ENVIRONMENTAL MONITORING PROJECT

Introduction:

Air pollution is a pressing global issue, posing serious threats to public health and the environment. In response to this challenge, we present "BreatheSmart," an innovative air pollution monitoring project driven by the Internet of Things (IoT). BreatheSmart aims to harness cutting-edge technology to provide real-time, accurate, and actionable air quality data, empowering communities to make informed decisions for a cleaner and healthier future.

Project Objectives:

1. **IoT Sensor Network:**
   Deploy a network of IoT-enabled air quality sensors across our region, strategically positioned to continuously measure key air pollutants, including particulate matter (PM2.5 and PM10), nitrogen dioxide (NO2), sulfur dioxide (SO2), carbon monoxide (CO), ozone (O3), and volatile organic compounds (VOCs).

2. **Data Integration:**
   Establish a centralized data platform that collects, processes, and analyzes sensor data in real-time, providing a comprehensive and up-to-date view of air quality conditions.

3. **User-Friendly Interface:**
   Develop intuitive mobile applications and web interfaces that make air quality data easily accessible to the public, enabling individuals to check the air quality in their vicinity and receive timely alerts.

4. **Advanced Analytics:**
   Utilize sophisticated analytics and machine learning algorithms to predict air quality fluctuations, enabling proactive measures to mitigate exposure to harmful pollutants.

5. **Community Engagement:**

Educate and engage local communities on the importance of air quality through awareness campaigns, workshops, and educational programs, fostering a sense of collective responsibility for air quality improvement.

6. **Policy Impact:**

Collaborate with local authorities and policymakers, providing data-driven insights to inform evidence-based policies and regulations aimed at improving air quality.

Expected Outcomes:

- Increased public awareness of air quality and its impact on health and well-being.
- Access to real-time air quality information, enabling individuals to make informed decisions.
- Improved public health outcomes by reducing exposure to harmful air pollutants.
- Evidence-based policy recommendations for more effective air quality management.
- A sustainable and healthier environment for current and future generations.

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. This project aims to design an IOT-based air pollution monitoring system using the internet from anywhere using a computer or mobile to monitor the air quality of the surroundings and environment. There are various methods and instruments available for the measurement and monitoring quality of air. The IoT-based air pollution monitoring system would not only help us to monitor the air quality but also be able to send alert signals whenever the air quality deteriorates and goes down beyond a certain level.

In this system, **NodeMCU** plays the main controlling role. It has been programmed in a manner, such that, it senses the sensory signals from the sensors and shows the quality level via led indicators. Besides the harmful gases (such as CO_2, CO, smoke, etc) temperature and humidity can be monitored through the temperature and humidity sensor by this system. Sensor responses are fed to the NodeMCU which displays the monitored data in the ThingSpeak cloud which can be utilized for analyzing the air quality of that area.The following simple flow diagram (as shown in Fig. 1) indicates the working mechanism of the IoT-based Air Pollution Monitoring System.

