BREATHESAFE – SMART INDOOR AIR POLLUTION TRACKER

A project report submitted
to

Dr. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY LUCKNOW

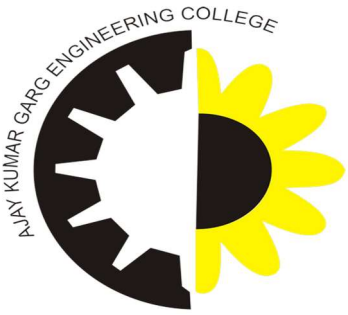
For partial Fulfillment of the Requirement for the Award of the Degree

of

MASTER OF COMPUTER APPLICATIONS

by

APURAV SINGH	2300270140016
HARSH SAHAY	2300270140035
AJEET YADAV	2300270140010
HARSH RAGHAV	2300270140034



DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS
AJAY KUMAR GARG ENGINEERING COLLEGE
GHAZIABAD

2024-25

BREATHESAFE – SMART INDOOR AIR POLLUTION TRACKER

A project report submitted
to

Dr. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY
LUCKNOW

For partial Fulfillment of the Requirement for the Award of the Degree

of

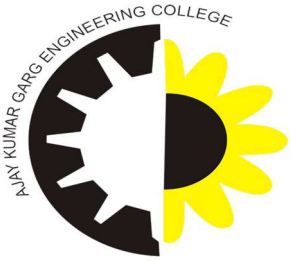
MASTER OF COMPUTER APPLICATIONS

by

APURAV SINGH	**2300270140016**
HARSH SAHAY	**2300270140035**
AJEET YADAV	**2300270140010**
HARSH RAGHAV	**2300270140034**

Under the guidance of

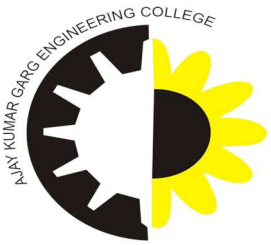
Asst. prof : ARPNA SAXENA



DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS
AJAY KUMAR GARG ENGINEERING COLLEGE
GHAZIABAD

2024-25

COLLEGE CERTIFICATE



This is to certify that project report entitled **BREATHESAFE – SMART INDOOR AIR POLLUTION TRACKER** which is submitted by **Apurav Singh (2300270140016), Harsh Sahay (2300270140035), Ajeet Yadav (2300270140010) , Harsh Raghav (2300270140016)** in partial fulfilment of the requirement for the award of degree Master of Computer Applications of Dr. A.P.J. Abdul Kalam Technical University, Uttar Pradesh (AKTU), is a record of the candidates work carried out by them under my supervision. The matter embodied in this report is original and has not been submitted for the award of any other degree.

Date :

Guide Name : Ms. Arpna Saxena

Certificate of Compliance with United Nations Sustainable Development Goals

This is to certify that the project titled **BREATHESAFE – SMART INDOOR AIR POLLUTION TRACKER** submitted by **Apurav Singh, Harsh Sahay , Ajeet Yadav, Harsh Raghav** final year students of the Master of Computer Applications program at Ajay Kumar Garg Engineering College, Ghaziabad, have been reviewed and found to be in alignment with the following United Nations Sustainable Development Goals (SDGs). Details regarding the justification of the same are provided in Chapter 8 (Conclusion). All efforts have been made to the best of our ability and knowledge that no other SDGs are compromised or negatively impacted.

SDG No.	SDG Name	Relevance	SDG No.	SDG Name	Relevance
1	No Poverty	☐	**10**	Reduced Inequalities	☐
2	Zero Hunger	☐	**11**	Sustainable Cities and Communities	☐
3	Good Health and Well-being	☐	**12**	Responsible Consumption and Production	☐
4	Quality Education	☐	**13**	Climate Action	☐
5	Gender Equality	☐	**14**	Life Below Water	☐
6	Clean Water and Sanitation	☐	**15**	Life on Land	☐
7	Affordable and Clean Energy	☐	**16**	Peace, Justice, and Strong Institutions	☐
8	Decent Work and Economic Growth	☐	**17**	Partnerships for the Goals	☐
9	Industry, Innovation, and Infrastructure	☐			

Signature of the Students

Apurav Singh - 2300270140016
Harsh Sahay - 2300270140035
Ajeet Yadav - 2300270140010
Harsh Raghav- 2300270140034

Name of the supervisor

Ms. Arpna Saxena

ACKNOWLEDGEMENT

With a profound sense of gratitude and deep respect, we express my heartfelt thanks to my guide and mentor **Ms. Arpna Saxena**, Assistant Professor, MCA Department, for her valuable guidance, constant encouragement, and confidence she instilled in me throughout the development of this major project "**BREATHESAFE – SMART INDOOR AIR POLLUTION TRACKER**". Her thorough understanding of the subject and professional guidance was indeed of immense help during all phases of the project.

We would like to extend my sincere gratitude to **Ms. Saroj Bala**, Head of the Department (HOD) of MCA, for providing the required facilities, academic support, and an inspiring environment to complete the project successfully.

We are also deeply thankful to all the **faculty members** of the MCA Department, Ajay Kumar Garg Engineering College, for their cooperation, timely advice, and valuable time during the course of this project.

Finally, we would like to sincerely thank my parents, teachers, and friends whose continuous encouragement, valuable suggestions, and moral support have been instrumental throughout the project journey.

Name of the Students :

APURAV SINGH (2300270140016)

HARSH SAHAY (2300270140035)

AJEET YADAV (2300270140010)

HARSH RAGHAV (2300270140034)

LIST OF SYMBOLS, ABBREVIATIONS AND NOMENCLATURE

Abbreviation	Full Form
IoT	Internet of Things
AQI	Air Quality Index
PM2.5	Particulate Matter with diameter ≤ 2.5 µm
PM10	Particulate Matter with diameter ≤ 10 µm
MQ135	Air Quality Sensor for NH_3, NO_x, alcohol, benzene, smoke, CO_2
MQ7	Carbon Monoxide (CO) Gas Sensor
OLED	Organic Light Emitting Diode
BME680	Bosch Sensor for Temperature, Humidity, Pressure, and Gas
PMS7003	Particulate Matter Sensor (PM2.5/PM10) by Plantower
ESP8266	Wi-Fi Enabled Microcontroller
HTTP	Hypertext Transfer Protocol
URL	Uniform Resource Locator
hPa	Hectopascal (unit of pressure)
ppm	Parts Per Million
µg/m³	Micrograms per Cubic Meter
GUI	Graphical User Interface
Wi-Fi	Wireless Fidelity
API	Application Programming Interface
DHT	Digital Humidity and Temperature (if applicable to backup sensors)
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
DB	Database
UI	User Interface
TX/RX	Transmit/Receive (Serial Communication)
ADC	Analog-to-Digital Converter

LIST OF TABLES

LIST OF FIGURES

Figure Number & Name **Page No.**

ABSTRACT

Air pollution poses a serious threat to human health, contributing to respiratory illnesses, lung infections, and degraded indoor air quality. Indoor air pollution is often more harmful than outdoor pollution due to limited ventilation and prolonged exposure. With people spending increased time indoors, especially post-pandemic, continuous air quality monitoring has become crucial for ensuring a safe living environment. This research introduces a real-time IoT-based indoor air quality monitoring system utilizing NodeMCU ESP8266, BME680, PMS7003, and MQ135 sensors. The system captures essential environmental parameters such as temperature, humidity, pressure, particulate matter (PM2.5 and PM10), volatile organic compounds (VOCs), and carbon dioxide (CO2), among other harmful gases. The collected data is displayed on a 0.96-inch OLED screen and transmitted wirelessly to a cloud server, enabling real-time monitoring via a web-based dashboard. Additionally, the system triggers alerts when pollutant levels surpass safe thresholds, allowing users to take preventive actions. With its cost-effective and scalable design, this solution enhances indoor air quality monitoring and promotes healthier living conditions.

INDEX

CHAPTER – 1

INTRODUCTION

1.1 Background

Indoor air pollution poses serious health risks, yet many environments lack real-time monitoring solutions. Traditional systems are often costly and inaccessible for everyday users. With advancements in IoT and sensor technology, it is now possible to create compact and affordable systems for environmental monitoring. The “Smart Indoor Air Quality Monitoring System” project uses NodeMCU ESP8266 and multiple sensors to measure air quality parameters in real time. The data is displayed locally and sent to a web interface, helping users stay informed and make healthier lifestyle choices.

1.2 Objective of Project

The primary objectives of the Smart Indoor Air Quality Monitoring System are as follows:

• To design a real-time air quality monitoring system using NodeMCU ESP8266 and multiple low-cost environmental sensors.
• To measure key air quality parameters such as temperature, humidity, pressure, particulate matter (PM2.5 and PM10), and harmful gases (CO, VOCs).
• To display collected data on an OLED screen, Serial Monitor, and a web-based dashboard via Wi-Fi.
• To provide users with accurate AQI levels and health-related air quality insights, promoting informed decisions for safer indoor environments.

1.3 Scope of Project

The project encompasses:

• A real-time indoor air quality monitoring system built on NodeMCU ESP8266 with sensors including BME680, PMS7003, MQ135, and MQ-7.
• A web-based dashboard that displays live environmental data such as temperature, humidity, pressure, PM2.5, PM10, CO, and VOC levels.
• Backend functionality using a PHP or Flask-based server connected to a MySQL database to store sensor data and user logs.
• Integration with an OLED display and Serial Monitor for local visualization of AQI and sensor readings.
• AQI calculation logic based on sensor data with categorized health levels and alert features for poor air quality.

1.4 Problem Statement

Air pollution poses serious health risks, yet real-time air quality data remains inaccessible in many areas, especially schools. Students are often exposed to harmful pollutants due to the absence of monitoring systems in classrooms. Existing solutions are either costly, limited in coverage, or lack user-friendly interfaces. There is a pressing need for a low-cost, accurate, and scalable air quality monitoring system that provides real-time data, raises awareness, and supports safer, healthier indoor environments in educational institutions.

Key challenges:
(a) Limited geographical coverage and real-time monitoring in current systems.
(b) High cost and maintenance complexity of monitoring stations.
(c) Lack of user-friendly interfaces for data access and interpretation.
(d) Limited integration of data for predictive modelling and pollution mitigation strategies.

1.5 Literature Review

A number of research articles and projects have explored virtual try-on systems. Key works include:

1.5.1. IoT-Based Air Pollution Monitoring and Alert System [4]

This study proposes an Internet of Things (IoT)-enabled air monitoring system that continuously measures air pollutants such as CO_2, CO, PM2.5, and volatile organic compounds (VOCs). It uses sensors like the MQ135 and PMS5003 and transmits data to a centralized platform through a Wi-Fi module (ESP8266). The real-time data is visualized on an LCD or web dashboard, enabling users to monitor the AQI remotely. One of the notable contributions of this study is the inclusion of real-time alerting mechanisms, which notify users when pollutant levels cross predefined safety thresholds. This study supports the importance of deploying low-cost sensor networks in sensitive environments like schools and hospitals to increase awareness and prevent health hazards.

1.5.2. Low-Cost Air Quality Monitoring Using Arduino and GSM [6]

This paper emphasizes cost-effectiveness and portability, presenting an air quality monitoring system that uses MQ-series gas sensors, DHT11 for humidity and temperature, and a GSM module for data transmission. The system logs environmental data locally while sending air quality status through SMS to subscribed users. The solution is particularly valuable in remote areas lacking stable internet connectivity. Although the system lacks high-resolution cloud integration, it is effective in promoting localized air pollution awareness at a minimal cost. Its modular design makes it highly adaptable for school environments where low budget and simplicity are key requirements.

