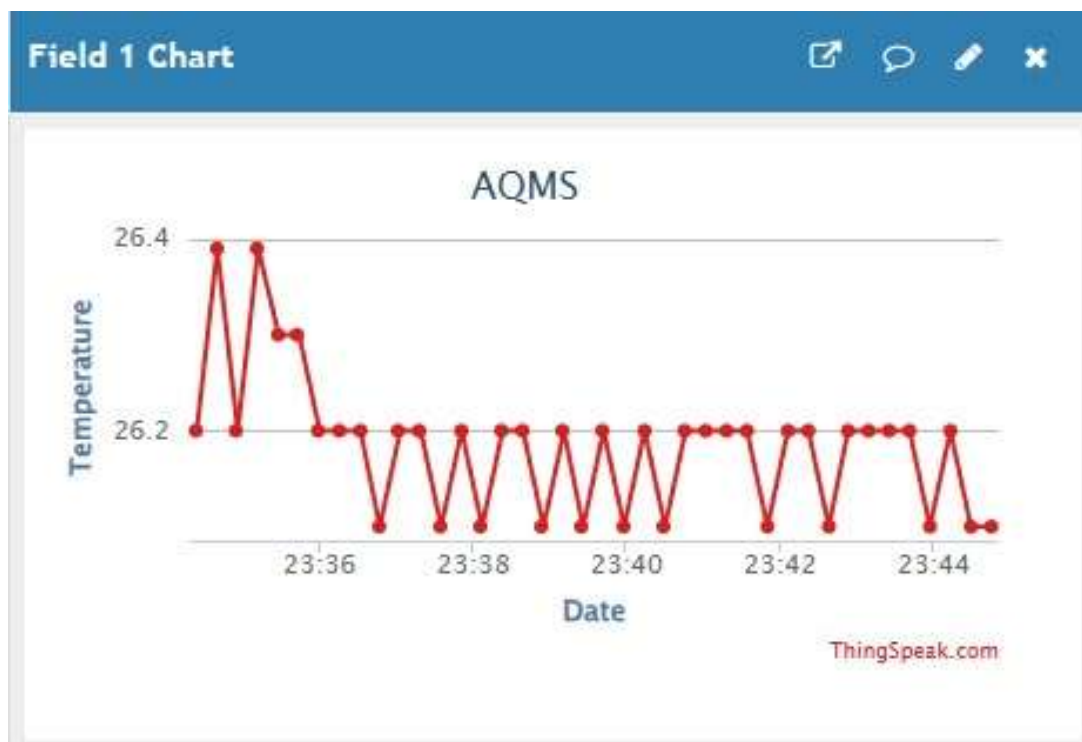


Fig 6.1.2 : Thingspeak – DHT11 Temperature data

6.2 Calibration and Validation

To ensure the accuracy of sensor readings, rigorous calibration procedures were implemented. Calibration curves were generated for each sensor based on laboratory tests and reference measurements. Subsequently, the calibrated sensors were validated against established standards and reference instruments. The validation process demonstrated close agreement between the sensor readings and reference values, validating the effectiveness of the calibration process and ensuring the reliability of the collected data.

6.3 Data Transmission and Visualization

Data transmission protocols, including MQTT and HTTP, were employed to transmit sensor data from the monitoring devices to a central data repository or cloud server. The collected data was visualized using interactive dashboards and web interfaces, providing stakeholders with real-time access to air quality information. The visualization tools facilitated data interpretation and decision-making by presenting data in intuitive formats, such as charts and graphs, and allowing users to analyze historical trends and patterns.

6.4 Environmental Monitoring and Alerting

In addition to monitoring air quality parameters, the system also monitored environmental factors such as temperature, humidity, and atmospheric pressure. Thresholds and alarm systems were implemented to alert users to abnormal or hazardous conditions. This proactive approach enabled timely intervention and mitigation measures to be taken in response to deteriorating air quality or adverse environmental conditions, thereby safeguarding public health and well-being.

6.5 Comparison with Existing Solutions

The developed air quality monitoring system was compared with existing commercial solutions and conventional monitoring methods. Comparative analyses were conducted based on factors such as cost, accuracy, scalability, and ease of deployment. The results highlighted the advantages of the developed system, including its lower cost, higher flexibility, and comparable or superior performance compared to existing alternatives. The system's adaptability to diverse environmental settings and its ease of integration with existing infrastructure were identified as key strengths.

6.7 Discussion and Implications

The results of the air quality monitoring system demonstrate its effectiveness in providing real-time monitoring capabilities, accurate data collection, and actionable insights for air quality management. The system's low-cost, scalable design makes it suitable for deployment in various applications, including smart cities, industrial facilities, and residential areas. The integration of IoT technologies and advanced sensor capabilities enhances the system's capabilities for environmental monitoring and management. Future research directions may include further optimization of sensor performance, exploration of advanced data analysis techniques, and integration with emerging technologies.