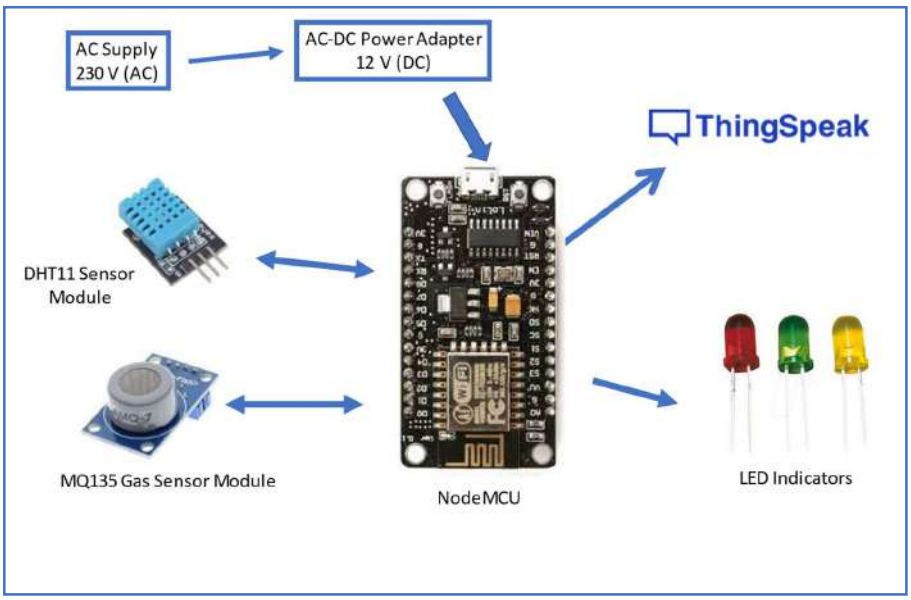


Fig.1. IoT based Air Pollution Monitoring System

2.2 Components Used

❖ Hardware Components

1. NodeMCU V3
2. DHT11 Sensor Module
3. MQ-135 Gas Sensor Module
4. Veroboard(KS100)
5. Breadboard
6. Connecting Wires
7. AC-DC Adapters
8. LEDs emitting green, yellow and red colours
9. Resistors

❖ SOFTWARE COMPONENTS

1. ThinkSpeak Cloud
2. Arduino IDE

2.3 Brief Description of the Components

❖ NodeMCU V3

NodeMCU V3 is an open-source ESP8266 development kit, armed with the CH340G USB-TTL 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.

The pinout Diagram of NodeMC3 is shown in Fig. 2.1.

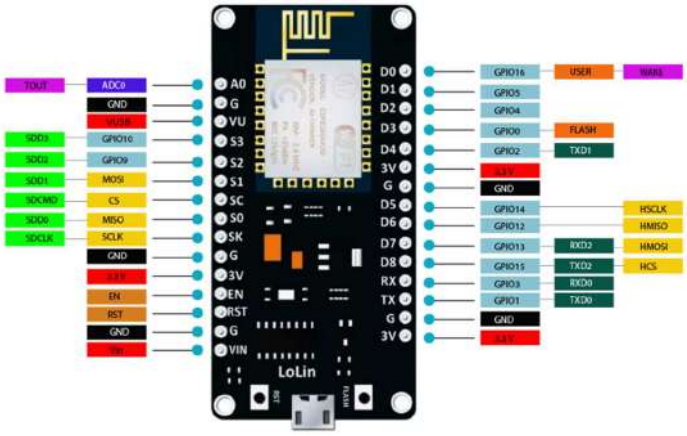


Fig. 2.1 (Pinout Diagram of NodeMCU V3)

❖ DHT11 Sensor Module

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. 2.2, 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).

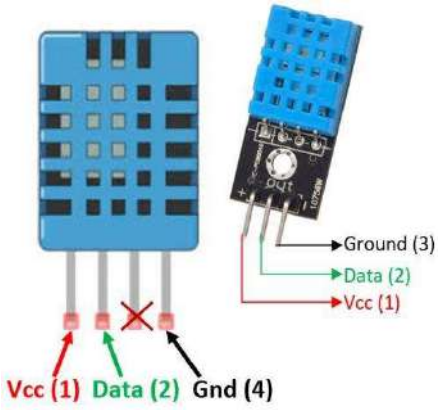


Fig 2.2 (Pinout Diagram of DHT11sensor)

Humidity Measurement: The humidity sensing capacitor has two electrodes with a moisture-holding substrate as a dielectric between them as shown in Fig 2.3. 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.