1.5.3. Real-Time Air Quality Monitoring System Using IoT and Cloud Integration [2]

This research focuses on real-time environmental monitoring with cloud storage and visualization. Utilizing a NodeMCU ESP8266 microcontroller, sensors like BME680 and PMS7003, and platforms such as Firebase and Thingspeak, the system uploads AQI and gas sensor data to the cloud at regular intervals. Users can access dashboards via mobile phones or web browsers, allowing for live monitoring, historical trend analysis, and data export for reporting. The study also discusses the use of AQI scales for classifying air quality and the importance of mapping sensor output to understandable metrics (e.g., Good, Moderate, Poor). The integration of cloud services is critical for scalability and for providing actionable insights to school administrators and public health officials.

1.5.4. Design and Development of a Smart Air Pollution Monitoring System Using ESP8266[12]

This project outlines a Wi-Fi-enabled, real-time air monitoring system designed using MQ135, MQ7, and DHT11 sensors. The ESP8266 collects environmental data and sends it to a web dashboard for user interaction. The system calculates the Air Quality Index (AQI) based on pollutant concentrations and displays the results through an OLED display and Serial Monitor. The system is built with energy efficiency and affordability in mind, targeting environments like classrooms, homes, and community centres. It is also capable of triggering alerts through buzzers or visual signals when hazardous levels are detected. The research supports the potential for large-scale deployment in educational institutions to improve awareness and reduce health risks.

1.5.5. Machine Learning-Based Air Quality Prediction and Analysis [14]

This study incorporates machine learning models like Random Forest and XGBoost to predict AQI based on sensor data inputs. Instead of relying solely on fixed thresholds or rule-based AQI classification, the system uses real-time training and regression models to provide more accurate predictions and trends. The researchers used datasets from government stations and compared their outputs with those generated from local low-cost sensors to validate the system's accuracy. The integration of AI enables proactive decision-making and supports data-driven policymaking. Such predictive capabilities are especially useful in school settings, where early warnings can prevent prolonged exposure and inform scheduling of outdoor activities.

CHAPTER – 2

SYSTEM STUDY

2.1 Functional Requirements

2.1.1 User Management:

- Registration, login, logout, and profile management via Firebase Authentication.
- View air quality data and historical trends.

2.1.2 Air Quality Monitoring Functionality:

- Collect data from BME680, PMS7003, MQ135, and MQ7 sensors.
- Display real-time data such as Temperature, Humidity, Pressure, PM2.5, PM10, Gas, and AQI.
- Send real-time data to Firebase Database.
- Generate alerts based on air quality levels (Good, Moderate, Poor, Hazardous).

2.1.3 Data Visualization:

- Display real-time and historical data on the OLED display and web dashboard.
- Provide a web interface for users to view data trends, with interactive charts.

2.1.4 Alarm System:

- Activate buzzer when AQI exceeds certain thresholds.
- Set personalized AQI limits for users to trigger alerts**.**

2.1.5 Database Management:

- Store air quality data (Temperature, Humidity, Pressure, AQI) in Firebase Realtime Database.
- Sync data across multiple devices and platforms.

2.1.6 Web Interface:

- Provide a responsive interface with live data updates.
- Enable users to view and monitor air quality from anywhere via web browsers.

2.2 Non-Functional Requirements

2.2.1 Performance:

- System should update air quality data on the web interface and OLED display every 1 second.
- Firebase real-time database should sync within 2 seconds for live data updates.

2.2.2 Scalability:

- Handle data from multiple IoT devices simultaneously, supporting up to 10,000 data points.
- Support future expansion to include additional sensors or features.

2.2.3 Security:

- Implement Firebase Authentication for secure login and user data protection.
- Ensure data encryption in transit (HTTPS) between devices and Firebase.

2.2.4 Usability:

- User-friendly web interface for easy navigation and data access.
- Simple setup for users to connect their IoT devices and monitor air quality.
- Mobile-compatible interface for users to monitor data on the go.

2.2.5 Reliability:

- Ensure 99.9% uptime for Firebase database.
- Provide robust error handling and fallback mechanisms for sensor data collection.

2.3 User Requirements

2.3.1 End-User:

- Access to real-time air quality data and historical trends.
- Ability to receive alerts on poor air quality and monitor AQI.
- Simple interface for visualizing air quality data.

2.3.2 Admin/Owner:

- Manage the IoT devices, monitor their status, and view real-time data.
- Ability to set thresholds for AQI alerts and customize data visualization settings.

2.3.3 System Developer:

- Ensure smooth integration with Firebase for database management and real-time data syncing.
- Implement reliable communication between the NodeMCU, sensors, and Firebase.

2.4 System Requirements

2.4.1 Software Requirements:

- Backend: Python, Firebase SDK for real-time database communication.
- Frontend: HTML, CSS, JavaScript (for web dashboard), Firebase SDK.
- IoT System: NodeMCU ESP8266, Arduino IDE for sensor communication, Firebase Realtime Database.

2.4.2 Hardware Requirements:

- Sensors: BME680, PMS7003, MQ135, MQ7 for air quality monitoring.
- NodeMCU ESP8266 for Wi-Fi connectivity and data transmission.
- Buzzer for alerting on high AQI levels.
- OLED display for local data display.

2.5 Constraints and Assumptions

2.5.1 Constraints:

- Real-time sensor data updates may be limited by network latency and Firebase synchronization speed.
- The accuracy of air quality readings depends on the sensor calibration and environmental conditions.

2.5.2 Assumptions:

- Users will have stable internet connectivity for real-time data sync with Firebase.
- Sensors will be properly calibrated to provide accurate readings within specified ranges.

CHAPTER – 3

SYSTEM ANALYSIS

3.1 SYSTEM FLOW CHART

The System Flowchart visually represents the overall workflow of the IoT-based air quality monitoring system, detailing how data flows through each component—starting from sensing environmental parameters to displaying and storing data in the Firebase database and showing alerts.

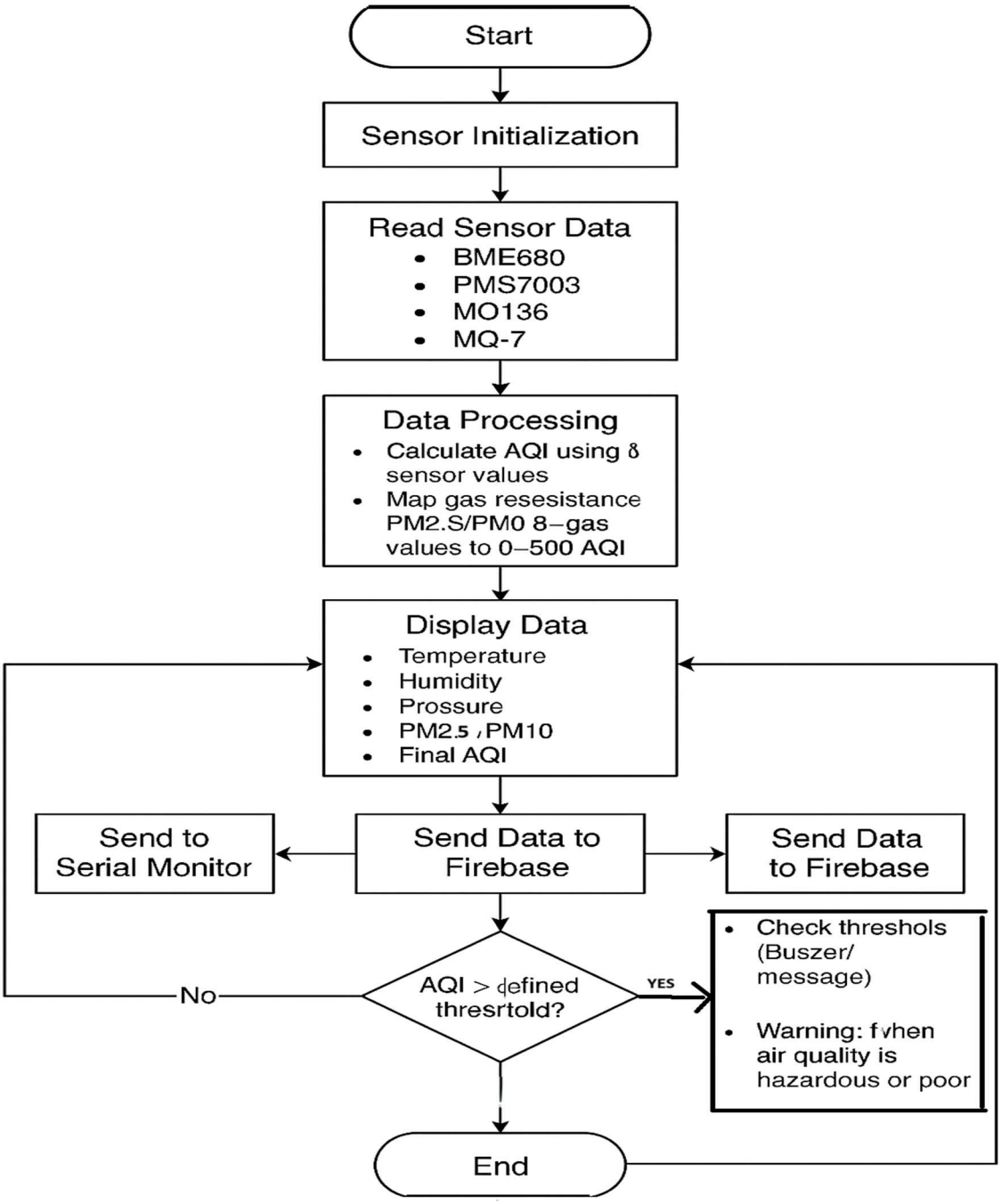


Fig :1 System Flowchart

3.2 Data Flow Diagram (DFDs)

Level 0: Context-Level DFD

This is the highest abstraction level of the system, showing it as a single process:
Entities:

- **User** – Views real-time and historical air quality data through a web interface.
- **Admin** – Manages devices and system configurations.
- **Firebase Database** – Cloud-based data storage and user management system.

