# IoT ENABLED AIR QUALITY MONITORING SYSTEM

**A MINI PROJECT REPORT**

***Submitted by***

**ARUN KARTHICK N (910621103011)**
**DEENA THAYALAN A (910621103016)**

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

***in***

**ELECTRONICS AND COMMUNICATION ENGINEERING**

**K.L.N. COLLEGE OF ENGINEERING**

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

**MAY 2024**

## K.L.N. COLLEGE OF ENGINEERING

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

## BONAFIDE CERTIFICATE

Certified that this report titled “ **IoT ENABLED AIR QUALITY MONITORING SYSTEM ”**, is the Bonafide work of **“ ARUN KARTHICK N ( 910621103011), DEENA THAYALAN A ( 910621103016 )** who carried out the mini project work under my supervision.

| **SIGNATURE** | **SIGNATURE** |
|---|---|
| Dr. V. KEJALAKSHMI, M.E., Ph.D., | Mrs. R. ANGAYARKANNI, M.E |
| Professor | Assistant Professor |
| **HEAD OF THE DEPARTMENT** | **SUPERVISOR** |
| Department of Electronics and | Department of Electronic and |
| Communication Engineering, | Communication Engineering, |
| K.L.N College of Engineering, | K.L.N College of Engineering, |
| Pottapalayam, | Pottapalayam, |
| Sivagangai-630 611. | Sivagangai-630 611. |

Submitted for the mini project viva-voce conducted on ___________________

**INTERNAL EXAMINER** **EXTERNAL EXAMINER**

# ACKNOWLEDGEMENT

We express our deepest gratitude to our esteemed Founder, Late. **Thiru. K.L.N. KRISHNAN**, K.L.N. College of Engineering, for making us march towards the glory of success.

We extend heartfelt thanks to our **President, Er. K.N.K.Karthik B.E.** and **Secretary** Dr.**K.N.K.Ganesh B.E.,Ph.D (Hons),**for their invaluable support and opportunities they have provided.

We also express our sincere thanks to our respected **Principal Dr.A.V. RAM PRASAD M.E.,Ph.D.,MISTE.FIE**, for all the facilities offered.Their belief in our endeavors and willingness to open doors have been instrumental in our personal and professional growth.

We would like to express our profound gratitude and heartfelt thanks to

**Dr.V. KEJA LAKSHMI, M.E., Ph.D.,** Head of the Department of Electronics and Communication Engineering, who motivated and encouraged us to do this outrival project for this academic year.

Our delightful and sincere thanks to our respected Project Coordinator **Mrs.R.Angayarkanni M.E** Assistant Professor, whose support was inevitable during the entire period of our work.

We thank our teaching staffs for sharing their knowledge and view to enhance our project. We also thank our non-teaching staff for extending their technical support to us. I thank my parents for giving me such a wonderful life and my friends for their friendly encouragement throughout the project.

Finally, we thank the Almighty for giving the full health to finish the project successfully.

# ABSTRACT

Air pollution is one of the biggest threats to the present-day environment. Everyone is being affected by air pollution day by day including humans, animals, crops, cities, forests and aquatic ecosystems. Besides that, it should be controlled at a certain level to prevent the increasing rate of global warming. The "IoT Enabled Air Quality Monitoring System" presents a sophisticated solution to the pressing concern of environmental air quality through the integration of cutting-edge Internet of Things (IoT) technology. Central to the system is the NodeMCU v3 ESP8266 microcontroller, orchestrating a network of sensors including the MQ135 gas sensor and the DHT11 temperature and humidity sensor. This amalgamation enables continuous surveillance of ambient air, discerning concentrations of carbon dioxide, carbon monoxide, alcohol, and other discernible gases, while concurrently capturing crucial atmospheric parameters such as temperature and humidity.

The acquired data is transmitted to the ThingSpeak platform, where it undergoes real-time analysis, visualization, and archiving, thus furnishing stakeholders with invaluable insights into temporal and spatial variations in air quality. Furthermore, leveraging Twilio's cloud-based communication infrastructure, the system promptly dispatches alert calls to designated recipients upon the detection of hazardous conditions surpassing pre-established thresholds. This proactive feature not only ensures user safety but also facilitates timely intervention and mitigation efforts. The system's user interface employs intuitive visual indicators, manifested through tri-color LEDs, to convey air quality levels, with green signifying optimal conditions, yellow indicating moderate concerns, and red denoting critical levels warranting immediate attention. In parallel, an audible alarm mechanism, facilitated by a buzzer, supplements the visual cues, enhancing situational awareness.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| DHT | Digital Humidity and Temperature |
| IoT | Internet of Things |
| PPM | Parts Per Molecule |
| PM | Particulate Matter |
| CO | Carbon Monoxide |
| CO2 | Carbon Dioxide |
| LED | Light Emitting Diode |
| LPG | Liquid Petroleum Gas |
| IDE | Integrated Development Environment |

# CHAPTER 1

# INTRODUCTION

## 1.1 BACKGROUND

Air quality has emerged as a critical global concern due to its profound implications for public health and environmental sustainability. Rapid industrialization, urbanization, and vehicular emissions have exacerbated pollution levels, necessitating innovative solutions for real-time monitoring and mitigation. In response to this challenge, the "IoT Enabled Air Quality Monitoring System" harnesses the power of Internet of Things (IoT) technology to provide a comprehensive solution for monitoring ambient air quality parameters.

## 1.2 OBJECTIVES

The primary objective of this project is to develop an advanced air quality monitoring system capable of detecting and analyzing harmful gases, such as carbon dioxide, carbon monoxide, and alcohol, in real-time. Additionally, the system aims to monitor temperature and humidity levels to provide a holistic understanding of environmental conditions. Through integration with cloud-based platforms like ThingSpeak and Twilio, the system facilitates remote monitoring and alerting, enabling timely intervention in response to deteriorating air quality conditions. Furthermore, the project seeks to enhance user awareness and engagement through intuitive visual and auditory feedback mechanisms.

## 1.3 SCOPE OF THE PROJECT

The scope of the project encompasses the design, implementation, and validation of an IoT-enabled air quality monitoring system using readily available hardware components and open-source software tools. Key components include the NodeMCU v3 ESP8266 microcontroller, MQ135 gas sensor, DHT11 temperature and humidity sensor, LEDs, and a buzzer. The system architecture incorporates sensor integration, data transmission to cloud platforms, alerting mechanisms, and user interface design.

The project focuses on achieving the following milestones:

1. Hardware configuration and sensor calibration to ensure accurate data acquisition.
2. Software development using Arduino IDE for programming the NodeMCU microcontroller and interfacing with sensors.
3. Integration with ThingSpeak for real-time data visualization, monitoring, and analysis.
4. Integration with Twilio for cloud-based communication and alerting.
5. Implementation of visual indicators (LEDs) and auditory alerts (buzzer) for user feedback.
6. Validation of system functionality through simulated scenarios and field testing in diverse environmental conditions.
7. By accomplishing these objectives, the project aims to deliver a robust, scalable, and user-friendly air quality monitoring solution that can be deployed in various settings, including homes, offices, industrial facilities, and urban areas.

# CHAPTER 2

# LITERATURE SURVEY

## 1. IOT-ENABLED AIR QUALITY MONITORING SYSTEMS:

**AUTHORS:** Chen, Q., Li, H., et al.

**PUBLISHED IN:** Environmental Science and Pollution Research, 2019.

**SUMMARY:** This study examines the design and implementation of IoT-enabled air quality monitoring systems. It discusses sensor technologies, including MQ135 and DHT11, used for detecting pollutants and environmental parameters. The paper explores cloud-based data management platforms like ThingSpeak for real-time monitoring and analysis. Additionally, it evaluates alerting mechanisms utilizing communication services such as Twilio to notify users of hazardous air quality conditions.

## 2. SENSOR TECHNOLOGIES FOR AIR QUALITY MONITORING:

**AUTHORS:** Kumar, S., Gupta, R., et al.

**PUBLISHED IN:** Sensors and Actuators B: Chemical, 2018.

**SUMMARY:** This research investigates the performance of various sensor technologies, including MQ135, in air quality monitoring applications. It evaluates sensor accuracy, sensitivity, and response time for detecting gases like carbon dioxide and carbon monoxide. The paper discusses calibration procedures and challenges associated with integrating multiple sensors into IoT-enabled monitoring systems.

## 3. CLOUD-BASED DATA MANAGEMENT AND ANALYSIS IN ENVIRONMENTAL SENSING:

**AUTHORS:** Zhang, L., Wang, Y., et al.

**PUBLISHED IN:** Journal of Cleaner Production, 2021.

**SUMMARY:** This paper explores the integration of cloud-based platforms like ThingSpeak for managing and analyzing environmental sensor data. It discusses data visualization techniques and analytical tools for identifying air quality trends and anomalies. The study highlights the scalability and accessibility of cloud-based solutions in supporting large-scale environmental monitoring initiatives.

## 4. ALERTING MECHANISMS AND USER INTERFACE DESIGN IN IOT APPLICATIONS:

**AUTHORS:** Lee, C., Wang, S., et al.

**PUBLISHED IN:** Computers & Industrial Engineering, 2020.

**SUMMARY:** This research investigates effective alerting mechanisms and user interface design principles for IoT applications, including air quality monitoring systems. It examines the use of visual indicators, such as LEDs, and audible alarms to communicate air quality information to users.

## 5. INTEGRATION OF COMMUNICATION PLATFORMS FOR REAL TIME ALERTING:

**AUTHORS:** Liu, Y., Zhou, H., et al.

**PUBLISHED IN:** IEEE Transactions on Industrial Informatics, 2019.

**SUMMARY:** This study explores the integration of communication platforms like Twilio for real-time alerting in IoT applications. It discusses the use of SMS notifications and cloud-based calling services to notify users of critical events, such as deteriorating air quality conditions. The paper evaluates the reliability and scalability of Twilio-based alerting systems in supporting mission-critical applications.

# CHAPTER 3

# HARDWARE OVERVIEW

## 3.1 ARCHITECTURE OVERVIEW

The IoT Enabled Air Quality Monitoring System embodies a sophisticated architecture designed to seamlessly integrate hardware components, software tools, and cloud-based platforms. At its core lies the NodeMCU v3 ESP8266 microcontroller, serving as the central processing unit orchestrating data acquisition, transmission, and analysis. This microcontroller interfaces with a suite of sensors, including the MQ135 gas sensor for detecting harmful gases and the DHT11 sensor for monitoring temperature and humidity levels. These sensors are strategically deployed within the monitoring environment to capture comprehensive environmental data.

## 3.2 COMPONENTS AND HARDWARE SETUP

The hardware setup comprises a meticulously curated selection of components tailored to meet the stringent requirements of air quality monitoring. The NodeMCU v3 ESP8266 microcontroller serves as the backbone of the system, interfacing with the MQ135 and DHT11 sensors to capture vital air quality metrics. Additionally, visual indicators in the form of red, yellow, and green LEDs convey air quality levels, while a buzzer provides auditory alerts in extreme conditions. These components are intricately connected and configured to ensure optimal performance and reliability in diverse environmental conditions.

## 3.2.1 HARDWARE COMPONENTS USED

1) NodeMCU V3 ESP8266
2) DHT11 Sensor Module
3) MQ-135 Gas Sensor Module
4) Buzzer
5) LED (Red, Green & Yellow)
6) Breadboard
7) Connecting Wires

### 3.2.1.1 NODEMCU V3 ESP8266

NodeMCU V3 is an open-source ESP8266 development kit, armed with the CH340G USBTTL Serial chip. It has firmware that runs on ESP8266 Wi-Fi SoC from Espressif Systems. Whilst cheaper, CH340 is super reliable even in industrial applications. It is tested to be stable on all supported platforms as well. It can be simply coded in Arduino IDE. It has a very low current consumption between 15 µA to 400 mA.

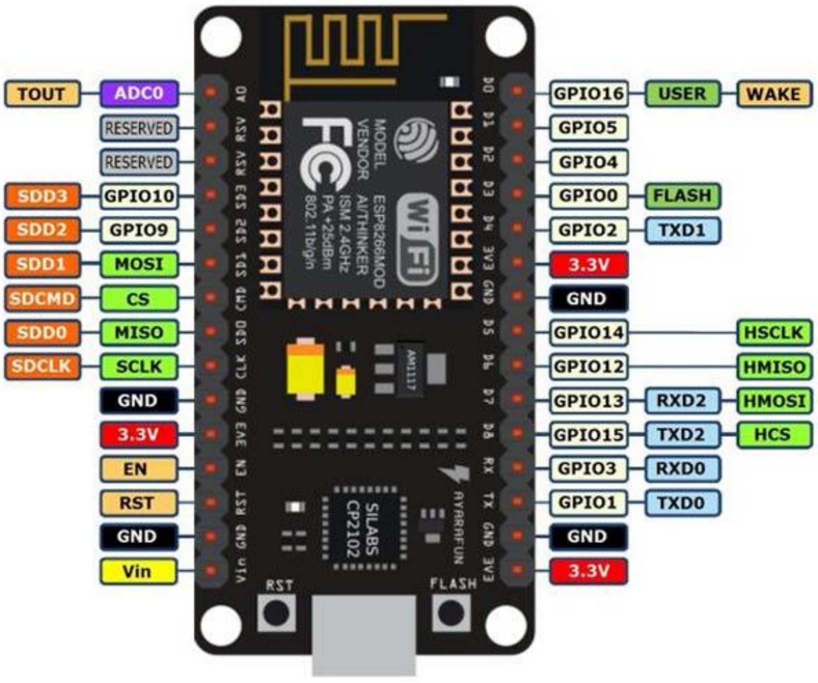


Figure 3.1 (Pinout Diagram of NodeMCU V3 ESP8266)

### 3.2.1.2 DHT11 TEMPERATURE SENSOR

The DHT11 is a temperature and humidity sensor that gives digital output in terms of voltage. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air. As shown in Fig. 3.2, we need to supply a voltage of 5V (DC) to the Vcc pin and ground it to the GND pin. The sensor output can be easily read from the Data pin in terms of voltage (in digital mode). Humidity Measurement: The humidity sensing capacitor has two electrodes with a moisture-holding substrate as a dielectric between them as shown in Fig 3.2. Change in the capacitance value occurs with the change in humidity levels. The IC measure, process these changed resistance values and then converts them into digital form. Temperature Measurement: For measuring the temperature, the DHT11 sensor uses a negative temperature coefficient thermistor, which causes a decrease in its resistance value with an increase in temperature. To get a wide range of resistance values, the sensor is made up of semiconductor ceramics or polymers.