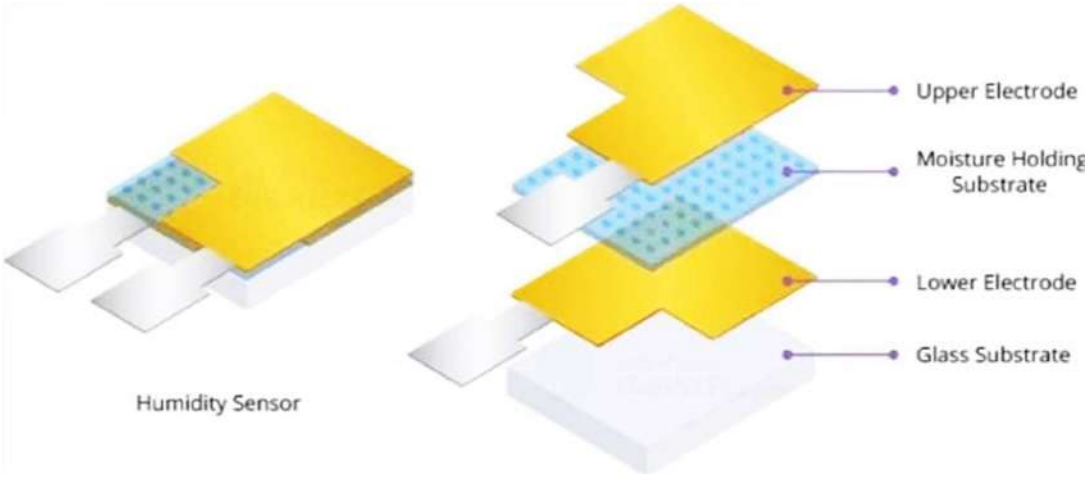


Fig 2.3(The structure of the humidity sensor)

❖ MQ-135 Gas Sensor Module

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.

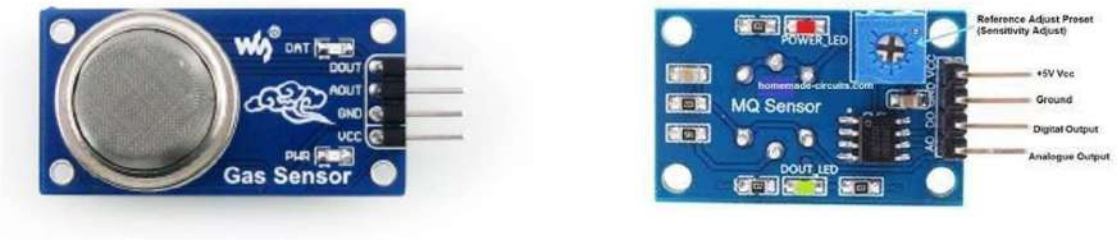


Fig 2.4 (MQ-135 Gas Sensor Module)

❖ Veroboard (KS100)

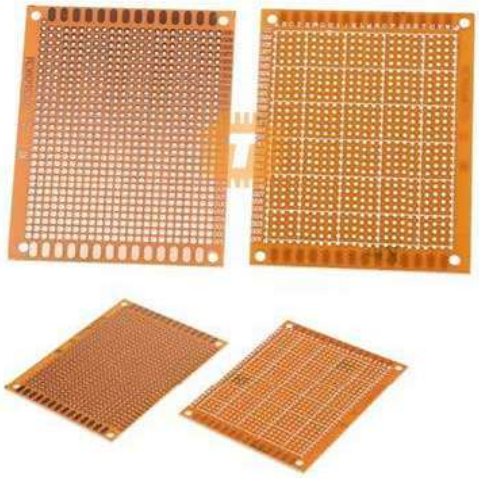


Fig 2.5 Veroboard

Veroboard is the original prototyping board. Sometimes referred to as 'stripboard' or 'matrix board' these offer total flexibility for hard wiring discrete components. Manufactured from a copper clad laminate board or Epoxy based substrate, it is offered in both single and double-sided formats. Vero boards are available in a wide range of board sizes and in both imperial and metric pitch – Veroboard is an ideal base for circuit construction and offers even greater adaptability using our range of terminal pins and assemblies. As with other stripboards, in using Veroboard, components are suitably positioned and soldered to the conductors to form the required circuit. Breaks can be made in the tracks, usually around holes, to divide the strips into multiple electrical nodes enabling increased circuit complexity. This type of wiring

board may be used for initial electronic circuit development, to construct prototypes for bench testing or in the production of complete electronic units in small quantities.