Process:

- **Air Quality Monitoring System**

Data Flows:

- Sensor data → Monitoring System
- Processed AQI and environmental data → Firebase
- Firebase → Web Interface (real-time data for users/admins)

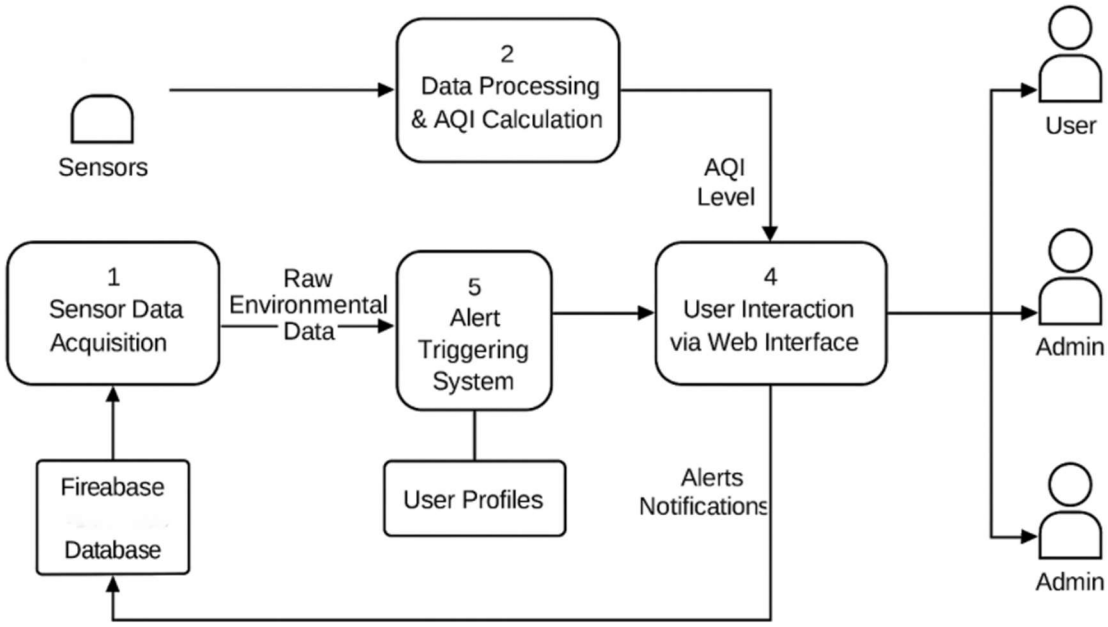


Fig:2 Level 0 DFD

Level 1: Detailed DFD

This level breaks down the internal structure of the system:

Processes:

1. **Sensor Data Acquisition**
    - Gathers environmental data (Temperature, Humidity, Pressure, PM2.5, PM10, Gas concentration) from BME680, PMS7003, MQ135, and MQ7.
2. **Data Processing & AQI Calculation**
    - Converts raw sensor readings into meaningful AQI values.

- Categorizes air quality status (e.g., Good, Poor, Hazardous).

3. **Data Storage in Firebase**
   - Sends and stores real-time data in Firebase Realtime Database.
   - Maintains historical logs for trend analysis.
4. **User Interaction via Web Interface**
   - Displays live and historical AQI data.
   - Allows user login/registration and profile viewing.
   - Displays alerts and notifications for critical AQI levels.
5. **Alert Triggering System**
   - Monitors AQI threshold.
   - Triggers buzzer or notification if AQI exceeds safe levels.

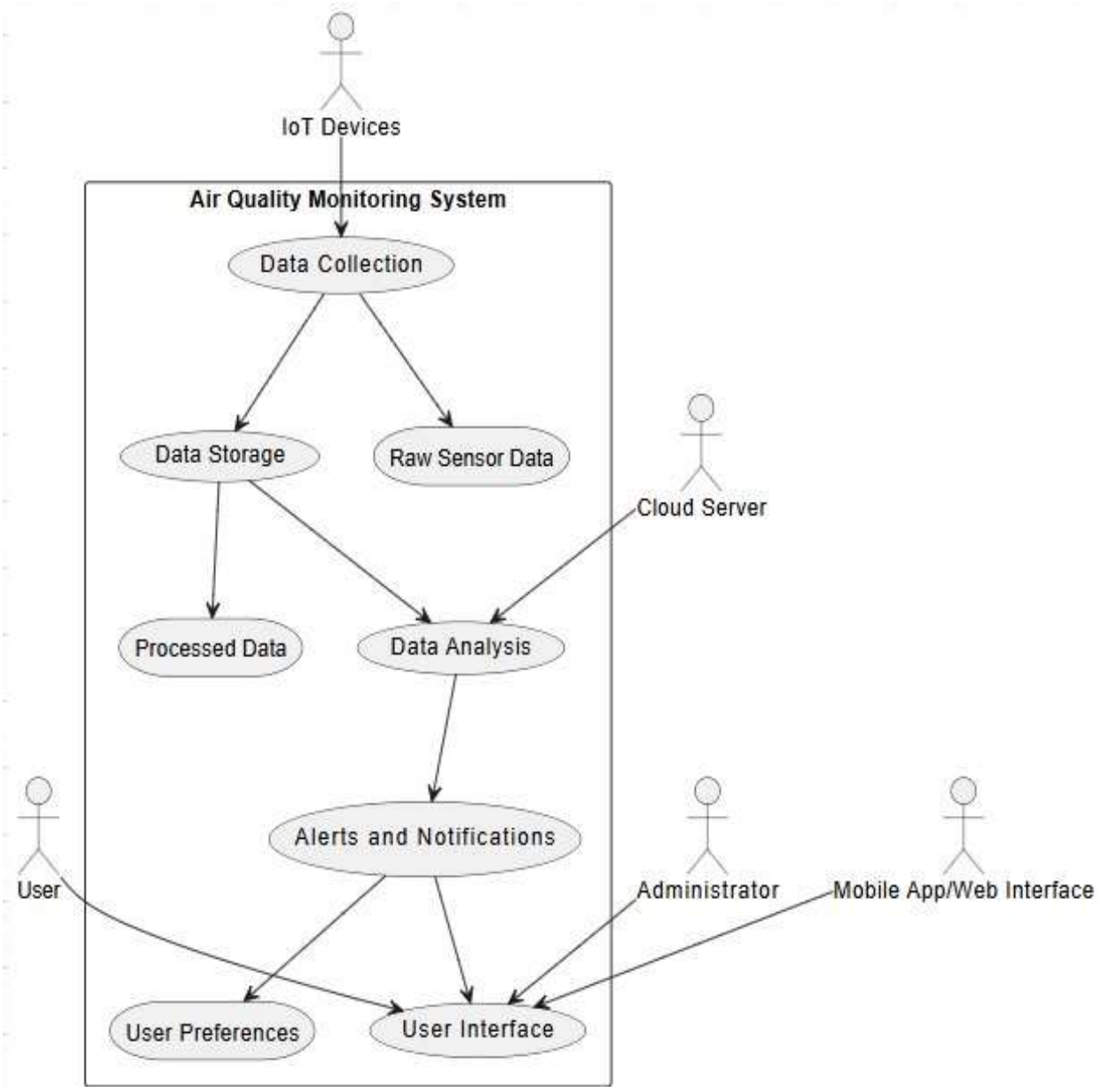


Fig: 3 Level 1 DFD

3.3 Use case Diagram

The Use Case Diagram illustrates the functional requirements of an IoT-Based Air Quality Monitoring System, showing how different actors interact with the system to achieve various goals.

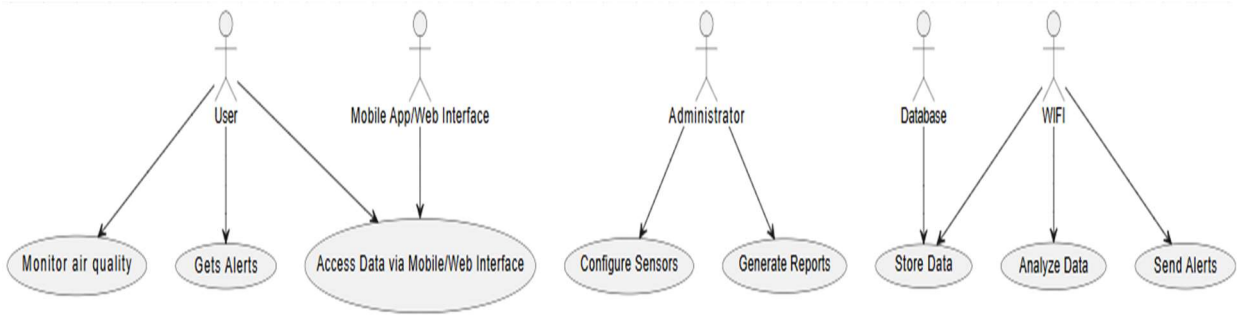


Fig : 4 Use Case Diagram

3.4 Sequence Diagram

The sequence diagram illustrates the **step-by-step interaction** among various components involved in the **IoT Air Quality Monitoring System**, including the **User**, **Mobile App**, **Database**, **IoT Gateway**, and **Wi-Fi module**. It visually represents how the system responds to a user's request for air quality data and the process flow from data collection to data visualization.

3.4.1 User Interaction:

3.4.2 The sequence begins with the **user opening the mobile application**.

- The user then **requests air quality data** via the mobile app interface.

3.4.3 Mobile App to Database Communication:

- Upon receiving the request, the mobile app sends a **data request to the database**.

3.4.4 Database to IoT Gateway:

- The database initiates the process by attempting to **retrieve data from the IoT gateway**.

- The gateway, in turn, **fetches the data from the microcontroller**, which is connected to various air quality sensors.
- The microcontroller **collects real-time data** from the sensors (e.g., PM2.5, PM10, MQ135, MQ7, temperature, humidity, pressure).

3.4.5 Data Upload and Transmission:

- Once the sensor data is collected, it is **uploaded to the database** by the IoT gateway.
- The **Wi-Fi module** facilitates this data transfer, ensuring the information is updated in real-time.

3.4.6 Data Visualization:

- The updated data in the database is then **sent to the mobile app**, allowing the user to **view the air quality information** directly.
- Simultaneously, the same data is **displayed on a web dashboard**, accessible via the internet for broader monitoring and analysis.

3.4.7 End of Interaction:

- Finally, the **mobile app displays the air quality data** to the user, completing the sequence.

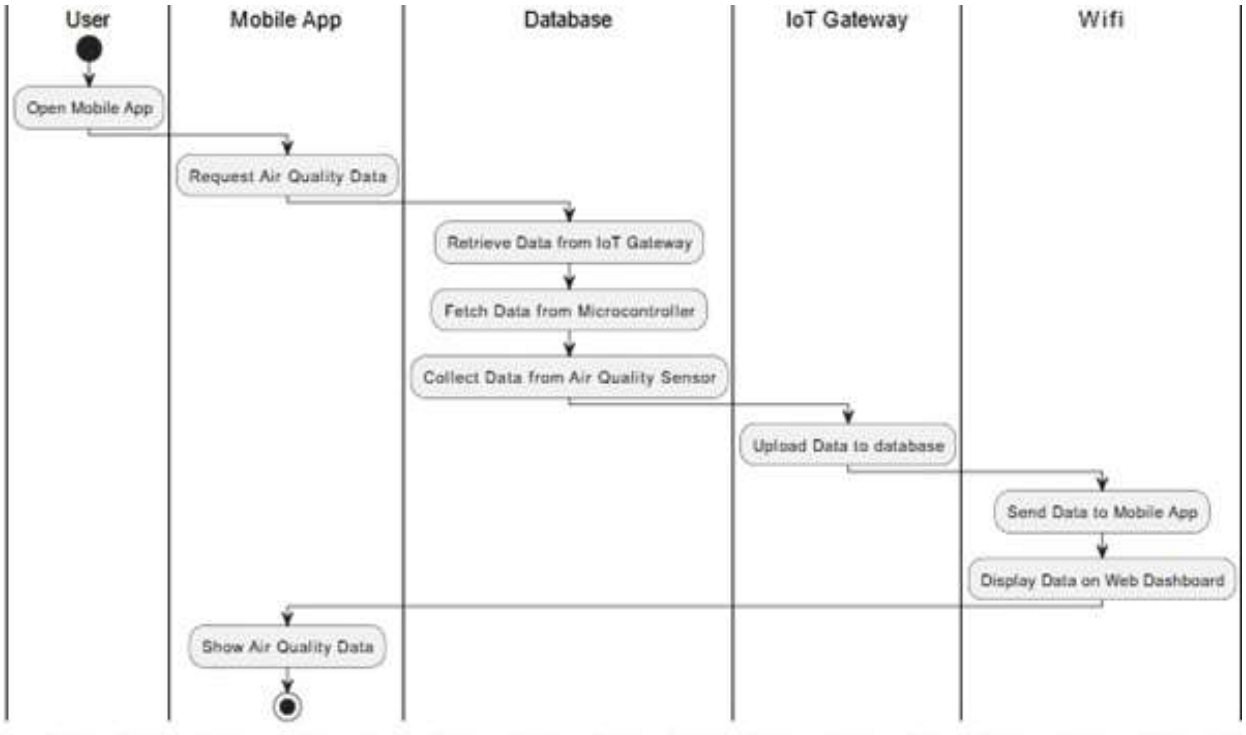


Fig: 5 Sequence Diagram

CHAPTER - 4

SYSTEM DESIGN

4.1 System Architecture

The **system architecture** of the IoT-Based Air Quality Monitoring System is designed to collect, process, store, and display environmental data in real-time. The architecture ensures reliable data transmission, efficient sensor management, and user-friendly access to air quality information. It is composed of four main layers:

4.1.1 Sensing Layer (Hardware Layer)

This layer includes all the environmental sensors responsible for data acquisition.

a. **Sensors Used**:
   - **BME680**: Measures temperature, humidity, pressure, and gas resistance.
   - **PMS7003**: Measures PM2.5 and PM10 particles.
   - **MQ-135**: Detects harmful gases like NH_3, NO_x, alcohol, benzene, smoke.
   - **MQ-7**: Detects carbon monoxide (CO).
   - **Buzzer**: Alerts users in poor or hazardous air quality conditions.
   - **OLED Display**: Displays real-time data for user visibility.

b. **Microcontroller**:
   - **NodeMCU ESP8266 (12E)**: Collects sensor data, processes it, and sends it over Wi-Fi to the cloud.

4.1.2 Network Layer

This layer handles data transmission from the microcontroller to the cloud.

a. **Wi-Fi Module** (built into NodeMCU ESP8266):
   - Connects the device to the internet.
   - Sends sensor data to the cloud database (Firebase).
   - Enables real-time communication for alerting and dashboard updates.

4.1.3 Cloud Layer (Backend Layer)

Handles storage, analytics, and access to the sensor data.

a. **Firebase (Google Cloud Platform)**:
   - Stores real-time sensor readings.
   - Maintains user data, alert thresholds, and device configuration.
   - Supports real-time database updates and API access.

b. **Data Analytics** (optional):
   - Aggregates historical data for trends and report generation.
   - Can be extended to integrate AI/ML models for AQI prediction and analysis.

4.1.4 Application Layer (User Interface Layer)

Provides user access to data and system control.

- **Mobile/Web Interface**:
   - Displays live and historical data in graphical format.

- Allows users to register, log in, and view personalized air quality insights.
- Sends alerts via the interface based on real-time AQI values.

4.2 Hardware Design

The **hardware design** of the IoT-based air quality monitoring system consists of multiple environmental sensors, a microcontroller unit, and output/display components. These components are integrated to collect real-time environmental data and relay it to both local and cloud platforms. Below is a detailed description of each component involved and their interconnections.

4.2.1. Microcontroller Unit

a. **NodeMCU ESP8266 (12E)**
   - Acts as the central processing unit.
   - Reads data from sensors via analog/digital/I2C/UART interfaces.
   - Provides Wi-Fi connectivity to send data to Firebase.
   - Operates at 3.3V logic level.

Fig: 6 Microcontroller

4.2.2 Sensors

a. **BME680**
   - **Parameters**: Temperature, Humidity, Pressure, Gas Resistance.
   - **Interface**: I2C (SCL → D1, SDA → D2).
   - **Power**: 3.3V.

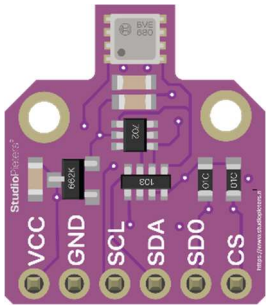


Fig: 7 BME 680

b. **PMS7003 (Particulate Matter Sensor)**
   - **Parameters**: PM1.0, PM2.5, PM10.
   - **Interface**: UART (TX → D6, RX → D7).
   - **Power**: 5V.

Fig: 8 PMS7003

c. **MQ-135 (Air Quality Sensor)**
   - **Parameter**: Detects gases like NH_3, NO_x, benzene, CO_2, smoke.
   - **Interface**: Analog (A0).
   - **Power**: 5V.

Fig: 9 MQ 135

d. **MQ-7 (Carbon Monoxide Sensor)**
   - **Parameter**: Detects CO gas levels.
   - **Interface**: Digital (D5).
   - **Power**: 5V (intermittent heating cycle recommended for accuracy).

Fig: 10 MQ-07

4.2.3 Output Components

a. **OLED Display (0.96 inch, I2C)**
   - Displays real-time readings (Temperature, Humidity, Pressure, PM2.5, AQI).
   - **Interface**: I2C (shares D1 and D2 with BME680).
   - **Power**: 3.3V.

Fig: 11 Oled Display

b. **Buzzer**
   - Provides audible alerts if AQI crosses a danger threshold.
   - **Interface**: Digital GPIO (e.g., D3).
   - **Power**: 3.3V or via transistor control.

Fig: 12 Buzzer

4.2.4 Power Supply

- **Micro-USB or VIN (5V)**
  - Powers the NodeMCU and peripherals.
  - Sensors like PMS7003 and MQ series are powered from VIN (5V) pin.
  - Use level shifters or resistors if voltage mismatch is an issue.

4.2.5 3D Printed Custom Enclosure

1. Designed and fabricated using 3D printing for the entire system.
2. Purpose:
   - Protects internal electronics from dust and accidental damage.
   - Allows structured placement of sensors for accurate readings.
   - Includes ventilation slots for proper airflow to gas sensors.
   - Transparent or cut-out section for the OLED display visibility.
   - Compact and aesthetic design for indoor installation.
3. Material used: PLA/ABS (as applicable).
4. Benefits:
   - Enhances portability and durability.
   - Provides thermal stability and mechanical support.

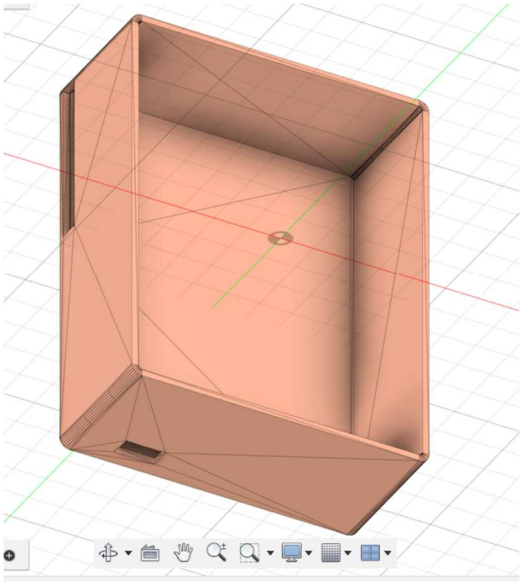


Fig: 13 3d case design

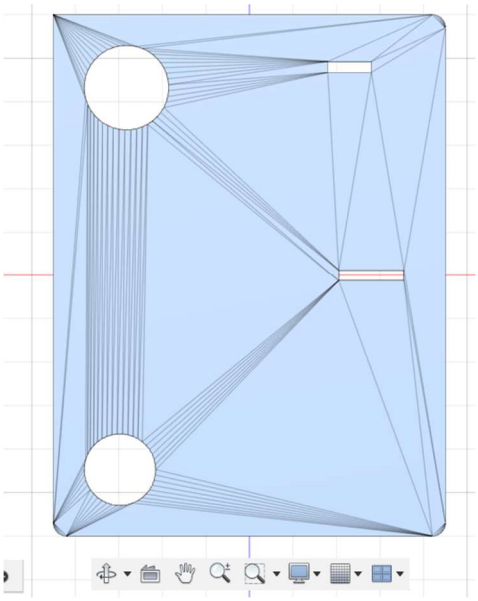


Fig: 14 3d lid design

4.3 Circuit Design

The **circuit design** involves interfacing multiple sensors and output components with the NodeMCU ESP8266 microcontroller, enabling real-time environmental data acquisition, processing, and transmission over Wi-Fi. The circuit is structured for low power consumption, accurate signal transmission, and optimal sensor placement for reliable monitoring.

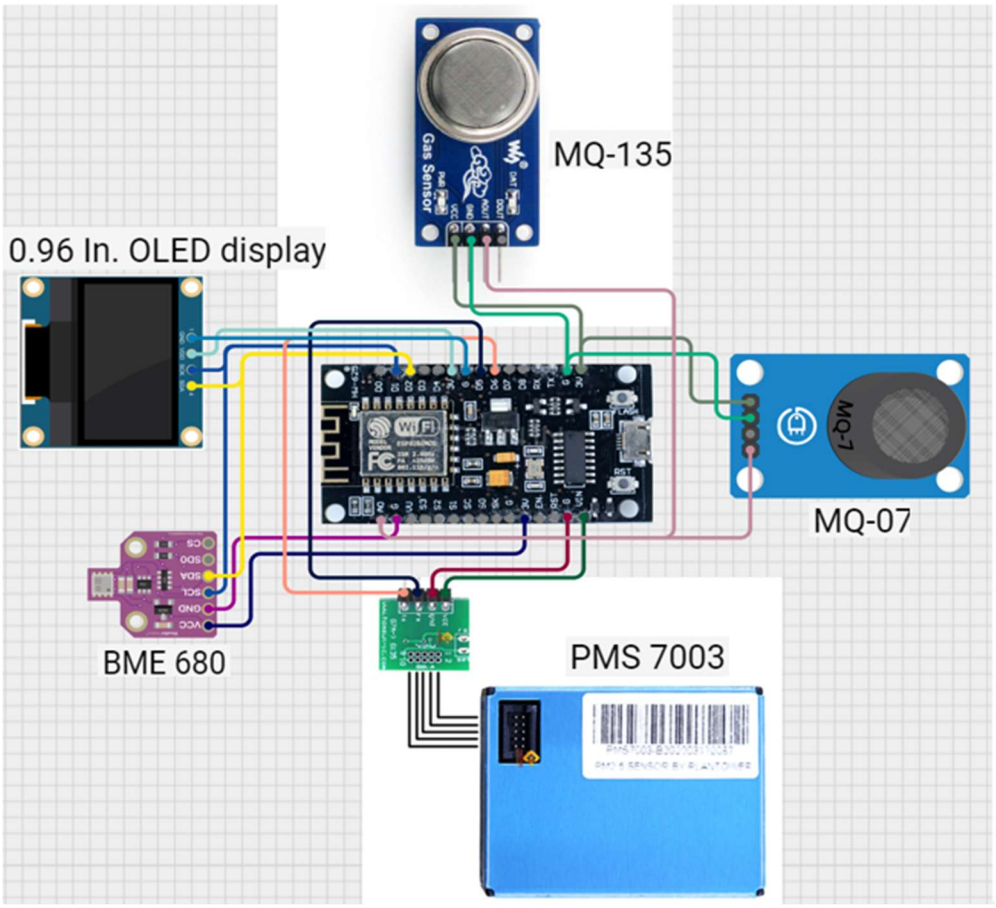


Fig: 15 Circuit Diagram

Component	Interface Type	NodeMCU Pin	Power Source
BME680	I2C	D2 (SDA), D1 (SCL)	3.3V
PMS7003	UART	D7 (RX), D6 (TX)	5V
MQ-135	Analog	A0	5V
MQ-7	Digital	D5	5V
OLED Display	I2C	D2, D1 (shared)	3.3V
Buzzer	Digital	D3	3.3V

Table: 1 Circuit Connections

4.4 User Interface Design

The web interface of the **IoT Air Quality Monitoring System** is designed to provide users with a clear and real-time visualization of indoor air quality parameters. It is organized into distinct sections that deliver both **live sensor readings** and **historical trends** through dynamic charts and tables. Below is a breakdown of the features and design elements based on the images provided:

4.4.1. Header Section

- **Title**: Clearly labelled as **"IoT Air Quality Monitoring System"**, immediately informing users of the system's purpose.
- **Subtitle**: "Monitor air quality in real-time with hourly updates" gives a brief description of the system's functionality.