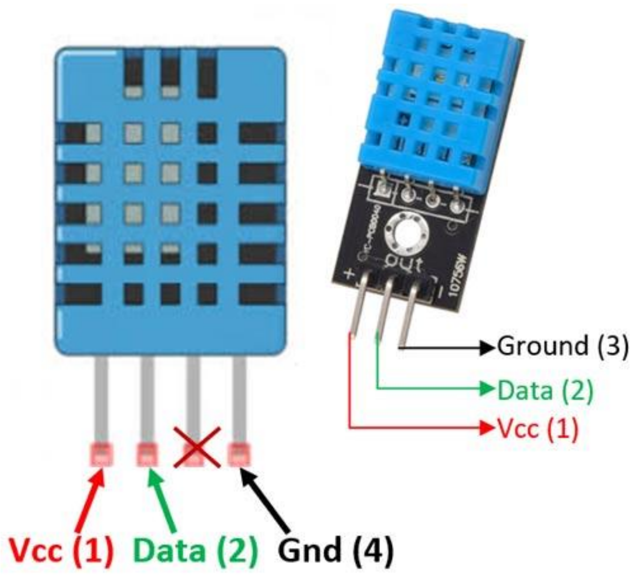


Figure 3.2 (Pinout Diagram of DHT11sensor)

### 3.2.1.3 MQ135 GAS SENSOR

The material of MQ135 is SnO2, it is a special material, when exposed to clean air, it is hardly being conducted, however, when put in an environment with combustible gas, it has a pretty performance of conductivity. Just make a simple electronic circuit, and convert the change of conductivity to a corresponding output signal. MQ135 gas sensor is sensitive to Ammonia, Sulphide, Benzene steam, smoke and other harmful gases. Used for family, surrounding environment noxious gas detection device, apply to ammonia, aromatics, sulphur, benzene vapor, and other harmful gases/smoke, gas detection, tested concentration range: 10 to 1000ppm. In a normal environment, the environment which doesn't have detected gas set the sensor's output voltage as the reference voltage, the analog output voltage will be about 1V, when the sensor detects gas, harmful gas concentration increases by 20ppm per voltage increase by 0.1V.

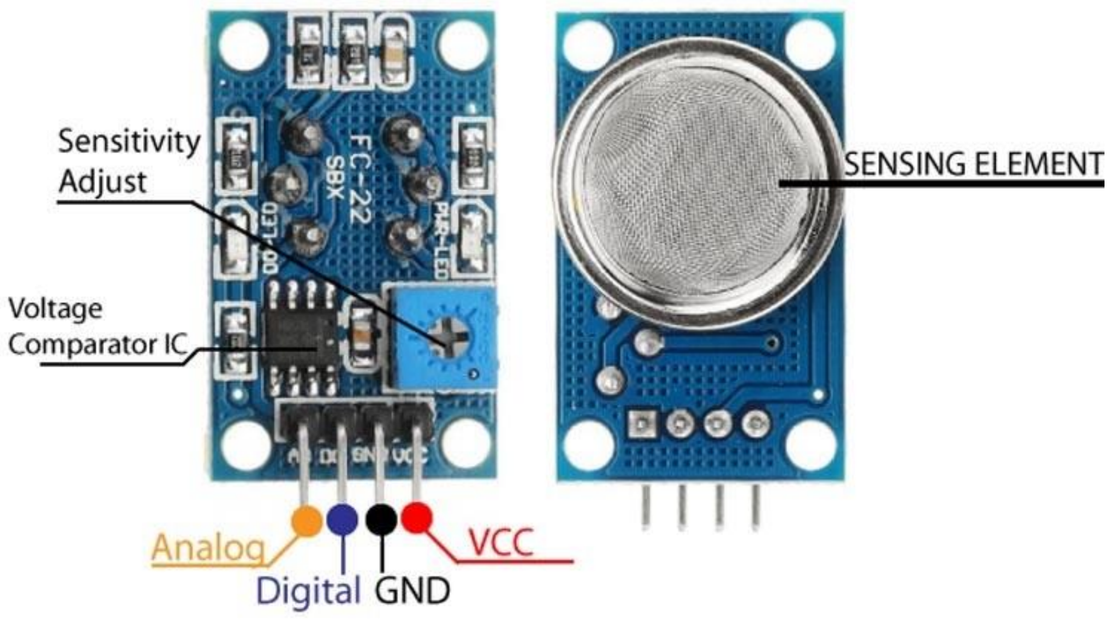


Figure 3.3 ( MQ-135 Gas Sensor )

### 3.2.1.4 BUZZER

A buzzer is an electromechanical device commonly used in electronic circuits to generate audible alerts or signals. It operates by converting electrical energy into mechanical vibrations, which in turn produce sound waves. Inside the buzzer, there is typically a piezoelectric element or a magnetic coil attached to a diaphragm. When an electrical signal is applied to the buzzer, it causes the element or coil to vibrate rapidly, creating sound waves that are audible to the human ear. Buzzer's sound can vary in frequency and intensity depending on the design and construction of the device.

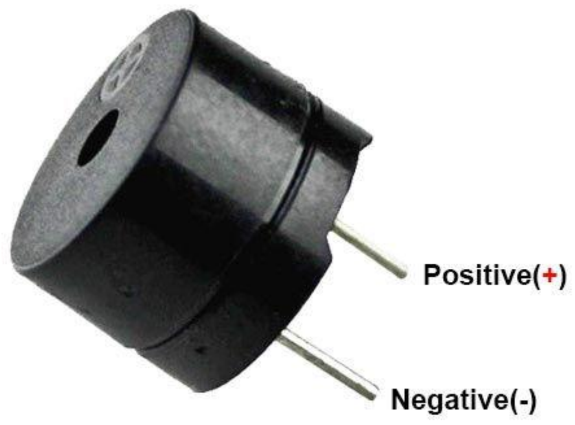


Figure 3.4 ( Buzzer )

### 3.2.1.5 LED (RED, GREEN & YELLOW)

Light Emitting Diode is a semiconductor device that emits light when an electric current passes through it. It consists of a semiconductor chip mounted within a transparent or colored plastic casing, with two leads that conduct electricity into the chip. When electrons recombine with electron holes within the semiconductor material, energy is released in the form of photons, producing light.

# CHAPTER 4

# SOFTWARE AND PLATFORMS

## 4.1 SOFTWARE TOOLS AND PLATFORMS

The software infrastructure of the system is underpinned by the versatile Arduino IDE, providing a robust development environment for programming the NodeMCU microcontroller. This platform facilitates the implementation of sensor data processing algorithms, threshold monitoring logic, and communication protocols. Furthermore, cloud-based platforms such as ThingSpeak and Twilio play pivotal roles in enabling real-time data monitoring, analysis, and alerting. ThingSpeak serves as the conduit for transmitting sensor data to the cloud, where it is visualized, analyzed, and archived in real-time. Meanwhile, Twilio facilitates cloud-based communication, enabling the system to dispatch alert calls to users when predetermined air quality thresholds are exceeded.

## 4.2 SOFTWARE AND PLATFORMS USED

1) Arduino IDE
2) ThingSpeak
3) ThingHTTP
4) React
5) Twilio

### 4.2.1 ARDUINO IDE

The Arduino IDE is open-source software, which is used to write and upload code to the Arduino boards. The IDE application is suitable for different operating systems such as Windows, Mac OS X, and Linux. It supports the programming languages C and C++. Here, IDE stands for Integrated Development Environment. The program or code written in the Arduino IDE is often called sketching. We need to connect the Genuino and Arduino board with the IDE to upload the sketch written in the Arduino IDE software. The sketch is saved with the extension '.ino.'

### 4.2.2 THINGSPEAK

ThingSpeak is open-source software written in Ruby which allows users to communicate with internet-enabled devices. It facilitates data access, retrieval and logging of data by providing an API to both the devices and social network websites. ThingSpeak was originally launched by ioBridge in 2010 as a service in support of IoT applications. ThingSpeak has integrated support from the numerical computing software MATLAB from MathWorks, allowing ThingSpeak users to analyse and visualize uploaded data using MATLAB without requiring the purchase of a MATLAB license from MathWorks.

### 4.2.3 THINGHTTP

ThingHTTP enhances the versatility and connectivity of IoT applications, enabling seamless integration and communication between IoT devices and external web services. Its custom HTTP request and response capabilities, event triggering mechanisms, and integration with ThingSpeak empower users to create sophisticated and interactive IoT solutions, capable of meeting diverse application requirements and use cases.

### 4.2.4 REACT

React is a feature that allows users to define custom actions or responses based on the data received from IoT devices or other sources. React enables users to create dynamic and automated workflows by setting up rules that trigger specific actions when certain conditions are met. These actions can include sending notifications, updating data visualizations, or even controlling IoT devices in real-time. With React, users can define rules using simple conditional statements based on data from ThingSpeak channels. For example, users can set up a React rule to send an email notification when the temperature in a specific location exceeds a certain threshold, or to turn on a smart device when a motion sensor detects movement. React provides a flexible and intuitive interface for creating and managing these rules, allowing users to customize their IoT applications according to their specific needs and preferences.

### 4.2.5 TWILIO

Twilio is a cloud communications platform that enables developers to integrate various communication channels, including voice calls, SMS, and messaging, into their applications and services. With Twilio, developers can easily incorporate features such as sending and receiving text messages, making and receiving phone calls, and even implementing two-factor authentication mechanisms. One of the key benefits of Twilio is its simplicity and scalability, allowing developers to quickly and efficiently add communication functionality to their applications without the need for complex infrastructure or specialized hardware. Twilio's API provides a straightforward interface for sending and receiving messages and managing phone calls, making it accessible to developers of all skill levels.

# CHAPTER 5

# SYSTEM ARCHITECTURE OVERVIEW

## 5.1 SYSTEM ARCHITECTURE OVERVIEW

The architecture of our Air Quality Monitoring System is designed to facilitate real-time data acquisition, processing, and transmission, enabling users to monitor air quality remotely. The system comprises interconnected hardware components, including sensors, microcontrollers, and indicators, which work in tandem to collect environmental data, analyze it, and provide alerts when necessary. The data flow between these components is orchestrated through predefined communication protocols, ensuring seamless integration and operation of the system.

## 5.2 HARDWARE COMPONENTS INTERCONNECTIONS

1. **SENSORS:** The system utilizes sensors such as the MQ135 gas sensor and DHT11 temperature and humidity sensor to collect air quality data. These sensors are connected to the NodeMCU v3 ESP8266 microcontroller via digital or analog interfaces, allowing for real-time data acquisition.
2. **NODEMCU V3 ESP8266 MICROCONTROLLER:** Serving as the central processing unit, the NodeMCU v3 ESP8266 microcontroller processes the sensor data and controls the operation of other hardware components. It is responsible for collecting sensor readings, analyzing them, and triggering alerts when predefined thresholds are exceeded.
3. **INDICATORS:** Visual indicators in the form of LEDs (red, yellow, green) and an audible buzzer are used to convey air quality status to users. These indicators are controlled by the NodeMCU microcontroller and are activated based on the analyzed sensor data.

## 5.3 DATA FLOW

1) **DATA ACQUISITION:**

Sensor data, including air quality measurements from the MQ135 sensor and temperature/humidity readings from the DHT11 sensor, is acquired by the NodeMCU microcontroller at regular intervals.

2) **DATA PROCESSING:**

The acquired sensor data is processed by the microcontroller to calculate relevant air quality metrics and determine the current air quality status. This processing involves applying predefined algorithms and threshold values to the sensor readings.

3) **ALERT GENERATION:**

Based on the processed data, the microcontroller triggers alerts when predefined thresholds for air quality parameters are exceeded. These alerts can include activating visual indicators (LEDs) and/or an audible buzzer to notify users of poor air quality conditions.