❖ AC-DC Power Adapter

An AC-DC power supply or adapter is an electrical device that obtains electricity from a grid-based power supply and converts it into a different current, frequency, and voltage. AC-DC power supplies are necessary to provide the right power that an electrical component needs. The AC-DC power supply delivers electricity to devices that would typically run-on batteries or have no other power source.

Fig 2.6 AC-DC Power Adapter

❖ LED (Red, Green & Yellow)

A light-emitting diode (LED) is a semiconductor light source that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons. The colour of the light (corresponding to the energy of the photons) is determined by the energy required for electrons to cross the band gap of the semiconductor. White light is obtained by using multiple semiconductors or a layer of light-emitting phosphor on the semiconductor device. LEDs have many advantages over incandescent light sources, including lower power consumption, longer lifetime, improved physical robustness, smaller size, and faster switching. In exchange for these generally favourable attributes, disadvantages of LEDs include electrical limitations to low voltage and generally to DC (not AC) power, inability to provide steady illumination from a pulsing DC or an AC electrical supply source, and lesser maximum operating temperature and storage temperature. In contrast to LEDs, incandescent lamps can be made to intrinsically run at virtually any supply voltage, can utilize either AC or DC current interchangeably, and will provide steady illumination when powered by AC or pulsing DC even at a frequency as low as 50 Hz. LEDs usually need electronic support

Fig 2.7LEDs

components to function, while an incandescent bulb can and usually does operate directly from an unregulated DC or AC power source.

❖ Resistors

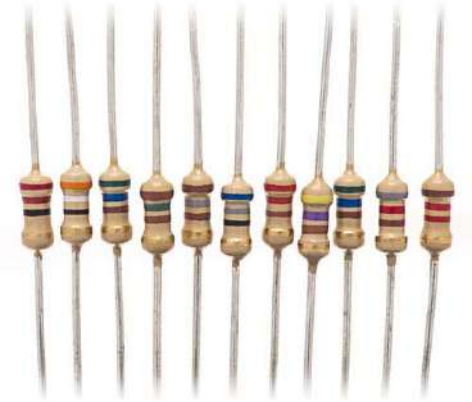


Fig 2.8 Resistors

A resistor is a passivetwo-terminalelectrical component that implements electrical resistance as a circuit element. In electronic circuits, resistors are used to reduce current flow, adjust signal levels, to divide voltages, bias active elements, and terminate transmission lines, among other uses. High-power resistors that can dissipate many watts of electrical power as heat may be used as part of motor controls, in power distribution systems, or as test loads for generators. Fixed resistors have resistances that only change slightly with temperature, time or operating voltage.

❖ Arduino IDE

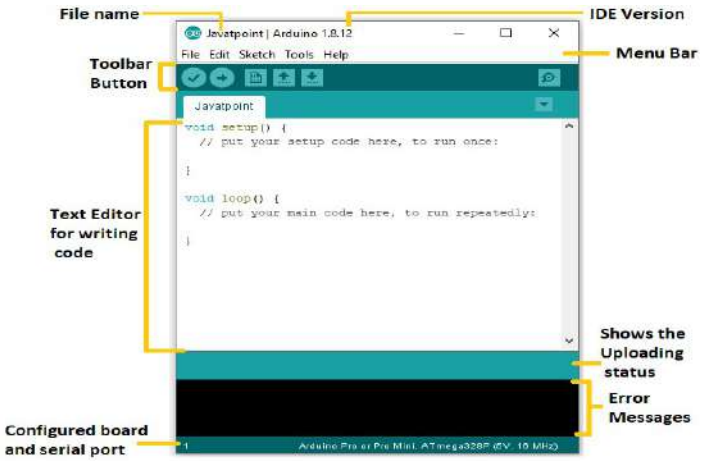


Fig 2.9Arduino IDE

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.'