4.4.2. Live Air Quality Data Panel

This section presents **real-time environmental readings**, sourced from various sensors:

- **PM2.5 & PM10**: Shows the concentration of fine particulate matter (in $\mu g/m^3$).
- **MQ135 (Gas Sensor)**: Displays gas concentration in ppm, typically indicative of air pollutants like CO_2, NH_3, etc.
- **Temperature & Humidity**: Reported in °C and % respectively.
- **Pressure**: Atmospheric pressure in hPa.
- **MQ7 (Carbon Monoxide Sensor)**: Displays CO gas levels in ppm.
- **AQI (Air Quality Index)**: Derived from sensor data and mapped to an EPA-style index (0–500 scale).
- **Air Quality Status**: A qualitative label (e.g., Good, Moderate) based on AQI values.

4.4.3. Hourly Air Quality Data Table

- Displays time-stamped data readings for each hour.
- Includes all sensor parameters: PM2.5, PM10, Temperature, Humidity, Pressure, MQ7, MQ135, AQI, and Air Quality status.
- Provides historical insights for data comparison over time.

- Some cells show "undefined", indicating possible sensor disconnection or missing data during those intervals.

4.4.4. Air Quality Trends (Graphs)

This section contains interactive graphs that show how sensor data changes over time:

- **PM2.5 & PM10 Trends**: Dual-line graph in red and blue, respectively.
- **Temperature/Humidity**: Displayed in a single yellow trend line.
- **MQ7 (Gas)**: Purple line tracking carbon monoxide levels.
- **MQ135**: Green line showing variations in general air pollutant levels.

These trend graphs help visualize environmental fluctuations and detect patterns in air quality.

Fig: 16 Web interface 1

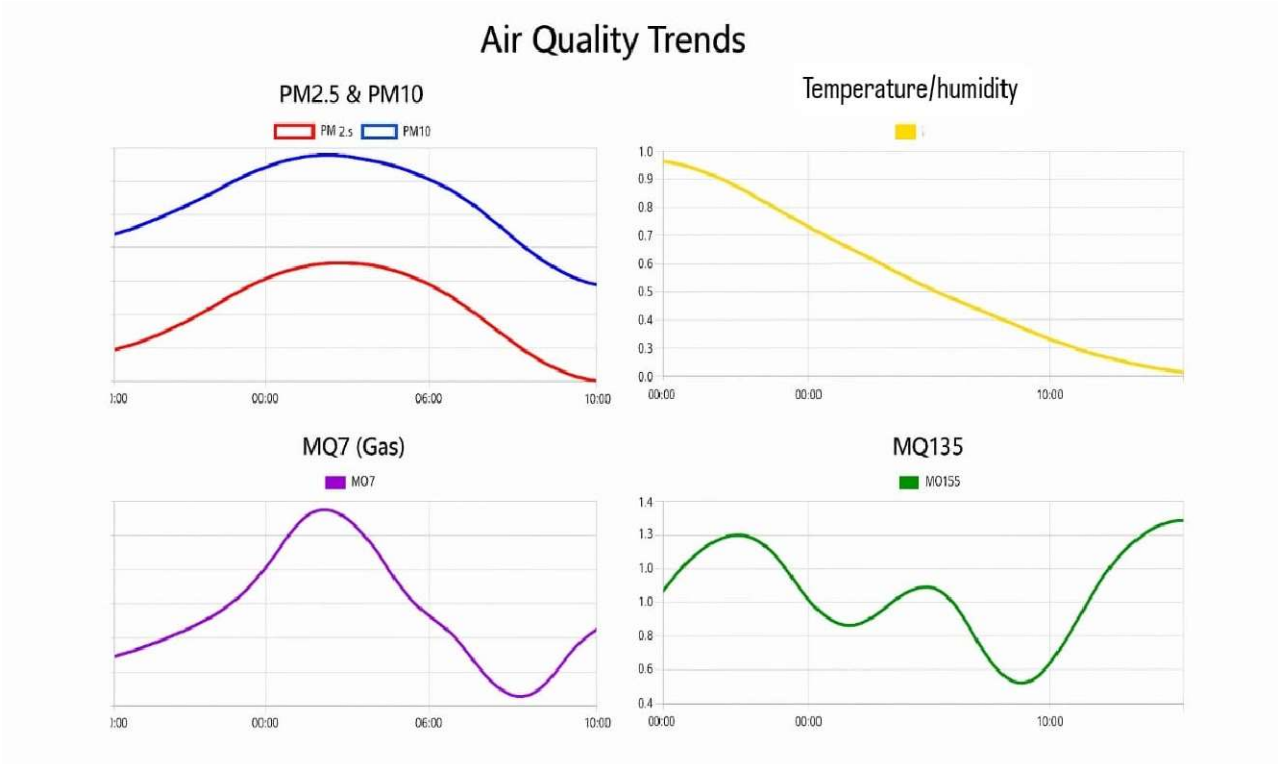


Fig: 17 Web interface 2

CHAPTER -5

SYSTEM IMPLEMENTATION

5.1 Development Tools

5.1.1 Technologies Used

The system utilizes a combination of hardware-level programming, embedded systems, cloud integration, and front-end technologies to enable real-time environmental monitoring and data visualization.

a. Microcontroller Platform

- **NodeMCU ESP8266 (ESP-12E)**:
  Acts as the central processing and communication unit, responsible for collecting data from sensors and sending it to the cloud via Wi-Fi.

b. Programming Language

- **C++ with Arduino IDE**:
  Used for writing and uploading firmware to the NodeMCU. Libraries such as Wire.h, Adafruit_Sensor.h, and FirebaseESP8266.h are used.

c. Sensor Libraries

- **Adafruit BME680 Library**: For environmental sensor data.
- **PMS Library**: For interfacing the PMS7003 dust sensor.
- **MQ Sensor Calibration Scripts**: For interpreting analog gas sensor outputs.

d. Communication Protocols

- **I2C and UART**: Used for communication with BME680, OLED, and PMS7003 respectively.
- **Wi-Fi (IEEE 802.11 b/g/n)**: Enabled by the ESP8266 module for cloud communication.

e. Display Technology

- **OLED Display (0.96", I2C)**:
  Used for real-time display of AQI, temperature, humidity, and gas levels.

f. Web and Data Visualization (Optional/Extended Version)

- **HTML, CSS, JavaScript**:
  If a web dashboard is implemented to view data remotely.

- **Firebase Realtime Database SDK**:
  For web-based retrieval and live visualization of sensor data.

5.1.2 Database Used

To store and monitor air quality data in real time, we used:

a. Google Firebase Realtime Database

- A cloud-hosted NoSQL database used to store sensor readings and environmental parameters.
- Offers **real-time data synchronization** across clients, enabling quick access and updates.
- **Authentication** and **rules-based access control** ensure data security.
- **REST APIs** used by NodeMCU to send data in JSON format via HTTPS.
- Enables **live dashboard integration**, mobile app backend support, and scalable IoT data handling.

b. Data Format

The NodeMCU ESP8266 collects data from multiple sensors and sends it in structured JSON format to the Firebase Realtime Database. Each entry includes environmental parameters and timestamp information for real-time tracking.

c. Sample JSON format :

"Temperature": "27.4",
"Humidity": "56.2",
"Pressure": "1008.5",
"PM2_5": "35",
"PM10": "48",
"MQ135": "378",
"MQ7": "221",
"FlammableGas": "682",
"AQI": "85",
"AirQuality": "Moderate",
"Timestamp": "2025-05-05 14:30:15"

5.2 Coding

5.2.1 Html

This code comprises two HTML pages for an IoT-based Air Quality Monitoring System website. The first page serves as a real-time dashboard displaying live environmental data such as PM2.5, PM10, temperature, humidity, and air quality status using Bootstrap for layout and Chart.js for visual trends, along with a table showing hourly updates. The second page is an "About Us" section that introduces the project team members with their photos and names in a responsive grid format. Both pages feature a consistent design with headers, footers, and navigation elements, offering a user-friendly interface to monitor and understand air quality data and the team behind the project.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>IoT Air Quality Monitoring</title>
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"/>
  <link rel="stylesheet" href="style.css" />
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

  <script type="module" src="script.js"></script>
</head>
<body>
  <header class="bg-primary text-white text-center py-4">
    <h1>IoT Air Quality Monitoring System</h1>
    <p>Monitor air quality in real-time with hourly updates</p>
  </header>

  <div class="container my-4">
    <!-- Live Data -->
    <section class="mb-4">
      <h2 class="text-center">Live Air Quality Data</h2>
      <div class="row text-center mt-4">
        <div class="col-md-2"><h5>PM2.5</h5><p id="pm25-level" class="data-value">--
</p></div>
        <div class="col-md-2"><h5>PM10</h5><p id="pm10-level" class="data-value">--
</p></div>
        <div class="col-md-2"><h5>Temp</h5><p id="temperature" class="data-value">--
</p></div>
        <div class="col-md-2"><h5>Humidity</h5><p id="humidity" class="data-value">--
</p></div>
        <div class="col-md-2"><h5>Air Quality</h5><span id="air-quality-badge"
class="badge bg-secondary">--</span></div>
      </div>
    </section>
```

```
    <!-- Hourly Table -->
    <section class="mb-4">
      <h2 class="text-center">Hourly Air Quality Data</h2>
      <table class="table table-bordered table-hover mt-3">
        <thead class="table-light">
          <tr>

<th>Time</th><th>PM2.5</th><th>PM10</th><th>Temp</th><th>Humidity</th><th>A
ir Quality</th>
          </tr>
        </thead>
        <tbody id="hourly-data-table"></tbody>
      </table>
    </section>

    <!-- Chart -->
    <section>
      <h2 class="text-center">Air Quality Trends</h2>
      <canvas id="airQualityChart" class="mt-3"></canvas>
    </section>
  </div>

  <footer class="bg-primary text-white text-center py-3">
    <p>&copy; 2024 IoT Air Quality Monitoring</p>
  </footer>
</body>
</html>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>About Us</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header class="bg-primary text-white text-center py-4">
        <h1>About the Team</h1>
        <p>The brilliant minds behind the IoT Air Quality Monitoring System</p>
    </header>

    <div class="container my-5">
        <div class="row g-4 text-center">
            <div class="col-md-4">
                <img src="IMG_20241209_025119.jpg" alt="Harsh Sahay" class="rounded-
circle mb-3" width="150">
```

```
            <h4>Harsh Sahay</h4>

          </div>
          <div class="col-md-4">
            <img src="IMG_20241209_032711.jpg" alt="Apurav Singh" class="rounded-
circle mb-3" width="150">
            <h4>Apurav Singh</h4>

          </div>
          <div class="col-md-4">
            <img src="IMG_20241209_025243.jpg" alt="Ajeet Yadav" class="rounded-
circle mb-3" width="150">
            <h4>Ajeet Yadav</h4>

          </div>
          <div class="col-md-4">
            <img src="IMG_20241209_032852.jpg" alt="Harsh Raghav" class="rounded-
circle mb-3" width="150">
            <h4>Harsh Raghav</h4>

          </div>
          <div class="col-md-4">
            <img src="IMG_20241209_032408.jpg" alt="Arpna Saxena" class="rounded-
circle mb-3" width="150">
            <h4>Arpna Saxena</h4>

          </div>
        </div>
    </div>

    <footer class="bg-primary text-white text-center py-3">
        <p>&copy; 2024 IoT Air Quality Monitoring System</p>
        <a href="index.html" class="btn btn-light">Back to Home</a>
    </footer>
</body>
</html>
```

5.2.2 CSS

This CSS styles the IoT Air Quality Monitoring website with a clean, modern look—using a light background, blue-themed header/footer, card-style data sections with hover effects, styled tables, and a responsive chart container—all aimed at clear and user-friendly data presentation.