4) **DATA TRANSMISSION:**

Processed sensor data, along with alert notifications, is transmitted from the NodeMCU microcontroller to cloud-based platforms such as ThingSpeak and Twilio. This data transmission is achieved using communication protocols such as HTTP and MQTT, ensuring secure and reliable communication with external services.

This architecture ensures the seamless operation of our Air Quality Monitoring System, enabling users to monitor and respond to air quality conditions effectively, both locally and remotely.

# CHAPTER 6

# SENSOR PREHEATING AND CALIBRATION

## 6.1 SENSOR PREHEATING

Sensor preheating involves heating the sensor element to a stable operating temperature before taking measurements. This is particularly important for gas sensors like the MQ135, which require a certain temperature to achieve optimal sensitivity and accuracy. Preheating the sensor ensures that it reaches a steady state, reducing the impact of temperature fluctuations on sensor readings.

The preheating process typically involves applying power to the sensor for a certain duration before taking measurements. This allows the sensor to stabilize and reach its operating temperature, ensuring consistent and reliable performance over time. In our system, we incorporate a preheating period at startup or during initialization to ensure that the sensors are operating under optimal conditions before data collection begins.

## 6.2 SENSOR CALIBRATION

Sensor calibration is the process of adjusting sensor readings to align with known reference values, ensuring accuracy and reliability. Calibration is essential for compensating for variations in sensor performance due to factors such as environmental conditions, aging, and manufacturing tolerances. The calibration process involves comparing sensor readings to reference standards or calibrated instruments and making adjustments to correct any deviations. This typically requires exposing the sensor to known concentrations of target gases or calibration gases in a controlled environment. By comparing the sensor's response to the reference values, calibration coefficients or correction factors can be determined and applied to adjust the sensor's output.

## 6.3 PREHEATING OF DHT11 SENSOR

The following steps were performed to preheat the DHT11 sensor module:

**STEP 1 :** The Vcc pin of the DHT11 sensor module was connected with the 3.3V pin of NodeMCU.

**STEP 2 :** The Gnd pin of the DHT11 sensor module was connected with the Gnd pin of NodeMCU.

**STEP 3 :** The NodeMCU is powered for 20 minutes.

**STEP 4 :** The setup was then disconnected.

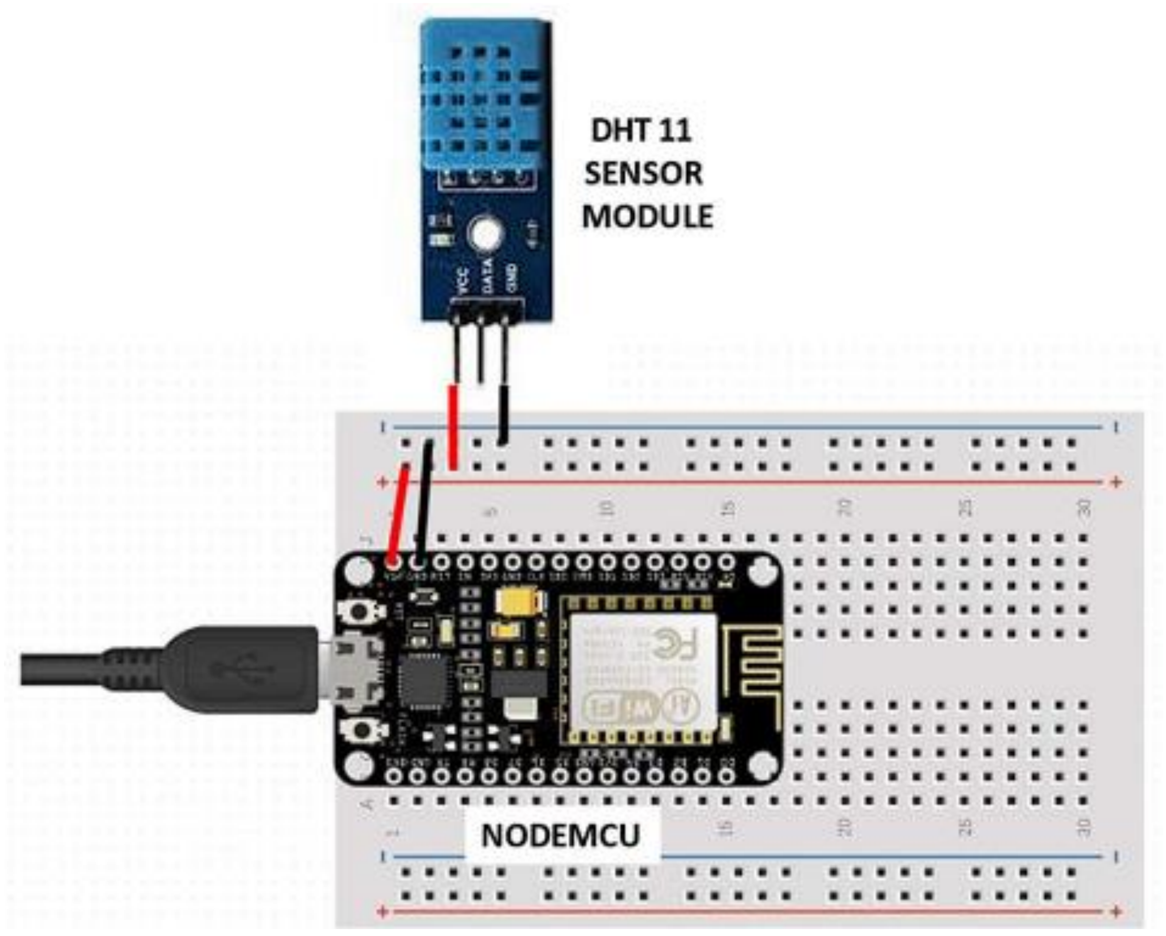


Figure 6.1 (Circuit Diagram to Preheat the DHT11 sensor)

## 6.4 PREHEATING OF MQ-135 GAS SENSOR

The following steps were performed to preheat the MQ-135 gas sensor module

**STEP 1 :** The Vcc pin of the MQ-135 gas sensor module was connected with the 3.3V pin of NodeMCU.

**STEP 2 :** The Gnd pin of the MQ-135 gas sensor module was connected with the Gnd pin of NodeMCU.

**STEP 3 :** The NodeMCU is powered for a day.

**STEP 4 :** The setup was then disconnected.

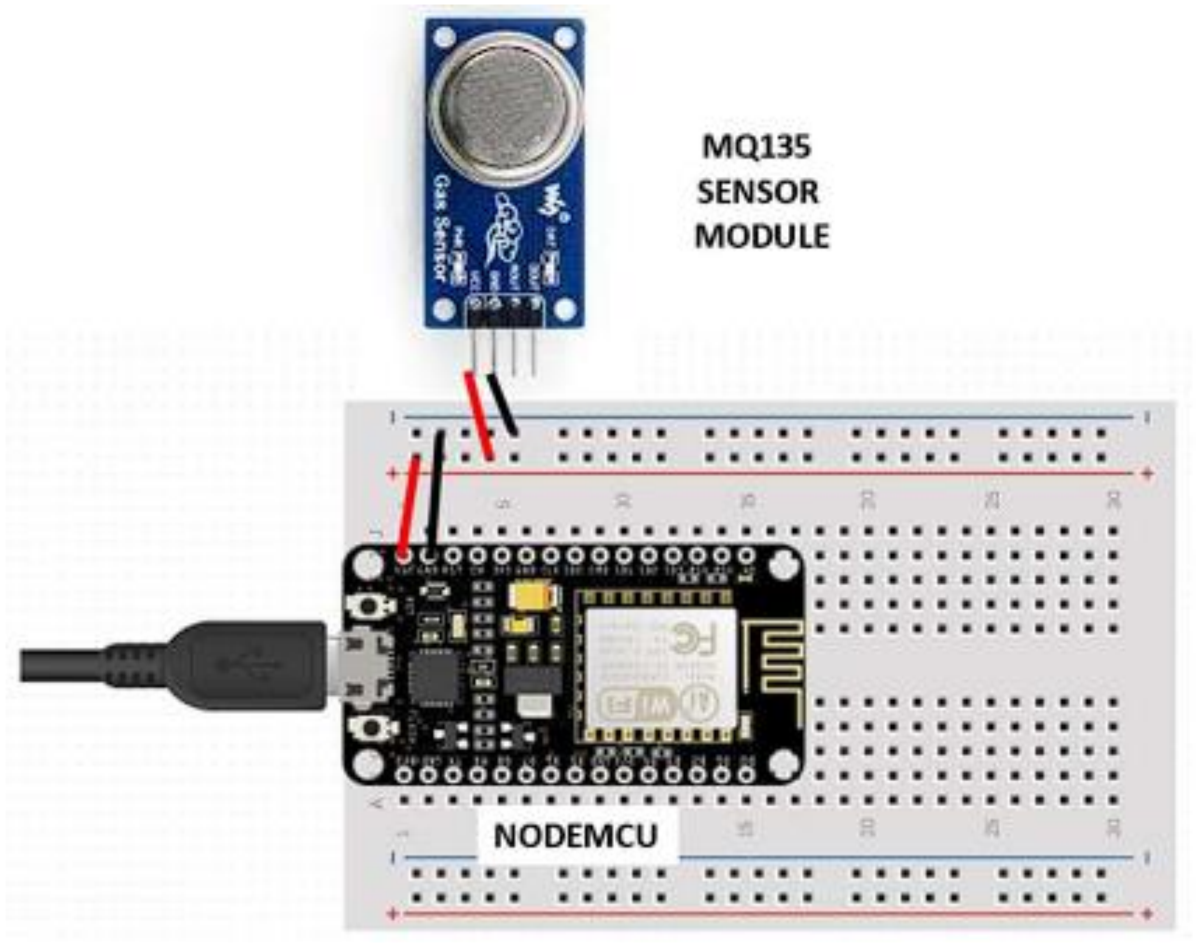


Figure 6.2 (Circuit Diagram to Preheat the MQ135 Gas sensor)

## 6.5 CALIBRATION OF MQ-135 GAS SENSOR MODULE

### 6.5.1 THEORY OF CALIBRATION

The most important step is to calibrate the sensor in the fresh air and then draw an equation that converts the sensor output voltage value into our convenient units PPM (parts per million). Here are the mathematical calculations derived,

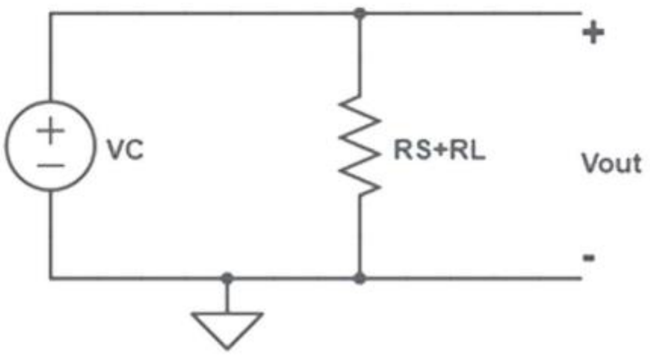


Figure 6.3 (Internal Circuit diagram of MQ-135 sensor)

From Ohm's Law, at a constant temperature, we can derive I as follows:

$$I = V/R \qquad (1)$$

From Figure 6.3, eqn. 1 is equivalent to

$$I = VC/RS+RL \qquad (2)$$

From Figure 6.3, we can obtain the output voltage at the load resistor using the value obtained for I and Ohm's Law at a constant temperature,

$$V = I \times R$$

$$VRL=[VC/(RS+RL)]RL \qquad (3)$$

$$VRL = [(VC* RL)/(RS+RL)] \qquad (4)$$

So now we solve for RS:

$$VRL \times (RS + RL) = VC \times RL \qquad (5)$$

$$(VRL \times RS) + (VRL \times RL) = VC \times RL \qquad (6)$$

$$VRL \times RS = (VC * RL) - (VRL * RL) \qquad (7)$$

$$RS = \{(VC * RL - (VRL * RL)\} / VRL \qquad (8)$$

$$RS = \{(VC * RL)\ VRL\} - RL \qquad (9)$$

Eqn. 9 helps us to find the internal sensor resistance for fresh air.