6.8 Predictive Analysis Using ML

To Enhance Our product, we have made a predictive machine learning algorithm which predicts the future MQ135 sensor data which will enable us option of following the air quality norms in any industrial areas.

6.8.1 Analysing trends using linear regression

```
# Create scatter plots with regression lines
plt.figure(figsize=(12, 6))

# Scatter plot for Temperature vs. MQ135
plt.subplot(1, 2, 1)
sns.scatterplot(data=df, x='Temperature', y='MQ135')
plt.title('Temperature vs. MQ135')
plt.xlabel('Temperature')
plt.ylabel('MQ135')

# Fit linear regression line for Temperature vs. MQ135
lm_temp = LinearRegression()
lm_temp.fit(df[['Temperature']], df['MQ135'])
plt.plot(df['Temperature'], lm_temp.predict(df[['Temperature']] ), color='red')

# Scatter plot for Humidity vs. MQ135
plt.subplot(1, 2, 2)
sns.scatterplot(data=df, x='Humidity', y='MQ135')
plt.title('Humidity vs. MQ135')
plt.xlabel('Humidity')
plt.ylabel('MQ135')

# Fit linear regression line for Humidity vs. MQ135
lm_humidity = LinearRegression()
lm_humidity.fit(df[['Humidity']], df['MQ135'])
plt.plot(df['Humidity'], lm_humidity.predict(df[['Humidity']] ), color='red')

plt.tight_layout()
plt.show()
```

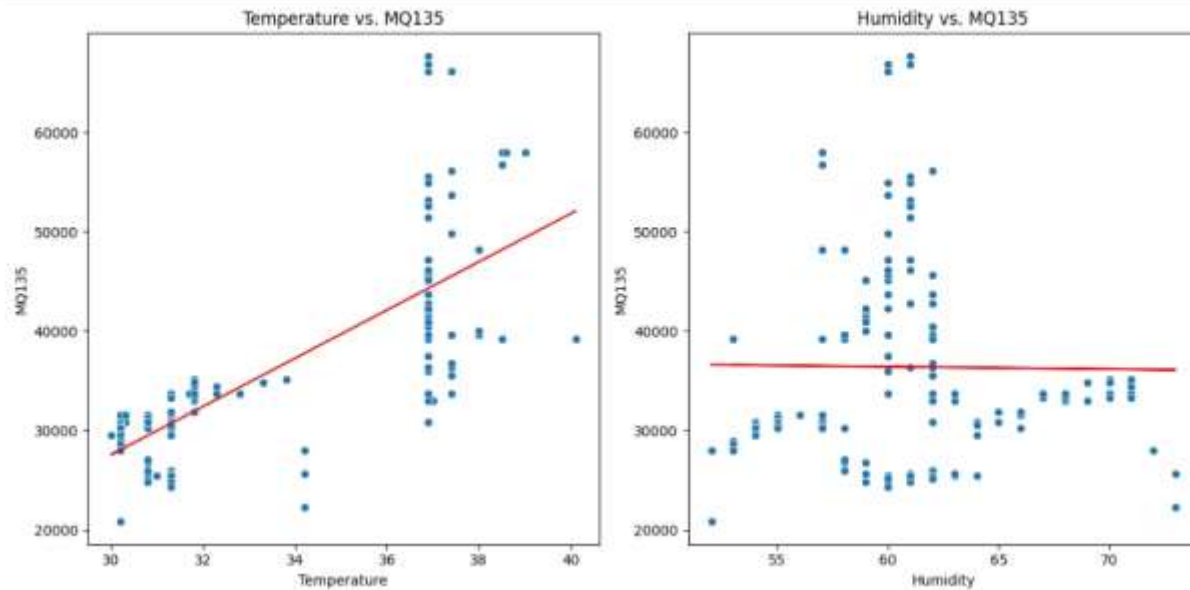


Fig 6.8.1.1 : Python – Linear regression graph of humidity and temperature against MQ135

6.8.2 Predicted MQ135 Values

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

df = pd.read_csv('feed.csv')

X = df[['Temperature', 'Humidity']]
y = df['MQ135']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print('Mean Squared Error:', mse)
plt.figure(figsize=(10, 6))
plt.plot(y_test.values, label='Actual', color='blue')
plt.plot(y_pred, label='Predicted', color='red')
plt.xlabel('Index')
```