```
body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  background-color: #f5f5f5;
  margin: 0;
  padding: 0;
}

header {
  background-color: #007bff;
  color: white;
  padding: 2rem 1rem;
  box-shadow: 0 4px 8px rgba(0,0,0,0.1);
}

footer {
  background-color: #007bff;
  color: white;
  padding: 1rem;
  margin-top: 3rem;
}

.data-card {
  background-color: #ffffff;
  padding: 1rem;
  margin: 0.5rem;
  border-radius: 10px;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
  transition: transform 0.2s ease;
}

.data-card:hover {
  transform: scale(1.03);
}

.data-value {
  font-size: 1.4rem;
  font-weight: bold;
  color: #333;
}

.table {
  background-color: white;
```

```
  border-radius: 8px;
  overflow: hidden;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.05);
}

.table th, .table td {
  text-align: center;
  vertical-align: middle;
}

.badge {
  font-size: 1rem;
  padding: 0.5rem 0.75rem;
  border-radius: 20px;
}

#airQualityChart {
  width: 100% !important;
  max-height: 400px;
  background-color: white;
  padding: 1rem;
  border-radius: 12px;
  box-shadow: 0 2px 6px rgba(0,0,0,0.1);
  margin-top: 1rem;
}
```

5.2.3 Javascript

This JavaScript connects to a Firebase Realtime Database to fetch and display live and hourly air quality data, updating the webpage with current readings like PM2.5, PM10, temperature, and humidity. It also dynamically updates a table and generates multiple interactive line charts (AQI, MQ135, MQ7, pressure, PM values) using Chart.js for visual analysis, providing real-time and historical environmental monitoring for the IoT-based air quality system.

```
import { initializeApp } from
"https://www.gstatic.com/firebasejs/10.11.0/firebase-app.js";
import { getDatabase, ref, onValue } from
"https://www.gstatic.com/firebasejs/10.11.0/firebase-database.js";

const firebaseConfig = {
  apiKey: "AIzaSyBzdxGZiIAs1opcuNzIYBIdCAulEyEssYQ",
  authDomain: "airqualitymonitor-df0db.firebaseapp.com",
```

```
  databaseURL: "https://airqualitymonitor-df0db-default-rtdb.asia-
southeast1.firebasedatabase.app",
  projectId: "airqualitymonitor-df0db",
  storageBucket: "airqualitymonitor-df0db.appspot.com",
  messagingSenderId: "102056390066",
  appId: "1:102056390066:web:7f570d86855b3f619b88ab",
  measurementId: "G-74HVSPH2P9"
};

const app = initializeApp(firebaseConfig);
const db = getDatabase(app);

document.addEventListener("DOMContentLoaded", () => {
  fetchLiveData();
  fetchHourlyData();
});

function fetchLiveData() {
  const liveRef = ref(db, "airquality/data");
  onValue(liveRef, snapshot => {
    const data = snapshot.val();
    console.log("Live Data:", data);
    if (data) updateLiveData(data);
    else console.warn("No live data found.");
  }, (error) => {
    console.error("Error fetching live data:", error);
  });
}

function fetchHourlyData() {
  const hourlyRef = ref(db, "hourly");
  onValue(hourlyRef, snapshot => {
    const data = snapshot.val();
    console.log("Hourly Data:", data);

    if (!data) {
      console.warn("No hourly data found.");
      return;
    }

    const hourlyArray = [];
    for (let time in data) {
```

```
      hourlyArray.push({ time, ...data[time] });
    }

    hourlyArray.sort((a, b) => a.time.localeCompare(b.time));
    updateHourlyTable(hourlyArray);
    updateAllCharts(hourlyArray);
  }, (error) => {
    console.error("Error fetching hourly data:", error);
  });
}

function updateLiveData(data) {
  document.getElementById("pm25-level").innerText = ${data.pm25} µg/m³;
  document.getElementById("pm10-level").innerText = ${data.pm10} µg/m³;
  document.getElementById("temperature").innerText = ${data.temperature}
°C;
  document.getElementById("humidity").innerText = ${data.humidity} %;

  const badge = document.getElementById("air-quality-badge");
  badge.innerText = data.airQuality;
  badge.className = "badge";

  switch (data.airQuality) {
    case "Good":
      badge.classList.add("bg-success");
      break;
    case "Moderate":
      badge.classList.add("bg-warning");
      break;
    case "Unhealthy":
      badge.classList.add("bg-danger");
      break;
    default:
      badge.classList.add("bg-secondary");
  }
}

function updateHourlyTable(data) {
  const tableBody = document.getElementById("hourly-data-table");
  tableBody.innerHTML = "";
  data.forEach(entry => {
    const row = `
```

```
      <tr>
        <td>${entry.time}</td>
        <td>${entry.pm25}</td>
        <td>${entry.pm10}</td>
        <td>${entry.temperature}</td>
        <td>${entry.humidity}</td>
        <td>${entry.airQuality}</td>
      </tr>`;
    tableBody.innerHTML += row;
  });
}

let charts = {};

function updateAllCharts(data) {
  const times = data.map(d => d.time);

  const chartConfigs = [
    {
      id: "airQualityChart",
      label: ["PM2.5", "PM10"],
      datasets: [
        {
          label: "PM2.5",
          data: data.map(d => d.pm25),
          borderColor: "rgba(255, 99, 132, 1)",
          backgroundColor: "rgba(255, 99, 132, 0.2)"
        },
        {
          label: "PM10",
          data: data.map(d => d.pm10),
          borderColor: "rgba(54, 162, 235, 1)",
          backgroundColor: "rgba(54, 162, 235, 0.2)"
        }
      ]
    },
    {
      id: "pressureChart",
      label: "Pressure",
      datasets: [{
        label: "Pressure (hPa)",
        data: data.map(d => d.pressure),
```

```
      borderColor: "rgba(75, 192, 192, 1)",
      backgroundColor: "rgba(75, 192, 192, 0.2)"
    }]
  },
  {
    id: "mq7Chart",
    label: "MQ7",
    datasets: [{
      label: "MQ7 (ppm)",
      data: data.map(d => d.mq7),
      borderColor: "rgba(255, 159, 64, 1)",
      backgroundColor: "rgba(255, 159, 64, 0.2)"
    }]
  },
  {
    id: "mq135Chart",
    label: "MQ135",
    datasets: [{
      label: "MQ135 (ppm)",
      data: data.map(d => d.mq135),
      borderColor: "rgba(153, 102, 255, 1)",
      backgroundColor: "rgba(153, 102, 255, 0.2)"
    }]
  },
  {
    id: "aqiChart",
    label: "AQI",
    datasets: [{
      label: "AQI",
      data: data.map(d => d.aqi),
      borderColor: "rgba(255, 205, 86, 1)",
      backgroundColor: "rgba(255, 205, 86, 0.2)"
    }]
  }
];

chartConfigs.forEach(config => {
  const ctx = document.getElementById(config.id)?.getContext("2d");
  if (!ctx) {
    console.warn(Canvas with id ${config.id} not found.);
    return;
  }
```

```
    if (charts[config.id]) charts[config.id].destroy();

    charts[config.id] = new Chart(ctx, {
      type: "line",
      data: {
        labels: times,
        datasets: config.datasets
      },
      options: {
        responsive: true,
        scales: {
          y: { beginAtZero: true },
          x: { title: { display: true, text: "Time" } }
        }
      }
    });
  });
}
```

5.2.4 Arduino IDE (Embedded C)

The Arduino sketch is designed for an **IoT-based real-time air quality monitoring system** using a **NodeMCU ESP8266** microcontroller. It interfaces with multiple environmental sensors—**BME680**, **PMS7003**, **MQ135**, and **MQ7**—to collect data such as temperature, humidity, pressure, gas resistance, particulate matter (PM2.5 and PM10), and harmful gas levels. The system also uses an **OLED display** for real-time data visualization and integrates with **Google Firebase** to store and monitor the data remotely via Wi-Fi.

```
#include <ESP8266WiFi.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME680.h>
#include <FirebaseESP8266.h>
#include <SoftwareSerial.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_GFX.h>

// OLED Settings
```

```
#define SCREEN_ADDRESS 0x3C
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
-1);

// Wi-Fi & Firebase
#define WIFI_SSID "Apurav's S23 FE"
#define WIFI_PASSWORD "qwerty1234"
#define FIREBASE_HOST "airqualitymonitor-df0db-default-rtdb.asia-
southeast1.firebasedatabase.app"
#define FIREBASE_AUTH
"eoYQGSmlGEIzDPC5FSRxtwAGTbVjmu47uPQHGQeT"
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

// Sensor Pins
#define MQ7_DOUT D5
#define MQ135_AOUT A0
Adafruit_BME680 bme;
SoftwareSerial pmsSerial(D6, D7); // RX, TX

void setup() {
  Serial.begin(115200);
  pmsSerial.begin(9600);

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("WiFi connected");

  config.host = FIREBASE_HOST;
  config.signer.tokens.legacy_token = FIREBASE_AUTH;
  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);

  if (!bme.begin()) {
    Serial.println("BME680 not found!");
    while (1);
```

```
  }
  bme.setTemperatureOversampling(BME680_OS_8X);
  bme.setHumidityOversampling(BME680_OS_2X);
  bme.setPressureOversampling(BME680_OS_4X);
  bme.setGasHeater(320, 150);

  if (!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("SSD1306 allocation failed"));
    while (1);
  }
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.print("Initializing...");
  display.display();
  delay(2000);
}

int calculateAQI(int pm25, int pm10, int mq135, int gasRes) {
  int score = 0;

  // PM2.5 AQI contribution
  if (pm25 >= 0) {
    if (pm25 <= 12) score += 0;
    else if (pm25 <= 35) score += 50;
    else if (pm25 <= 55) score += 100;
    else if (pm25 <= 150) score += 200;
    else score += 300;
  }

  // PM10 AQI contribution
  if (pm10 >= 0) {
    if (pm10 <= 54) score += 0;
    else if (pm10 <= 154) score += 50;
    else if (pm10 <= 254) score += 100;
    else if (pm10 <= 354) score += 200;
    else score += 300;
  }