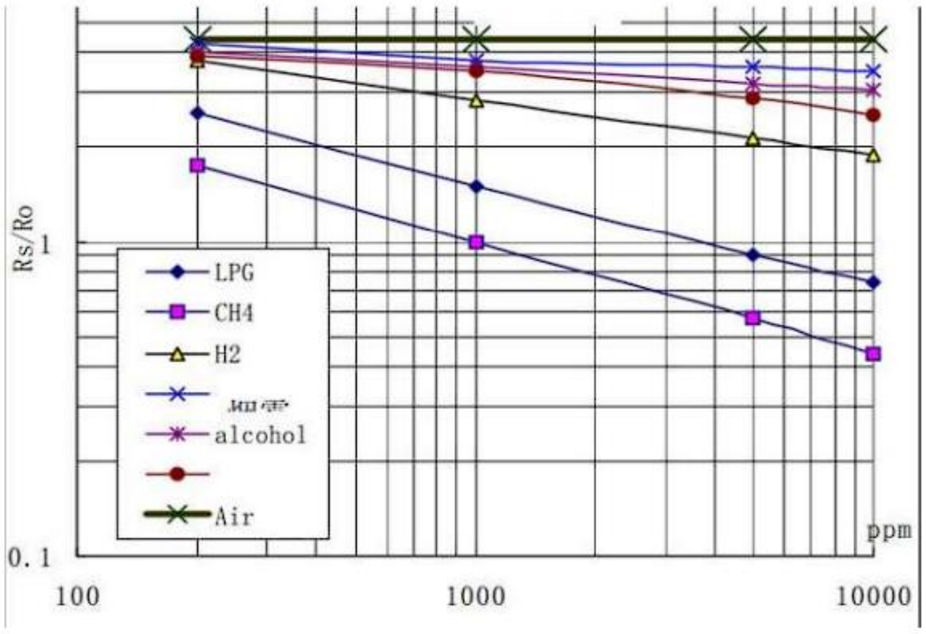


Figure 6.4 (Graph representing ratio vs ppm variations)

From the graph shown in fig 6.4, we can see that the resistance ratio in fresh air is a constant:

$$RS / R0 = 3.6 \qquad (10)$$

Value 3.6 which is mentioned in eqn. 10 is depicted in the datasheet shown in Figure 6.4. To calculate R0, we will need to find the value of the RS in the fresh air. This will be done by taking the analog average readings from the sensor and converting them to voltage. Then we will use the RS formula to find R0. First of all, we will treat the lines as if they were linear. This way we can use one formula that linearly relates the ratio and the concentration. By doing so, we can find the concentration of a gas at any ratio value even outside of the graph's boundaries. The formula we will be using is the equation for a line, but for a log-log scale. From above Figure 6.4, we try to derive the following calculations.

$$y = mx + b \qquad (11)$$

For a log-log scale, the formula looks like this:

$$\log_{10}y = m * \log_{10}x + b \qquad (12)$$

Let's find the slope. To do so, we need to choose 2 points from the graph. In our case, we chose the points (200,2.6) and (10000,0.75). The formula to calculate slope m(here) is the following:

$$m = \{\log y - \log(y0)\} / \{ \log x - \log(x0)\} \qquad (13)$$

If we apply the logarithmic quotient rule, we get the following:

$$m = \log( y/y0 ) / \log( x/x0 ) \qquad (14)$$

Now we substitute the values for x, x0, y, and y0:

$$m = \log(0.75/2.6) / \log(10000/200) \qquad (15)$$

$$m =-0.318 \qquad (16)$$

Now that we have m, we can calculate the y-intercept. To do so, we need to choose one point from the graph (once again from the CO2 line). In our case, we chose (5000,0.9)

$$\log(y) = m * \log(x) + b \qquad (17)$$

$$b = \log(0.9) - (-0.318) * \log(5000) \qquad (18)$$

$$b = 1.13 \qquad (19)$$

Now that we have m and b, we can find the gas concentration for any ratio with the following formula:

$$\log(x) = \{\log(y) - b\} / m \qquad (20)$$

However, in order to get the real value of the gas concentration according to the log-log plot we need to find the inverse log of x:

$$x = 10 \wedge [\{\log(y) - b] / m] \qquad (21)$$

Using eqns. 9 and 21, we will be able to convert the sensor output values into PPM (Parts per Million). Now we developed the Code and flashed into the NodeMCU giving proper connections.

## 6.5.2 HARDWARE CALIBRATE MQ-135 GAS SENSOR

**STEP 1 :** The Vcc pin of the MQ-135 gas sensor module was connected with the 3.3V pin of NodeMCU.

**STEP 2 :** The Gnd pin of the MQ-135 gas sensor module was connected with the Gnd pin of NodeMCU.

**STEP 3 :** The analog DATA pin of the MQ-135 gas sensor module was connected with the A0 Pin of the NodeMCU.

**STEP 4 :** The software code to calibrate the sensor is then uploaded to the NodeMCU and the value of R0 in fresh air is collected from the serial monitor of the Arduino IDE. The setup was then disconnected.

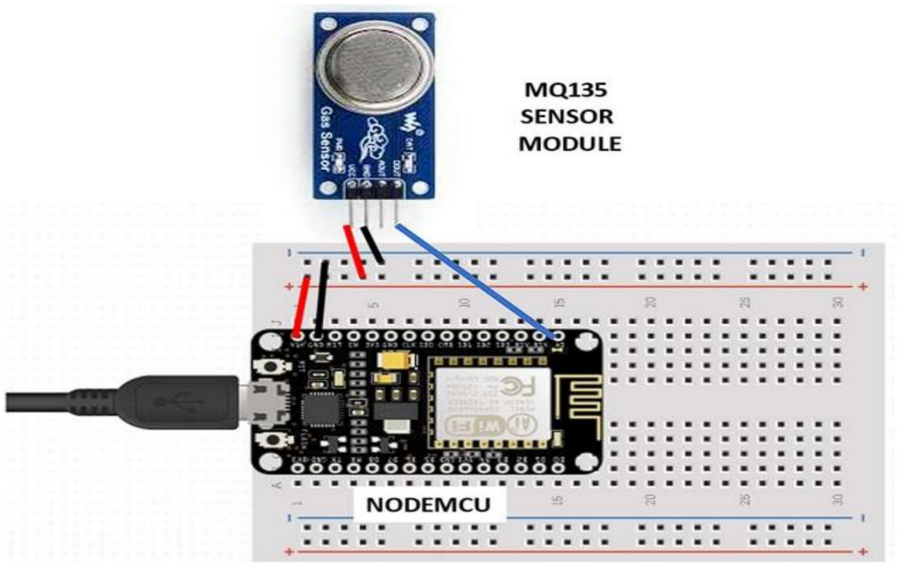


Figure 6.5 (Circuit Diagram to Calibrate the MQ-135 Gas sensor module)

### 6.5.2.1 SOFTWARE CODE FOR CALIBRATION OF MQ135 SENSOR

```
void setup()
        {
                Serial.begin(9600);
                pinMode(A0,INPUT);
        }
void loop()
        {
                float sensor_volt;
                float RS_air;
                float R0;
                float sensorValue=0.0;
                Serial.print("Sensor Reading = ");
                Serial.println(analogRead(A0));
                for(int x = 0 ; x < 500 ; x++)
                {
                        sensorValue = sensorValue + analogRead(A0);
                }
                sensorValue = sensorValue/500.0;
                sensor_volt = sensorValue*(5.0/1023.0);
                RS_air = ((5.0*1.0)/sensor_volt)-1.0;
                R0 = RS_air/3.7;
                Serial.print("R0 = ");
                Serial.println(R0);
                delay(1000);
        }
```

# CHAPTER 7

# DATA ACQUISITION AND TRANSMISSION MECHANISMS

## 7.1 DATA ACQUISITION AND TRANSMISSION

In our project, data acquisition involves gathering and measuring air quality parameters using hardware components such as the NodeMCU v3 ESP8266, MQ135 gas sensor, and DHT11 temperature/humidity sensor.

Sensors are strategically placed to detect gases like carbon dioxide, carbon monoxide, and alcohol, alongside temperature and humidity levels. The NodeMCU microcontroller processes sensor data in real-time, converting analog signals to digital data.

Using Arduino IDE, the microcontroller analyzes data, triggering alerts if values exceed predefined thresholds. Processed data is then transmitted to ThingSpeak, a cloud platform, for storage and visualization.

Stringent measures, including sensor calibration, ensure data accuracy and reliability. Overall, data acquisition is pivotal for monitoring air quality and facilitating informed decision-making for health and safety.

### 7.1.1 DATA ACQUISITION

The NodeMCU v3 ESP8266 microcontroller employs a multifaceted approach to acquire sensor data with precision and reliability. Integrated with a myriad of digital and analog interfaces, it interfaces seamlessly with a gamut of sensors, including the MQ135 gas sensor and DHT11 temperature and humidity sensor. Leveraging its formidable computational prowess, the microcontroller orchestrates the systematic retrieval of sensor readings at predetermined intervals, ensuring a continuous stream of accurate and up-to-date environmental data.

### 7.1.2 DATA TRANSMISSION

Upon acquisition, the NodeMCU v3 ESP8266 microcontroller initiates a meticulously choreographed data transmission process to relay the collected sensor data to cloud-based platforms like ThingSpeak. Employing industry-standard communication protocols such as HTTP and MQTT, the microcontroller establishes secure and reliable connections with remote servers, safeguarding the integrity and confidentiality of transmitted data.

Utilizing HTTP requests, the microcontroller meticulously formats and encapsulates sensor data into structured packets before dispatching them to designated endpoints on ThingSpeak's servers. Alternatively, MQTT protocols may be employed for lightweight and bandwidth-efficient transmission of telemetry data to subscribed channels on the cloud platform.

Through this streamlined data acquisition and transmission pipeline, our Air Quality Monitoring System ensures the expeditious and uninterrupted dissemination of critical environmental data to cloud-based platforms for real-time monitoring, analysis, and visualization. This enables stakeholders to gain invaluable insights into air quality dynamics, facilitating informed decision-making and proactive intervention strategies to safeguard public health and environmental integrity.

## 7.2 MAIN PROGRAM

```
#include <ESP8266WiFi.h>
#include <DHT.h>
#include <ThingSpeak.h>
DHT dht(D2, DHT11);
#define LED_GREEN D6
#define LED_YELLOW D7
#define LED_RED D8
#define MQ_135 A0
#define BUZZ D0
float m = -0.3376;
float b = 0.7165;
float R0 = 3.12;
WiFiClient client;
long myChannelNumber = 2482135;
const char myWriteAPIKey[] = "8LOV7URQYJ18Y60J";

void setup()
{
  Serial.begin(9600);
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(LED_GREEN, OUTPUT);
  pinMode(LED_YELLOW, OUTPUT);
  pinMode(LED_RED, OUTPUT);
  pinMode(BUZZ, OUTPUT);
  pinMode(MQ_135, INPUT);
  WiFi.begin("123", "72345678");
```

```
while(WiFi.status() != WL_CONNECTED)
  {
    delay(100);
    Serial.print(".");
  }
  Serial.println();
  Serial.println("NodeMCU is connected!");
  Serial.println(WiFi.localIP());
  dht.begin();
  ThingSpeak.begin(client);
}
void loop()
{
  digitalWrite(LED_BUILTIN, HIGH);
  float h = dht.readHumidity();
  delay(150);
  float t = dht.readTemperature();
  delay(150);
  int sensorValue = analogRead(MQ_135);
  float sensor_volt = sensorValue * (5.0 / 1023.0);
  float RS_gas = ((5.0 * 1.0) / sensor_volt) - 1.0;
  float ratio = RS_gas / R0;
  float ppm_log = (log10(ratio) - b) / m;
  float ppm = pow(10, ppm_log);
  Serial.println("------------------------------------------------------------------------");
  Serial.println("Temperature: " + (String) t);
  Serial.println("Humidity: " + (String) h);
  Serial.println("Our desired PPM = " + (String) ppm);
  Serial.println("------------------------------------------------------------------------");
  ThingSpeak.writeField(myChannelNumber, 1, ppm, myWriteAPIKey);
  delay(150);
  ThingSpeak.writeField(myChannelNumber, 2, t, myWriteAPIKey);
  delay(150);
  ThingSpeak.writeField(myChannelNumber, 3, h, myWriteAPIKey);
  delay(150);
  if (ppm > 1000)
{
digitalWrite(LED_BUILTIN, LOW);
Serial.println("☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣");
Serial.println("☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣");
Serial.println("☣☣☣☣          TOXIC GAS DETECTED          ☣☣☣☣");
Serial.println("☣☣☣☣       ALERTING USER : CALLING        ☣☣☣☣");
Serial.println("☣☣☣☣           SENDING MESSAGE            ☣☣☣☣");
Serial.println("☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣");
Serial.println("☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣☣");
digitalWrite(LED_GREEN, LOW);
digitalWrite(LED_YELLOW, LOW);
digitalWrite(LED_RED, HIGH);
digitalWrite(BUZZ, HIGH);
}
  else if (t > 45) {
    digitalWrite(LED_BUILTIN, HIGH);
    digitalWrite(LED_GREEN, LOW);
    digitalWrite(LED_YELLOW, LOW);
    digitalWrite(LED_RED, HIGH);
    digitalWrite(BUZZ, HIGH);
  }
```

```
else if (ppm > 700) {
    digitalWrite(LED_BUILTIN, HIGH);
    digitalWrite(LED_GREEN, LOW);
    digitalWrite(LED_YELLOW, HIGH);
    digitalWrite(LED_RED, LOW);
    digitalWrite(BUZZ, LOW);
  }
  else
  {
    digitalWrite(LED_BUILTIN, HIGH);
    digitalWrite(LED_GREEN, HIGH);
    digitalWrite(LED_YELLOW, LOW);
    digitalWrite(LED_RED, LOW);
    digitalWrite(BUZZ, LOW);
  }
  delay(150);
}
```

# CHAPTER 8

# ALERTING MECHANISMS AND USER INTERFACE DESIGN

## 8.1 ALERTING MECHANISMS AND USER INTERFACE

In our IoT Enabled Air Quality Monitoring System, robust alerting mechanisms and a user-friendly interface are pivotal components for ensuring effective communication of air quality status and facilitating user interaction with the system. This chapter elucidates the intricate implementation of alerting mechanisms, encompassing Twilio integration and LED indicators, as well as the thoughtful considerations underlying the design of the user interface for intuitive access to air quality data and system controls.