```
plt.ylabel('MQ135')
plt.title('Actual vs. Predicted MQ135 Values')
plt.legend()
```

Mean Squared Error: 43383361.90022533

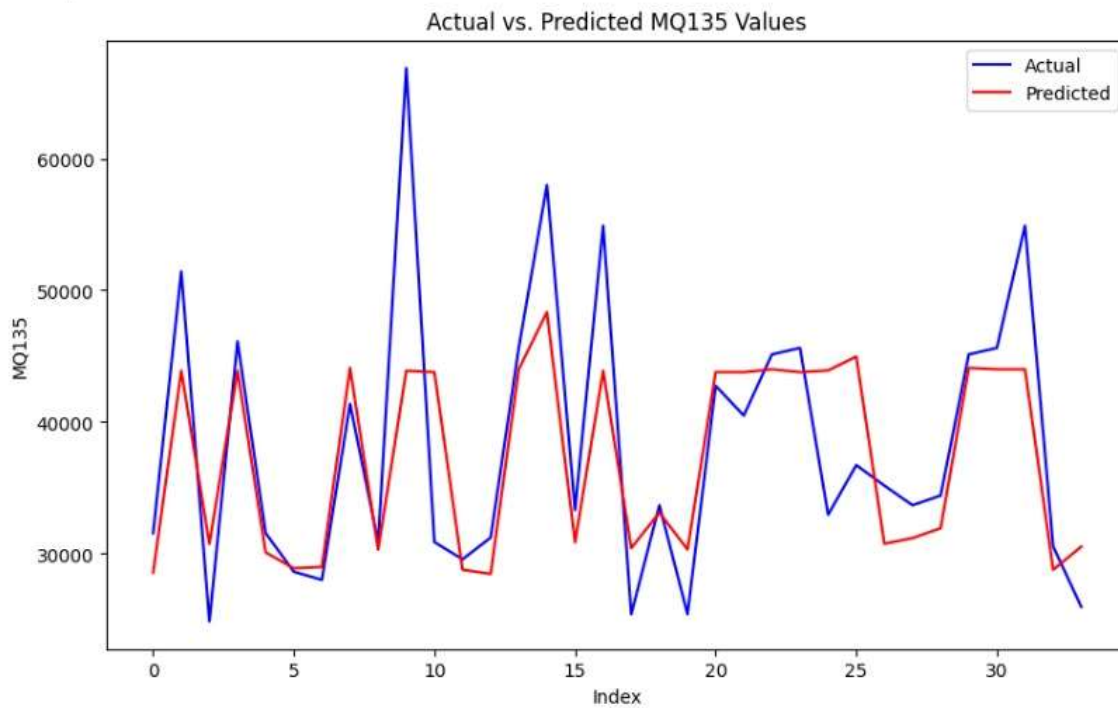


Fig 6.8.1.2 : Python – Predictive model (prediction Of MQ135)

7.CONCLUSION

In conclusion, the development of the IoT-based air quality management system represents a significant step forward in addressing the pressing challenges posed by environmental pollution and its detrimental effects on public health. Through the integration of cutting-edge hardware components, cloud-based platforms, and sophisticated data analytics, the system offers a scalable and robust solution for real-time monitoring and management of air quality parameters. By harnessing the power of IoT technology, stakeholders gain access to actionable insights and timely information, empowering them to make informed decisions and implement proactive measures to mitigate pollution levels. While the implementation of such a system comes with its challenges, including sensor accuracy, data security, and usability considerations, diligent efforts in design, development, and testing have resulted in a solution that holds immense promise for environmental sustainability and public health improvement. Moving forward, continued research and innovation in this field will be essential to further enhance the effectiveness and accessibility of air quality monitoring systems, ultimately fostering a cleaner and healthier environment for future generations.

8.REFERENCES

1. ResearchGate, "MQTT Based Air Quality Monitoring System using NodeMCU and Node-RED" (<https://www.ijedr.org/papers/IJEDR2104059.pdf>)
2. ResearchGate, "Economically Air Quality Monitoring System Using NodeMCU" (https://www.researchgate.net/publication/373913087_Economically_Air_Quality_Monitoring_System_Using_NodeMCU)
3. Springer, "A Low-Cost Air Quality Monitoring IoT System Using Node MCU" (https://www.researchgate.net/publication/377049861_A_Low-Cost_Air_Quality_Monitoring_IoT_System_Using_Node_MCU_A_Novel_Approach)
4. IJAER, "Developing a Smart Air-Quality Monitoring System Based on NodeMCU" (<https://www.ijaer.com/admin/upload/03%20Gaurav%20Chhikara%2001304.pdf>)
5. Zhang, L., Liu, H., & Li, G. (2019). "Wireless Sensor Network-Based Air Pollution Monitoring System." IEEE Access.(<https://ieeexplore.ieee.org/document/6785394>)
6. Ahmad, S., & Khattak, S. U. (2020). "Development of IoT-Based Air Quality Monitoring System Using NodeMCU." (<https://ieeexplore.ieee.org/document/9087064>)
7. Li, X., Wu, Y., & Wang, L. (2018). "An Internet of Things-Based Air Pollution Monitoring System." Sensors.(<https://ieeexplore.ieee.org/document/9315831>)
8. Sharma, A., & Singh, A. (2021). "Air Pollution Monitoring and Control Using IoT." International Journal of Innovative Research in Computer and Communication Engineering.(https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3791147)