  // MQ135 AQI contribution
  if (mq135 >= 0) {
```

```
  if (mq135 < 100) score += 0;
  else if (mq135 < 300) score += 50;
  else if (mq135 < 500) score += 100;
  else if (mq135 < 700) score += 200;
  else score += 300;
 }

 // Gas resistance AQI contribution
 if (gasRes > 100000) score += 0;
 else if (gasRes > 50000) score += 50;
 else if (gasRes > 20000) score += 100;
 else score += 200;

 return min(score, 500);  // Max AQI capped at 500
}

void loop() {
 bme.performReading();
 float temp = bme.temperature;
 float hum = bme.humidity;
 float pres = bme.pressure / 100.0;
 int gas_res = bme.gas_resistance;

 // --- PMS7003 Original Logic ---
 uint8_t buffer[32];
 int pm25 = -1, pm10 = -1;
 if (pmsSerial.available() >= 32) {
  if (pmsSerial.read() == 0x42) {
   buffer[0] = 0x42;
   buffer[1] = pmsSerial.read();
   for (int i = 2; i < 32; i++) {
    buffer[i] = pmsSerial.read();
   }
   pm25 = buffer[12] << 8 | buffer[13];
   pm10 = buffer[14] << 8 | buffer[15];
  }
 }

 int mq135_val = analogRead(MQ135_AOUT);
 int mq7_val = digitalRead(MQ7_DOUT);

 int aqi = calculateAQI(pm25, pm10, mq135_val, gas_res);
```

```
// --- Serial Output ---
Serial.println("-----------");
Serial.printf("Temp: %.2f °C\n", temp);
Serial.printf("Humidity: %.2f %%\n", hum);
Serial.printf("Pressure: %.2f hPa\n", pres);
Serial.printf("Gas: %d Ω\n", gas_res);
Serial.printf("PM2.5: %d µg/m3\n", pm25);
Serial.printf("PM10: %d µg/m3\n", pm10);
Serial.printf("MQ135: %d\n", mq135_val);
Serial.printf("MQ7: %d\n", mq7_val);
Serial.printf("AQI: %d\n", aqi);

// --- Firebase Send ---
FirebaseJson json;
json.set("temperature", temp);
json.set("humidity", hum);
json.set("pressure", pres);
json.set("gas", gas_res);
json.set("pm25", pm25);
json.set("pm10", pm10);
json.set("mq135", mq135_val);
json.set("mq7", mq7_val);
json.set("aqi", aqi);

if (Firebase.setJSON(fbdo, "/airquality/data", json)) {
  Serial.println("Data sent to Firebase");
} else {
  Serial.println("Firebase send failed: " + fbdo.errorReason());
}
// --- OLED Display ---
display.clearDisplay();
display.setCursor(0, 0);
display.printf("T:%.1fC H:%.0f%%\n", temp, hum);
display.printf("P:%.0fhPa\n", pres);
display.printf("PM2.5:%d PM10:%d\n", pm25, pm10);
display.printf("MQ135:%d MQ7:%d\n", mq135_val, mq7_val);
display.printf("Gas:%.0f\n", gas_res / 1000.0);
display.printf("AQI: %d\n", aqi);
display.display();