## 8.2 ALERTING MECHANISMS

The integration of Twilio serves as a cornerstone for alerting users in real-time when hazardous air quality conditions are detected. Through Twilio's cloud communication platform, our system promptly initiates alert calls to users, thereby providing timely notifications regarding critical air quality fluctuations. Additionally, LED indicators, comprising red, yellow, and green lights, are strategically employed to visually convey the current air quality status. Green signifies good air quality, yellow denotes moderate conditions, and red indicates poor or hazardous conditions. Furthermore, in extreme circumstances, a buzzer is activated, emitting a distinct buzzing sound to further alert users, ensuring heightened awareness and prompt response to deteriorating air quality conditions.

## 8.3 USER INTERFACE DESIGN

The design of the user interface is meticulously crafted to provide users with intuitive access to air quality data and system controls. Emphasizing simplicity and clarity, the interface offers a user-friendly dashboard where users can effortlessly visualize real-time air quality metrics, including levels of harmful gases, temperature, and humidity. Moreover, interactive elements allow users to customize alert thresholds and preferences, empowering them to tailor the system to their specific requirements. Through seamless integration with ThingSpeak, users can access comprehensive historical data and perform detailed analysis of air quality trends over time, facilitating informed decision-making and proactive air quality management.

By harmonizing robust alerting mechanisms with an intuitive user interface, our Air Quality Monitoring System ensures effective communication of air quality status and empowers users to take proactive measures to safeguard their well-being. Through continuous refinement and optimization, we strive to uphold the highest standards of usability and functionality, thereby enhancing the overall user experience and promoting sustainable air quality management practices.

# CHAPTER 9

# TESTING AND VALIDATION

## 9.1 TESTING AND VALIDATION OVERVIEW

Testing and validation are critical phases in the development of our IoT Enabled Air Quality Monitoring System, ensuring that the system functions reliably and accurately under various conditions. This chapter provides an overview of the comprehensive testing procedures employed to validate the functionality and performance of the system, along with a discussion of challenges encountered during testing and their resolution.

## 9.2 TESTING PROCEDURES

### Functional Testing:

Functional testing was conducted to verify that all system components, including hardware sensors, microcontrollers, and communication modules, operate as intended. This involved testing sensor readings, LED and buzzer functionality, and communication with cloud platforms.

### Integration Testing:

Integration testing was performed to validate the seamless integration of hardware and software components. This included testing the communication between the NodeMCU microcontroller, sensors, and cloud platforms such as ThingSpeak and Twilio.

**Performance Testing:**

Performance testing focused on assessing the system's responsiveness, accuracy, and reliability under various environmental conditions. This involved subjecting the system to different air quality scenarios and evaluating its ability to detect and alert users of hazardous conditions in a timely manner.

## 9.3 CHALLENGES ENCOUNTERED

**Sensor Calibration:**

One of the primary challenges encountered during testing was calibrating the sensors, particularly the MQ135 gas sensor, to ensure accurate and consistent readings. Calibration involved fine-tuning sensor parameters and addressing environmental factors that could affect sensor performance.

**Communication Reliability:**

Ensuring reliable communication between the NodeMCU microcontroller and cloud platforms posed a challenge, especially in environments with weak or intermittent Wi-Fi connectivity. Strategies such as retry mechanisms and data buffering were implemented to mitigate communication issues.

**User Interface Optimization:**

Optimizing the user interface for usability and responsiveness across different devices and screen sizes presented challenges. Iterative testing and refinement were performed to enhance the UI's performance and ensure a seamless user experience.

# CHAPTER 10

# RESULTS AND ANALYSIS

## 10.1 EXPERIMENT – 1

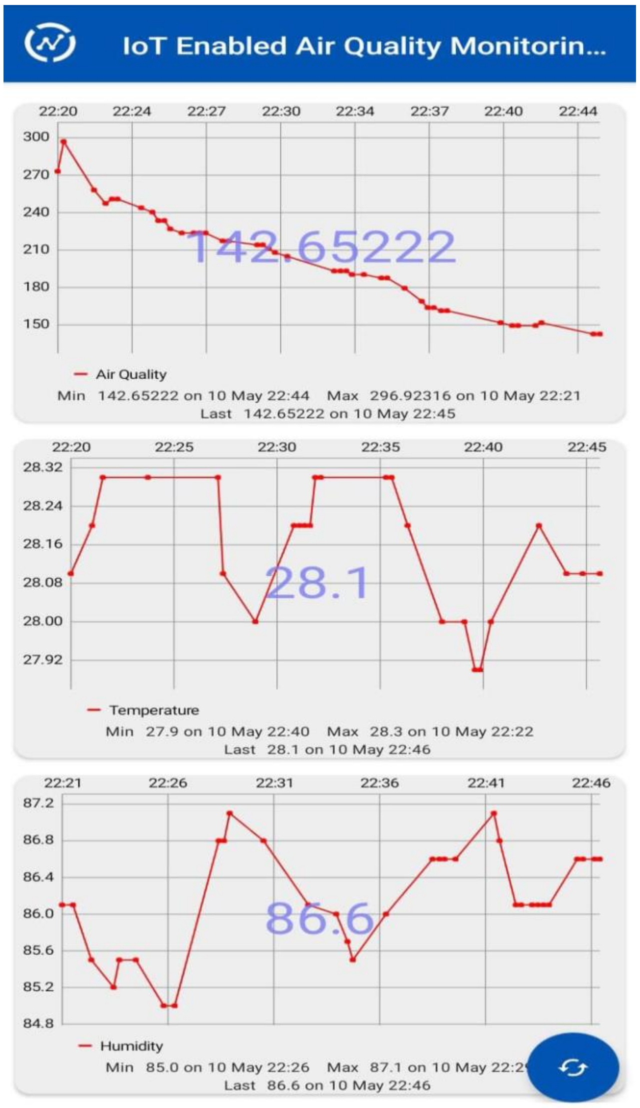
### AIM:

To demonstrate the working of the system in a warm and humid outdoor atmosphere.

### EXPERIMENTAL CONDITION:

The experiment was performed at night.

### OBSERVATIONS IN THINGVIEW MOBILE APPLICATION :



Min 142.65222 on 10 May 22:44 Max 296.92316 on 10 May 22:21
Last 142.65222 on 10 May 22:45

Min 27.9 on 10 May 22:40 Max 28.3 on 10 May 22:22
Last 28.1 on 10 May 22:46

Min 85.0 on 10 May 22:26 Max 87.1 on 10 May 22:2
Last 86.6 on 10 May 22:46

Figure 10.1 Observations for Experiment 1

## OBSERVATIONS IN THINGSPEAK CLOUD:

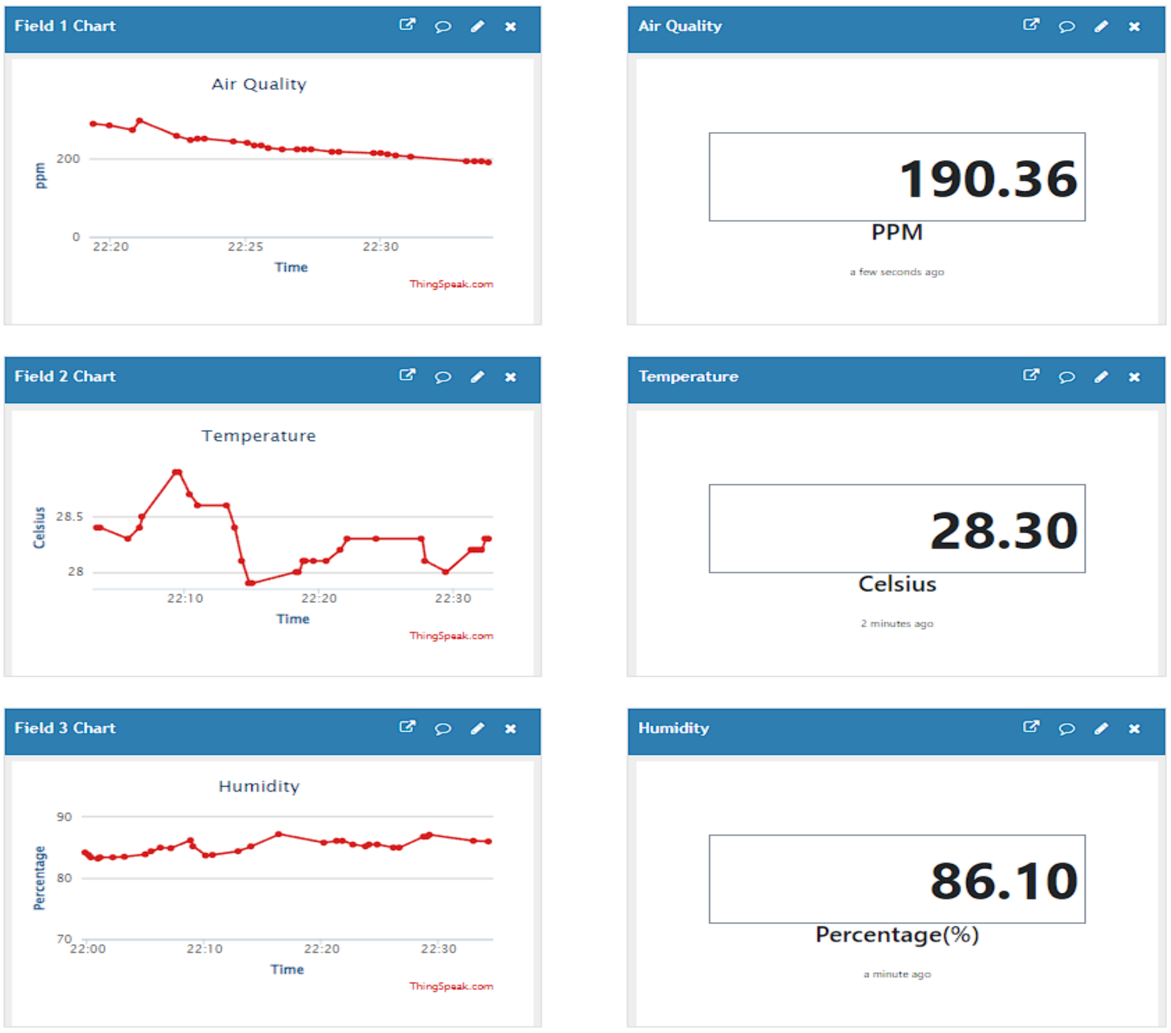


Figure 10.2 Observations for Experiment 1

## SETUP:

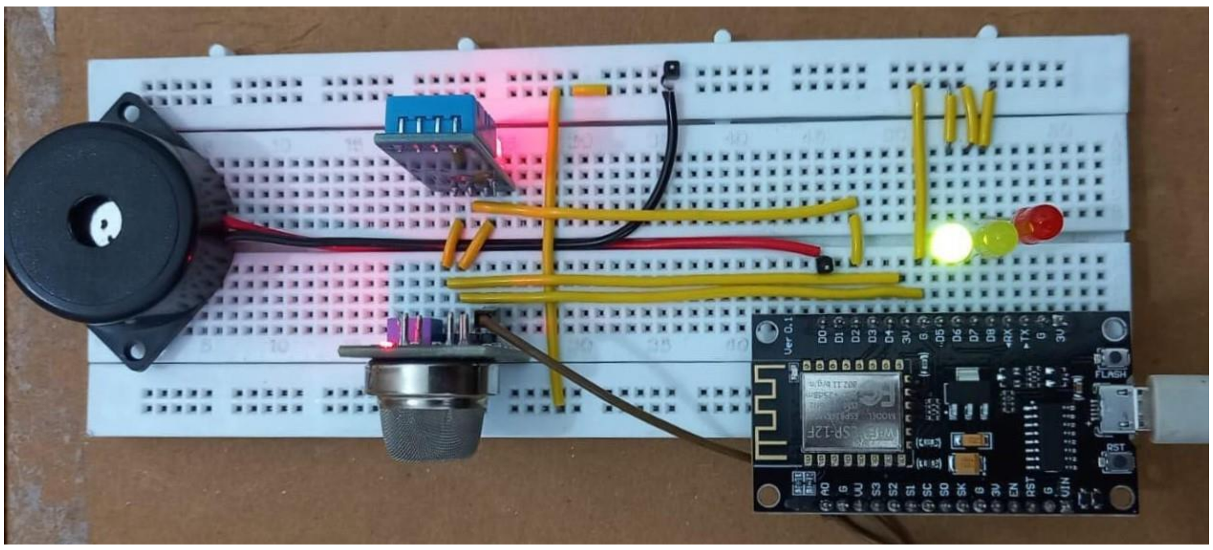


Figure 10.3 Setup for Experiment 1

## 10.2 EXPERIMENT – 2

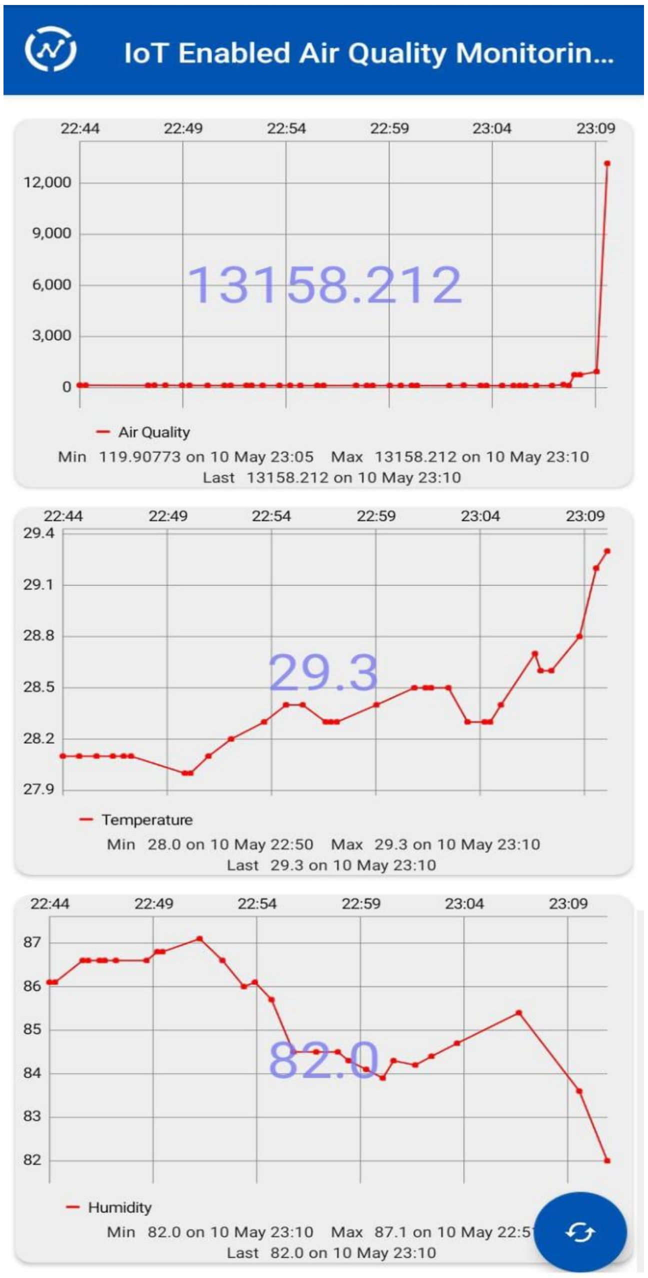
**AIM:**

To demonstrate the working of the system in smoky conditions.

**EXPERIMENTAL CONDITION:**

The experiment was performed in the presence of smoke coming from an incense stick placed near the setup.

**OBSERVATIONS IN THINGVIEW MOBILE APPLICATION:**



Min 119.90773 on 10 May 23:05 Max 13158.212 on 10 May 23:10
Last 13158.212 on 10 May 23:10

Min 28.0 on 10 May 22:50 Max 29.3 on 10 May 23:10
Last 29.3 on 10 May 23:10

Min 82.0 on 10 May 23:10 Max 87.1 on 10 May 22:5
Last 82.0 on 10 May 23:10

Figure 10.4 Observations for Experiment 2

## OBSERVATIONS IN THINGSPEAK CLOUD:

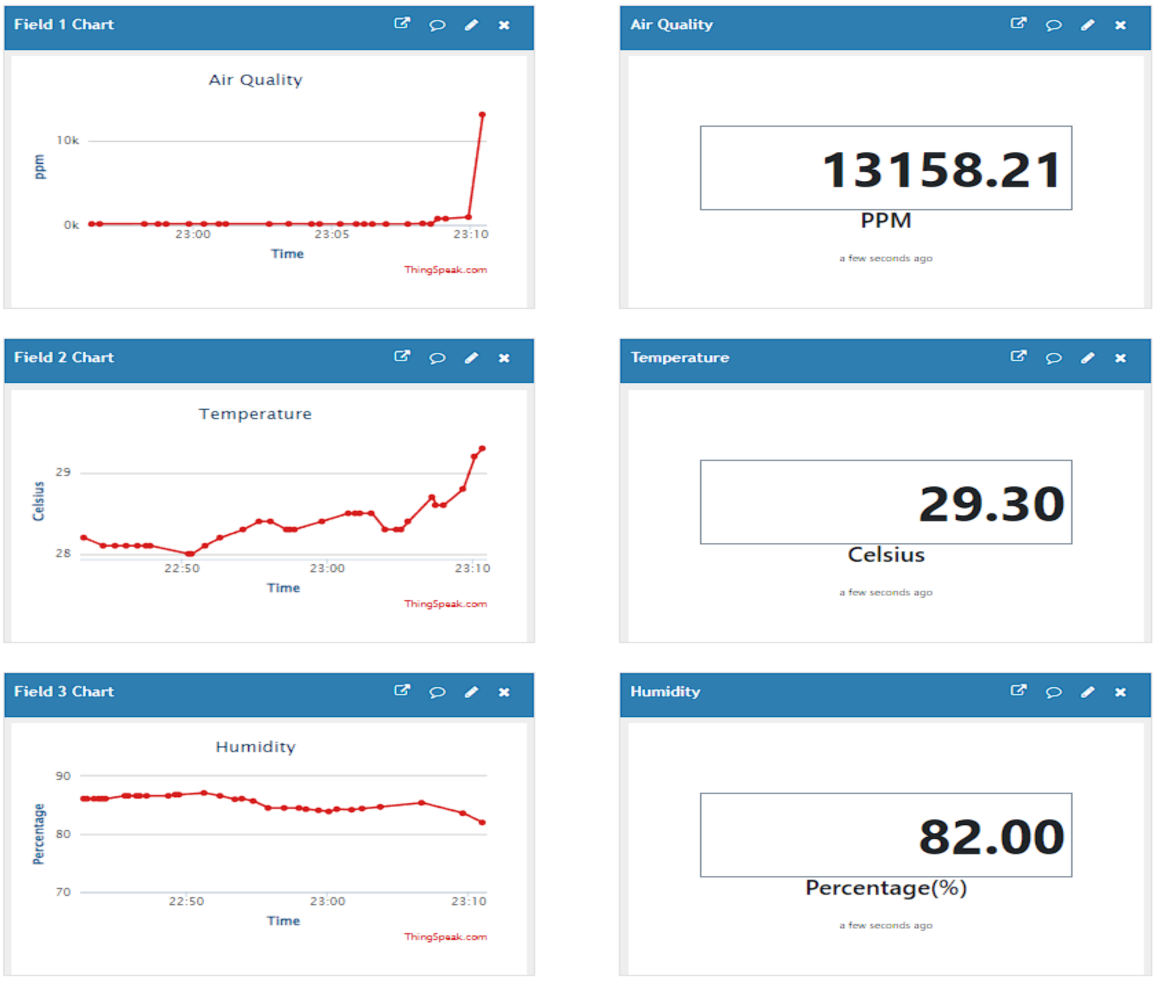


Figure 10.5 Observations for Experiment 2

## SETUP:

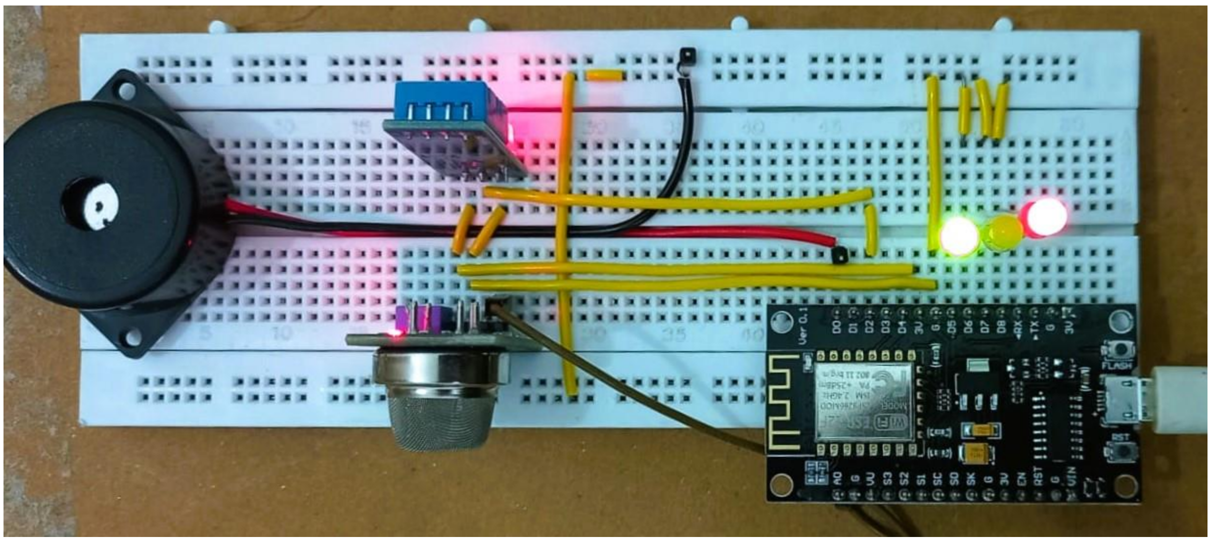
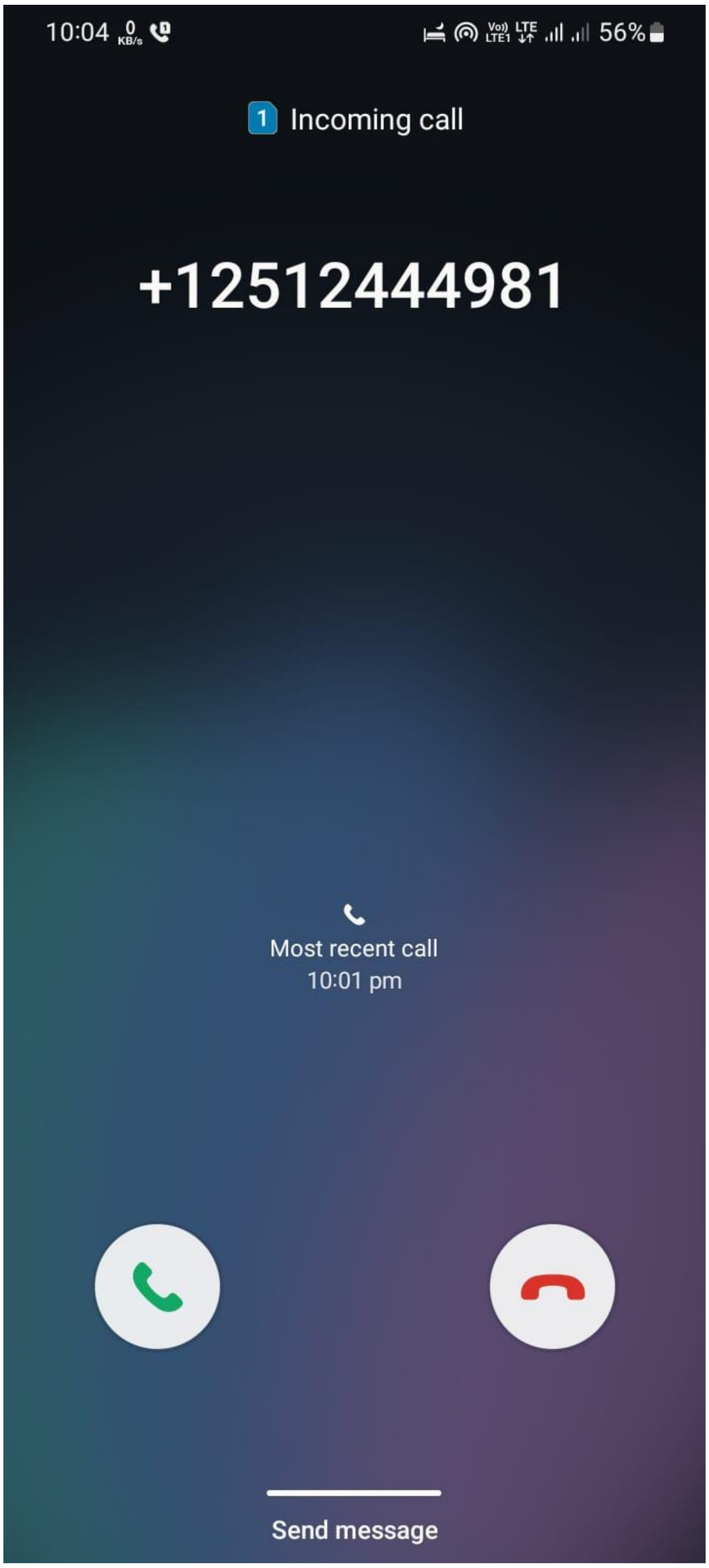