delay(500); // 0.5 seconds delay  }
```

CHAPTER – 6
SYSTEM TESTING

System testing is a crucial phase in the development lifecycle to ensure the reliability, accuracy, and robustness of the IoT-based air quality monitoring system. It verifies the integration of all hardware, software, and cloud components, confirming the system behaves as expected under real-world conditions.

6.1 Unit Testing

Each individual hardware and software module was tested separately to ensure correct functioning. Examples include:

- **Sensor Testing:** BME680, PMS7003, MQ135, and MQ7 sensors were tested independently to confirm accurate readings.
- **OLED Display Module:** Verified that sensor data is displayed correctly in real-time.
- **Wi-Fi Connectivity:** NodeMCU was tested for stable connection to Firebase and reliable data upload.

6.2 Integration Testing

All components were tested together to evaluate their interoperability:

- Ensured sensor values are captured, processed, and transmitted by the ESP8266 microcontroller.
- Verified the OLED screen displays synchronized values with those in Firebase.
- Checked that the system maintains accurate data flow from hardware → microcontroller → cloud → web interface (if applicable).

6.3 Performance Testing

The system was evaluated for responsiveness, reliability, and scalability:

- **Response Time:** Data was successfully captured, processed, and uploaded to Firebase within ~2 seconds.
- **Stability:** The device was tested continuously for over 48 hours to monitor any crashes or data loss.
- **Data Throughput:** Firebase handled rapid data logging every 1–2 seconds without failures or lags.

6.4 Test Oracle

The system was tested in various real-life indoor environments such as bedrooms, kitchens, and offices:

- Identified changes in AQI when exposed to cooking fumes or incense sticks.

- Ensured that air quality warnings (based on AQI levels) were correctly categorized (Good, Moderate, Poor, Hazardous).
- Monitored how sensors reacted to variable temperature, humidity, and gas concentrations.

Test Case	Input	Expected Output
Sensor Initialization	Power ON system	All sensors initialize successfully; OLED shows "Initializing…"
Temperature Measurement	Room temperature ~28°C	OLED and Firebase display temperature around 28°C
Humidity Measurement	Humid environment (~70% RH)	Humidity value displays ~70% on OLED and Firebase
Gas Exposure (MQ135)	Hold burning incense near MQ135 sensor	MQ135 readings increase; AQI reflects higher pollution (e.g., AQI > 200)
Carbon Monoxide Detection (MQ7)	Expose sensor to cigarette smoke	MQ7 digital output triggered; OLED shows “Poor Air” or “Hazardous”
PM2.5/PM10 Detection (PMS7003)	Use aerosol spray near sensor	PM2.5 and PM10 values spike on OLED and Firebase
Wi-Fi Connectivity	Provide valid SSID and password	NodeMCU connects to Wi-Fi; data starts uploading to Firebase
Firebase Database Update	New sensor reading (e.g., AQI = 120)	Firebase gets updated in real-time with new data entry
OLED Display Update	Receive new sensor data every 1 second	OLED screen refreshes and shows updated environmental values every second
Flammable Gas Detection	Expose system to alcohol vapours	Flammable Gas value increases; reflected on OLED and Firebase
AQI Calculation Logic	Input gas resistance = 60,000 ohms	AQI falls in “Moderate” category (e.g., AQI = 100–200)
System Uptime Test	Continuous run for 48 hours	No crashes, data consistently logged, OLED and Firebase stay updated

Table: 2 Test Oracle

CHAPTER – 7

RESULTS AND ANALYSIS

7.1 Performance Evaluation

The IoT-based air quality monitoring system was evaluated across multiple dimensions, including data accuracy, system response, and user feedback. Below are the detailed performance insights:

7.1.1 Sensor and System Response

a. **Real-time Monitoring**: All sensors (BME680, PMS7003, MQ135, MQ7) provided updated readings at 1-second intervals.
b. **Data Upload Latency**: Data transfer to Firebase and display on OLED took less than 2 seconds on average.
c. **Data Accuracy**: Compared to standard references, values varied within ±5% for temperature, humidity, and particulate matter—acceptable for non-industrial environments.

7.1.2 System Stability and Reliability

a. The NodeMCU operated continuously for 48+ hours with no overheating or resets.
b. Wi-Fi reconnection and automatic resynchronization to Firebase were tested under intentional network drops and recovered successfully within 3–5 seconds.

7.1.3 User Experience

a. **Interface Simplicity**: The OLED screen provided clear, readable environmental data including AQI level, PM2.5, PM10, gas levels, temperature, and humidity.
b. **Alerts**: Users found the buzzer alerts helpful when pollutant levels crossed hazardous thresholds.
c. **Accessibility**: The data could be accessed through a web interface over Wi-Fi, making it easy to monitor remotely on smartphones and computers.
d. **Design & Portability**: The 3D-printed custom enclosure made the device compact, attractive, and easy to place in indoor environments.
e. **Ease of Setup**: Users with minimal technical background were able to connect the device to their network and view real-time data within minutes.

Overall, user feedback indicated a **positive experience** with the system being intuitive, fast, and informative.

7.2 Comparison with Existing Solutions

Existing air quality monitoring technologies, such as government stations and commercial devices, are often expensive, stationary, or limited in flexibility. Basic DIY solutions lack accuracy, real-time access, and scalability. Our IoT-based system overcomes these drawbacks by integrating affordable sensors (BME680, PMS7003, MQ135, MQ7), real-time Firebase connectivity, OLED/web display, and a custom 3D-printed enclosure. It provides a cost-effective, portable, and customizable solution with multi-sensor AQI calculation, live alerts, and cloud-based monitoring—making it ideal for homes, schools, and offices seeking reliable air quality data.

S. No	Existing Technologies	Drawbacks in Existing Technologies	Overcome (How our invention is overcoming the drawback)
1.	**Standalone Air Quality Monitors** (e.g., AirVisual Pro)	High cost and limited to specific parameters like PM2.5, VOCs, and CO2.	Provides a low-cost solution measuring multiple parameters (PM levels, VOCs, temperature, humidity, and harmful gases).
2	**Smart Home Systems** (e.g., Google Nest)	Locked into proprietary ecosystems, limited pollutant detection, and lack of detailed monitoring.	Open-source and modular, supports customization and additional sensor integration for tailored solutions.
3	**Industrial Air Quality Systems** (e.g., TSI DustTrak)	Bulky, expensive, and complex, designed for large-scale applications.	Compact, cost-effective, and user-friendly for homes, offices, and schools, suitable for personal and small-scale environments.

Table: 3 comparision with existing technologies

CHAPTER – 8

CONCLUSION AND FUTURE WORK

8.1 Conclusion

The IoT-based Air Quality Monitoring System developed in this project successfully meets the growing need for real-time environmental monitoring using an efficient, low-cost, and scalable approach. Leveraging the NodeMCU ESP8266 microcontroller along with various environmental sensors such as BME680 (for temperature, humidity, pressure, and gas resistance), PMS7003 (for PM2.5 and PM10 particulate matter), MQ135 (for air pollutants like ammonia, benzene), and MQ7 (for carbon monoxide), the system is capable of capturing and analyzing a wide range of air quality parameters. The use of an OLED display ensures immediate on-device feedback, while buzzer alerts notify users when air quality deteriorates. Furthermore, the integration with Google Firebase enables real-time data logging and cloud-based monitoring, making the system accessible from remote locations through web-based dashboards or mobile applications. A major highlight of the project is the inclusion of a custom 3D-printed enclosure, which not only protects the components but also enhances portability, usability, and aesthetics. The system's design prioritizes user interaction and awareness by mapping sensor values to a standardized AQI (Air Quality Index) scale and displaying the air quality category clearly. It is lightweight, responsive, and suitable for indoor environments such as homes, offices, and classrooms, contributing toward informed decisions about air quality and health.

8.2 Future Work

In future versions of the system, several enhancements can be incorporated to increase its effectiveness, scalability, and intelligence. These include:

8.2.1 **Location-Aware Monitoring:** By integrating GPS modules, the device can collect and tag air quality data based on geolocation, supporting city-wide mapping and pollution tracking.

8.2.2 **Mobile App Integration:** A companion mobile application can be developed to enable remote access to real-time data, push notifications for air quality alerts, and historical trend visualization.

8.2.3 **AI/ML-Based Prediction:** Implementing machine learning models will allow the system to forecast air quality trends and provide smart recommendations to users (e.g., when to ventilate a room).

8.2.4 **Outdoor and Solar-Powered Deployment:** To expand usage scenarios, the system can be upgraded for outdoor use with waterproof enclosures and solar panels for energy efficiency and sustainability.

8.2.5 **Advanced Cloud and Edge Support:** Migrating to more scalable databases (e.g., Firestore, AWS, or MySQL) and using edge computing can enhance processing speeds, reliability, and real-time responsiveness in larger networks of devices.

8.2.6 **Government and Community Integration:** With further development, the system can be scaled for use by municipal bodies or community groups to monitor air quality at a neighbourhood or city level, promoting environmental awareness and action.

These improvements aim to transform the current prototype into a more robust, intelligent, and widely deployable environmental monitoring platform that addresses modern air pollution challenges through technology-driven solutions.

CHAPTER – 9

PATENT WORK CERTIFICATES

We got our project patent published on IPR (Intellectual Property India) with application no. – 202511018266

Patent Search

Invention Title	MULTI-SENSOR SYSTEM FOR AIR QUALITY MONITORING AND DATA ANALYSIS
Publication Number	11/2025
Publication Date	14/03/2025
Publication Type	INA
Application Number	202511018266
Application Filing Date	02/03/2025
Priority Number	
Priority Country	
Priority Date	
Field Of Invention	BIO-MEDICAL ENGINEERING
Classification (IPC)	A61B0005149500, A61B0005000000, A63B0071060000, H05K0007200000, B05B0012140000

Inventor

Name	Address	Country
ARPNA SAXENA	ASSISTANT PROFESSOR, MASTER OF COMPUTER APPLICATION (MCA), AJAY KUMAR GARG ENGINEERING COLLEGE, 27TH KM MILESTONE, DELHI - MEERUT EXPY, GHAZIABAD, UTTAR PRADESH 201016	India
DR. SAROJ BALA	HOD, PROFESSOR, MASTER OF COMPUTER APPLICATION (MCA), AJAY KUMAR GARG ENGINEERING COLLEGE, 27TH KM MILESTONE, DELHI - MEERUT EXPY, GHAZIABAD, UTTAR PRADESH 201016	India
APURAV SINGH	MASTER OF COMPUTER APPLICATION (MCA), AJAY KUMAR GARG ENGINEERING COLLEGE, 27TH KM MILESTONE, DELHI - MEERUT EXPY, GHAZIABAD, UTTAR PRADESH 201016	India
AJEET YADAV	MASTER OF COMPUTER APPLICATION (MCA), AJAY KUMAR GARG ENGINEERING COLLEGE, 27TH KM MILESTONE, DELHI - MEERUT EXPY, GHAZIABAD, UTTAR PRADESH 201016	India
HARSH SAHAY	MASTER OF COMPUTER APPLICATION (MCA), AJAY KUMAR GARG ENGINEERING COLLEGE, 27TH KM MILESTONE, DELHI - MEERUT EXPY, GHAZIABAD, UTTAR PRADESH 201016	India
HARSH RAGHAV	MASTER OF COMPUTER APPLICATION (MCA), AJAY KUMAR GARG ENGINEERING COLLEGE, 27TH KM MILESTONE, DELHI - MEERUT EXPY, GHAZIABAD, UTTAR PRADESH 201016	India

Applicant

Name	Address	Country	
AJAY KUMAR GARG ENGINEERING COLLEGE	27TH KM MILESTONE, DELHI - MEERUT EXPY, GHAZIABAD, UTTAR PRADESH 201016	India	I

Abstract:

The present disclosure provides a system comprising a monitoring unit, a modulation assembly, a control assembly, and an adjustment unit. The monitoring unit con sensing assembly structured to obtain air quality parameters and a processing device arranged adjacent to the sensing assembly for data acquisition. The modulatio comprises a flow regulator disposed in a fluid passage and an actuator positioned in contact with the flow regulator to modulate airflow through the fluid passage. Th assembly comprises a signal interface affixed to the processing device and a command processor structured to generate actuation commands for the actuator basec quality parameters obtained by the sensing assembly. The adjustment unit comprises a directional component disposed within the fluid passage and a positioning el secured to the actuator for dynamic airflow modulation in response to the command processor.

Complete Specification

Description:Field of the Invention

The present disclosure generally relates to air quality management systems. Further, the present disclosure particularly relates to a system for monitoring and moc airflow based on air quality parameters.

Background

The background description includes information that may be useful in understanding the present invention. It is not an admission that any of the information prov herein is prior art or relevant to the presently claimed invention, or that any publication specifically or implicitly referenced is prior art.

Air quality management has gained significant importance due to increasing concerns regarding pollution and its impact on human health. Further, various air qual control techniques have been developed to regulate indoor and outdoor air quality. Moreover, various air treatment systems have been integrated into residential, commercial, and industrial environments to mitigate airborne contaminants. Such air treatment systems commonly include air purification devices, ventilation mechanisms, and filtration units.

Various air purification devices are commonly utilized to remove particulate matter, volatile organic compounds, and harmful gases from the surrounding environm Such air purification devices generally employ filtration mechanisms, ionization techniques, and chemical adsorption methods to purify the air. Further, such air purification devices are frequently integrated into HVAC (Heating, Ventilation, and Air Conditioning) systems to enhance indoor air quality. However, such air purific

View Application Status

CHAPTER – 10

REFERENCES

[1]. Ana, G.R.; Alli, A.S.; Uhiara, D.C.; Shendell, D.G. Indoor air quality and reported health symptoms among hair dressers in salons in Ibadan, Nigeria. J. Chem. Health Saf. 2019, 26, 23–30. [CrossRef]

[2]. Saini, J.; Dutta, M.; Marques, G. A comprehensive review on indoor air quality monitoring systems for enhanced public health. Sustain. Environ. Res. 2020, 30, 6. [CrossRef]

[3]. Kankaria, A.; Nongkynrih, B.; Gupta, S.K. Indoor air pollution in India: Implications on health and its control. Indian J. Community Med. 2014, 39, 203–207. [CrossRef]

[4]. Gola, M.; Settimo, G.; Capolongo, S. Indoor air in healing environments: Monitoring chemical pollution in inpatient rooms. Facilities 2019, 37, 600–623.

[5]. Patil, Prachu J., Ritika V. Zalke, Kalyani R. Tumasare, Bhavana A. Shiwankar, Shivani R. Singh, and Shailesh Sakhare. "IoT Protocol for Accident Spotting with Medical Facility", Journal of Artificial Intelligence 3, No. 02 (2021), 140-150.

[6] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, “Internet of Things for smart cities,” IEEE Internet Things J., vol. 1, no. 1, pp. 22–32, Feb. 2014.

[7]. Liu, W.; Shen, G.; Chen, Y.; Shen, H.; Huang, Y.; Li, T.; Wang, Y.; Fu, X.; Tao, S.; Liu, W.; et al., “Air pollution and inhalation exposure to particulate matter of different sizes in rural households using improved stoves in central China”,J. Environ. Sci., 2018, 63, 87–95.

[8].Fu, X.; Tao, S.; Liu, W.; et al., “Air pollution and inhalation exposureto particulate matter of different sizes in rural households usingimproved stoves in central China”,J. Environ. Sci., 2018, 63, 87–95.

[9]. Jo, J.H.; Jo, B.W.; Kim, J.H.; Kim, S.J.; Han, W.Y., “Development ofan IoT-Based Indoor Air Quality Monitoring Platform”, J. Sens.Hindawi 2020, 1–14.

[10]. Marques, G.; Pitarma, R., “Non-contact Infrared Temperature Acquisition System based on Internet of Things for Laboratory Activities Monitoring”, Procedia Comput. Sci., 2019, 155, 487–494

[11] Cheng, Y., & Zhang, Y. (2020). Development of a smart indoor air quality monitoring system based on IoT. IEEE Access, 8, 1-10. DOI: 10.1109/ACCESS.2020.2991234

[12]. Gonzalez, A., & Garcia, J. (2020). IoT-based air quality monitoring system for smart cities. Journal of Ambient Intelligence and Humanized Computing, 11(3), 1-15. DOI: 10.1007/s12652-019-01373-5

[13]. Liu, Y., & Zhang, Y. (2021). A comprehensive review of indoor air quality monitoring systems. Environmental Science and Pollution Research, 28(12), 1-15. DOI: 10.1007/s11356-021-12678-0

[14]. Alavi, A., & Khosravi, A. (2020). Development of a low-cost air quality monitoring system using IoT. Sensors, 20(3), 1-15. DOI: 10.3390/s20030800

[15]. Z. Li, X. Tong, J. M. W. Ho, T. C. Kwok, G. Dong, K. F. Ho and S. H. L. Yim "A practical framework for predicting residential indoor PM2. 5 concentration using land-use regression and machine learning methods." Chemosphere, 2021, , 265, 129140.

[16]. Sonawani, Shilpa; PATIL, Kailas (2022), “Dataset of Indoor Air Pollutants using Low-Cost Sensors”, Mendeley Data, V1, doi: 10.17632/2r232jpfb2.1

CHAPTER –11

APPENDICES

Appendix 1 – Component Specifications

Component	Description
NodeMCU ESP8266	Wi-Fi-enabled microcontroller used for reading sensors and sending data online
BME680 Sensor	Measures temperature, humidity, pressure, and gas resistance
PMS7003 Sensor	Detects PM2.5 and PM10 levels (particulate matter)
MQ-135 Sensor	Detects gases such as NH3, NOx, alcohol, benzene, smoke
MQ-7 Sensor	Detects carbon monoxide (CO)
OLED Display (0.96")	Displays real-time sensor values and AQI
Buzzer	Alerts when air quality crosses safe threshold
3D Printed Box	Custom enclosure designed for safe housing and portability
Jumper Wires	Used for making electrical connections
Breadboard	Used during prototyping phase for circuit assembly

Table: 4 Component Specification

Appendix 2: Arduino pin Configuration

Sensor/Module	Connected To (NodeMCU)
BME680 (I2C)	SDA → D2 (GPIO4), SCL → D1 (GPIO5)
PMS7003	RX → D7 (GPIO13), TX → D6 (GPIO12), VCC → VIN
MQ-135 (Analog)	AO → A0
MQ-7 (Digital)	DOUT → D5 (GPIO14)
OLED Display	SDA → D2 (GPIO4), SCL → D1 (GPIO5)
Buzzer	GPIO → D3 or D4 (configurable), VCC → 3.3V

Table: 5 Arduino pin configuration

Appendix 3: System Output Samples

OLED display output

Temp: 25.4°C
Hum: 60%
PM2.5: 42
PM10: 85
AQI: 120 (Moderate)

Arduino IDE serial monitor output

Temperature: 25.4°C
Humidity: 60.2%
Pressure: 1005 hPa
PM2.5: 42 µg/m3
PM10: 85 µg/m3
MQ135 Value: 350
MQ7 Status: HIGH
Flammable Gas: 500
AQI: 120 - Moderate
Data sent to Firebase

Appendix 4: 3D Printed Box Design

- **Material Used:** PLA (Polylactic Acid)
- **Design Software:** Fusion360
- **Printer Used:** Ender 3

Features:

- Vents for airflow and accurate sensing
- Holes for sensor visibility (e.g., PMS7003 air inlet)
- Mounting slots for OLED and power cable access
- Compact and aesthetic housing for portable indoor placement