Figure 10.6 Setup for Experiment 2

**CALL ALERT:**



Incoming call

+12512444981

Most recent call
10:01 pm

Figure 10.7 Call Alert for Experiment 2

## 10.3 EXPERIMENT – 3

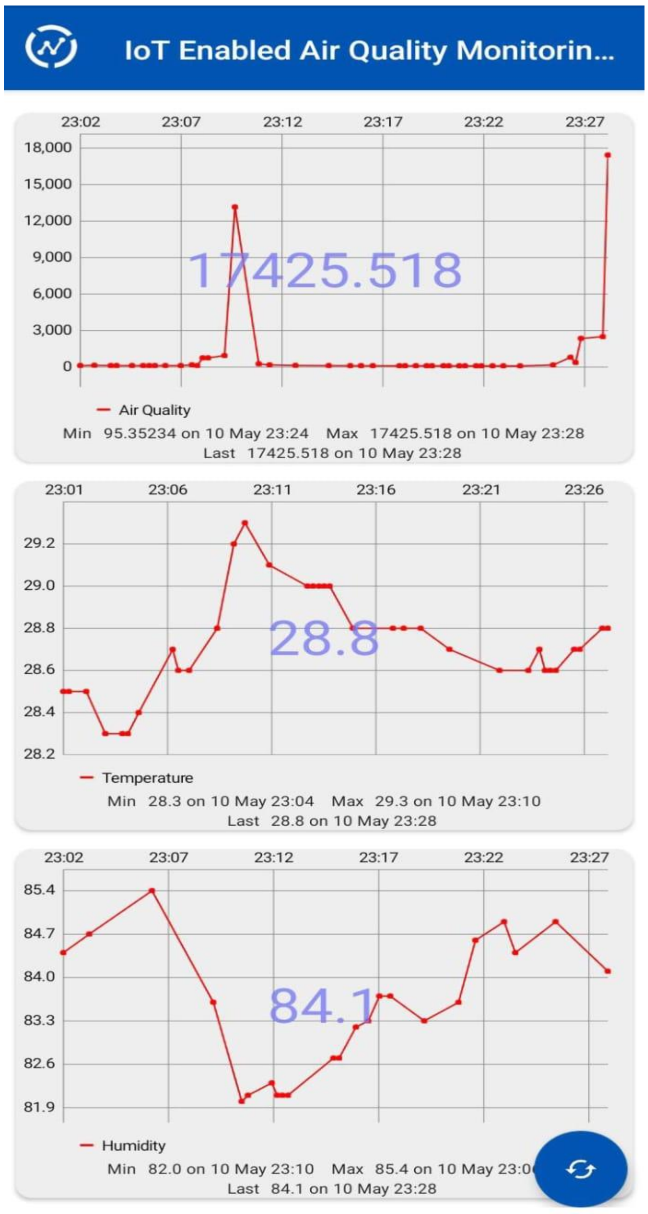
### AIM:

To demonstrate the working of the system in the presence of alcoholic gases.

### EXPERIMENTAL CONDITION:

The experiment was performed in the presence of alcoholic gases.

### OBSERVATIONS IN THINGVIEW MOBILE APPLICATION:



Min 95.35234 on 10 May 23:24 Max 17425.518 on 10 May 23:28
Last 17425.518 on 10 May 23:28

Min 28.3 on 10 May 23:04 Max 29.3 on 10 May 23:10
Last 28.8 on 10 May 23:28

Min 82.0 on 10 May 23:10 Max 85.4 on 10 May 23:0
Last 84.1 on 10 May 23:28

Figure 10.8 Observations for Experiment 3

## OBSERVATIONS IN THINGSPEAK CLOUD:

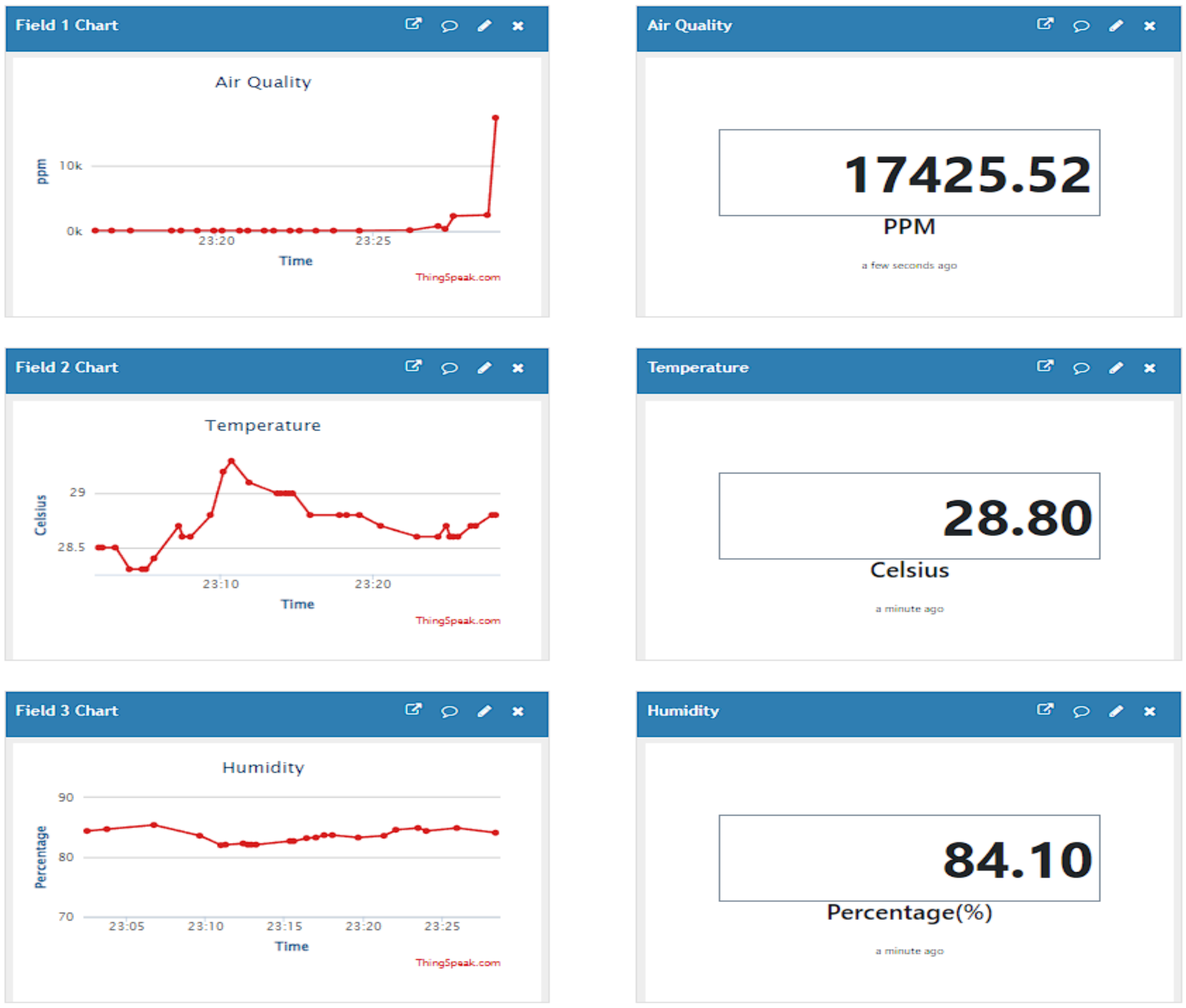


Figure 10.9 Observations for Experiment 3

## SETUP:

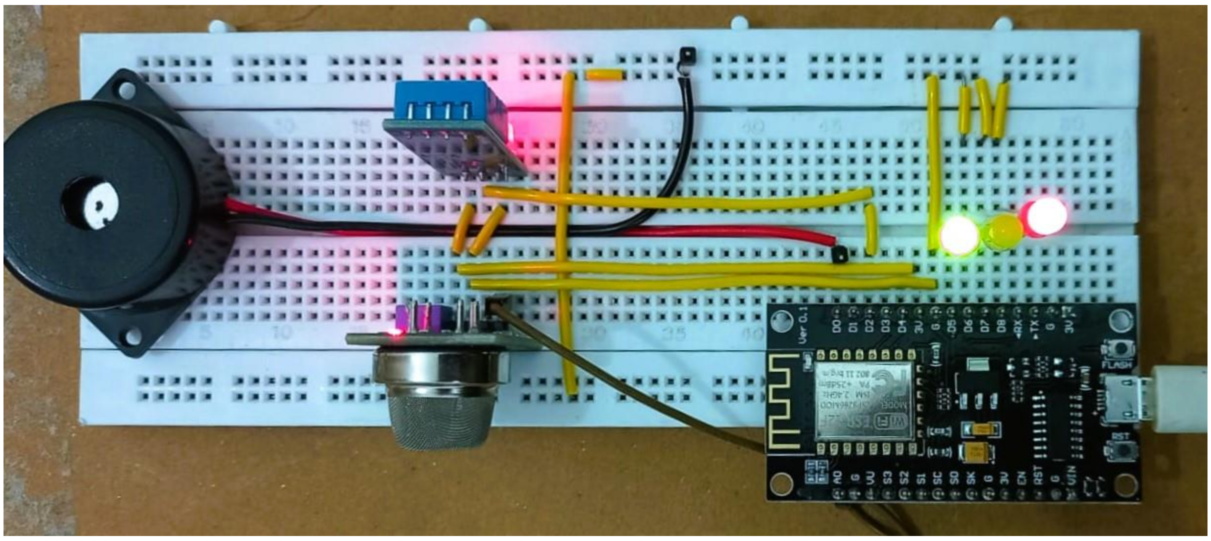
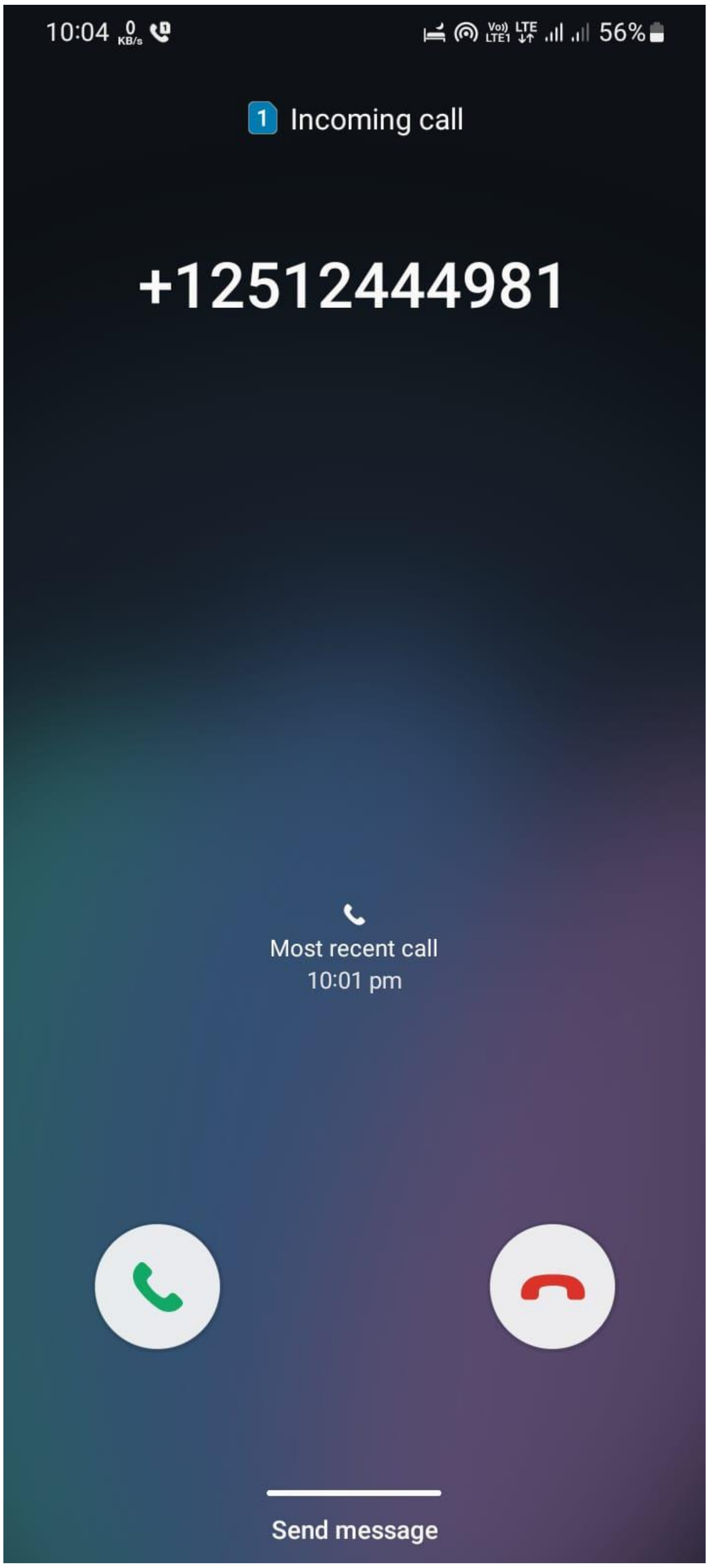


Figure 10.10 Setup for Experiment 3

**CALL ALERT:**


10:04 0 KB/s

56%

Incoming call

+12512444981

Most recent call
10:01 pm

Send message

Figure 10.11 Call Alert for Experiment 3

## 10.4 EXPERIMENTAL CONCLUSIONS

### EXPERIMENT 1: WARM AND HUMID OUTDOOR ATMOSPHERE

The system demonstrated robust performance in a warm and humid outdoor atmosphere, where air quality remained within acceptable parameters. Throughout the experiment, the system effectively monitored temperature, humidity, and other air quality parameters. LED indicators maintained a steady green glow, indicating optimal air quality conditions. This showcases the system's reliability in typical outdoor settings.

### EXPERIMENT 2: SMOKY CONDITIONS

In the presence of smoke, representing deteriorating air quality conditions, the system promptly detected the harmful particles and gases. The MQ135 gas sensor efficiently identified the presence of pollutants, triggering an immediate response. LED indicators transitioned to a red glow, signaling the onset of poor air quality, while the buzzer sounded an alarm to alert users. This highlights the system's capability to respond effectively to adverse environmental conditions.

### EXPERIMENT 3: PRESENCE OF ALCOHOLIC GASES

When exposed to alcoholic gases, such as those found in sanitizers, the system swiftly detected the presence of these volatile compounds. The MQ135 sensor accurately identified the alcoholic gases, initiating appropriate alerts. LED indicators illuminated red, signaling hazardous conditions, while the integrated Twilio platform swiftly delivered emergency alert calls to users. This underscores the system's efficacy in detecting a wide range of harmful substances and promptly notifying users to ensure their safety.

# CHAPTER 11

# CONCLUSION

Based on the project's implementation, it is evident that an IoT-based Air Quality Monitoring System has been successfully developed to measure and display Air Quality Index (AQI), humidity, and temperature of the atmosphere. Despite the limitation of the MQ135 sensor in not being able to directly measure Carbon Monoxide or Carbon Dioxide levels, it offers the advantage of detecting various harmful gases like smoke, CO, CO2, and NH4.

Through multiple experiments, it has been demonstrated that the system can accurately measure air quality in ppm, temperature in Celsius, and humidity in percentage. The validity of the results has been confirmed through comparison with Google data. Additionally, LED indicators have been incorporated into the setup to provide real-time visual cues regarding air quality levels.

However, a notable drawback of the project is its inability to measure ppm values of individual pollutant components separately. This limitation could potentially be addressed by integrating additional gas sensors for different pollutants. Nonetheless, such enhancements would increase the cost of the setup and may not be necessary for basic air quality monitoring purposes.

Moreover, the project leverages IoT capabilities by utilizing a stable internet connection for uploading data to the ThingSpeak cloud platform. This ensures that the system can continuously monitor and analyze air quality, humidity, and temperature data in real-time.

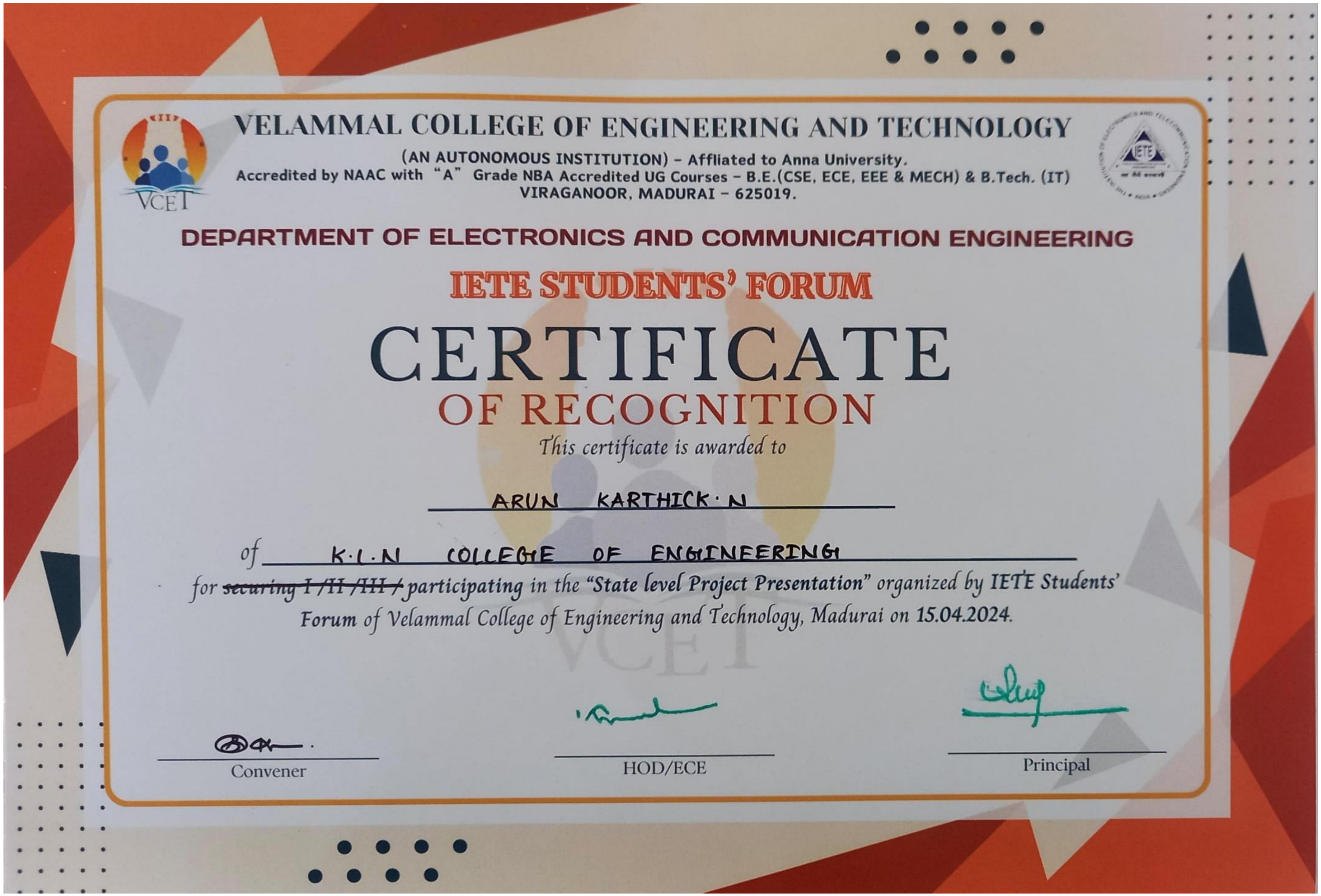
Furthermore, an alert system has been implemented using the Twilio platform, facilitated by ThingHTTP and ThingReact integration. This feature enables the system to promptly notify users via calls or messages in case of hazardous air quality conditions, enhancing its practical utility and user engagement.

In conclusion, the designed prototype demonstrates the successful implementation of an IoT-based solution for monitoring air quality, humidity, and temperature of the surrounding atmosphere. Despite some limitations, the system offers valuable insights into environmental conditions and can contribute to efforts aimed at enhancing public health and environmental awareness.

# CHAPTER 12

# REFERENCE

1. https://gaslab.com/blogs/articles/carbon-monoxide-levels
2. https://www.instructables.com/Measuring-Humidity-Using-Sensor-DHT11
3. https://pdf1.alldatasheet.com/datasheet-pdf/view/1307647/WINSEN/MQ135.html
4. https://components101.com/development-boards/nodemcu-esp8266-pinout-featuresand-datasheet
5. https://www.arduino.cc
6. https://thingspeak.com
7. https://www.codrey.com/electronic-circuits/how-to-use-mq-135-gas-sensor
8. Pasha, S. (2016). ThingSpeak based sensing and monitoring system for IoT with Matlab Analysis. International Journal of New Technology and Research, 2(6).
9. Kumar, N. S., Vuayalakshmi, B., Prarthana, R. J., & Shankar, A. (2016, November). IOT based smart garbage alert system using Arduino UNO. In 2016 IEEE Region 10 Conference (TENCON) (pp. 1028-1034). IEEE.
10. IoT based Air Quality monitoring system using MQ135 & MQ7 with Machine Learning analysis by Kinnera Bharath Kumar Sai M.Tech CSE VIT University, Vellore Subhaditya Mukherjee B.Tech CSE VIT University, Vellore Dr. Parveen Sultana H Associate Professor Department of CSE, VIT University.



VELAMMAL COLLEGE OF ENGINEERING AND TECHNOLOGY

(AN AUTONOMOUS INSTITUTION) - Affliated to Anna University.
Accredited by NAAC with "A" Grade NBA Accredited UG Courses - B.E.(CSE, ECE, EEE & MECH) & B.Tech. (IT)
VIRAGANOOR, MADURAI - 625019.

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

IETE STUDENTS' FORUM

CERTIFICATE
OF RECOGNITION

*This certificate is awarded to*

ARUN KARTHICK · N

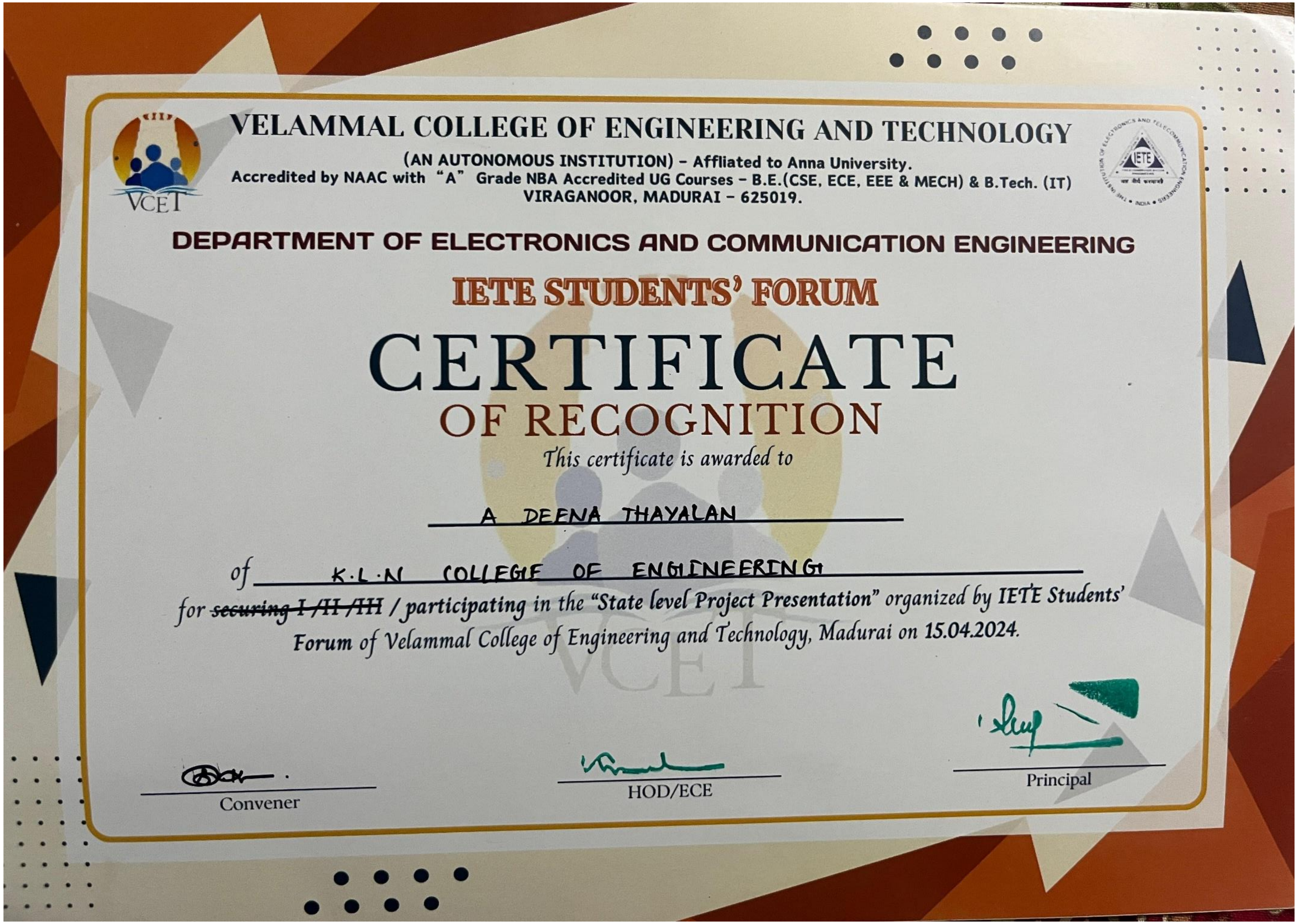
*of* K·L·N COLLEGE OF ENGINEERING

*for ~~securing I / II / III /~~ participating in the "State level Project Presentation" organized by IETE Students' Forum of Velammal College of Engineering and Technology, Madurai on 15.04.2024.*

[illegible] Convener

[illegible] HOD/ECE

[illegible] Principal



VELAMMAL COLLEGE OF ENGINEERING AND TECHNOLOGY

(AN AUTONOMOUS INSTITUTION) - Affliated to Anna University.
Accredited by NAAC with "A" Grade NBA Accredited UG Courses - B.E.(CSE, ECE, EEE & MECH) & B.Tech. (IT)
VIRAGANOOR, MADURAI - 625019.

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

IETE STUDENTS' FORUM

CERTIFICATE
OF RECOGNITION

This certificate is awarded to

A DEENA THAYALAN

of K.L.N COLLEGE OF ENGINEERING

for ~~securing I /II /III~~ / participating in the "State level Project Presentation" organized by IETE Students' Forum of Velammal College of Engineering and Technology, Madurai on 15.04.2024.

[illegible] Convener

[illegible] HOD/ECE

[illegible] Principal