❖ ThingSpeak Cloud

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.

Fig 2.10 ThingSpeak Cloud

2.4 Working Procedures

NodeMCU plays the main controlling role in this project. It has been programmed in a manner, such that, it senses the sensory signals from the sensors and shows the quality level via led indicators. The DHT11 sensor module is used to measure the temperature and the humidity of the surroundings.With the help of the MQ-135 gas sensor module, air quality is measured in ppm. These data are fed to the ThinkSpeak cloud over the internet. We have also provided LED indicators to indicate the safety levels.

STEP 1. Firstly, the calibration ofthe MQ-135 gas sensor module is done. The sensor is set to preheat for 24 minutes. Then the software code is uploaded to the NodeMCU followed by the hardware circuit to calibrate the sensor has been performed.

STEP 2. Then, the DHT11 sensor is set to preheat for 10 minutes.

STEP 3. The result of calibration found in STEP 1 is used to configure the final working code.

STEP 4. The final working code is then uploaded to the NodeMCU.

STEP 5. Finally, the complete hardware circuit is implemented.

The software codes and the hardware circuits are described in the following chapters.

Table-1: Components used in proposed system

Sr. No.	Sensor/Device	Description
1	**Arduino UNO**	Arduino UNO is one of the leading prototype boards. It is compact and abundant in characteristics. The board comes with an Arduino boot loader integrated. It has fourteen GPIO pins, six PWM pins, six analogue inputs and UART, SPI or TWI interfaces onboards and is an Atmega 328-based controllers. Nine pins on the board are included in this IOT device. Six pins can normally interface the LCD character. Two pins are used for the Wi-Fi interface and an analogue control pin is used to connect to the MQ-135 [3].
2	**16X2 Character LCD**	The 16X2 LCD screen is utilized to exhibit the sensor values that are read by the Arduino MQ-135 module. The interface between the data pins D4 and D7 with the pins 6 up to three of the controller is interconnected with Arduino UNO respectively. The LCD RS and E pins are wired to 13 and 12 pins on each controller. The RW pin is attached to the base of the LCD module.
3	**ESP8266 WIFI**	ESP8266 is one Wi-Fi module for controlling device by using internet. It is cheap and easy to find. It is also worked on microcontroller. That is why we are using this module on Arduino. The ESP8266 comes with pre-installed firmware that allow you to control it with standard "AT commands". You can easily create and upload your own code. That's make it powerful and flexible.
4	**MQ135**	The MQ-135 is a sensor for air quality control. It detects dangerous gases such as NH3, NOx, smoke, and CO2 in air condition monitoring gear. The MQ-135 has a digital pin that allows this device work without the microcontroller and is effective in this project. This is how it tests the gas that is reliably damaging to us.
5	**MQ135 Sensors**	Air temperature sensors for a variety of gases such as CO2, NH3, alcohol, NOx, smoke, and benzene. The MQ135 gas sensor is also vulnerable to smoke and other noxious chemicals, such as ammonia, sulphide and benze steam.
6	**LPG Sensors**	This sensor is used to sense LPG in the air.
7	**Humidity Sensor**	The humidity sensor is an electronic device that is used to sense the humidity in the air and then react. after sensing he send the digital signal to the circuit.
8	**Temperature Sensor**	The temperature sensor is an electronic device that is used to sense the temperature of the surroundings and record the dets in the form of digital signal.
9	**Arduino**	The Arduino Uno is a microcontroller-powered open-source

	UNO	board based on Arduino.cc's ATmega328P micro-controller.The board has optical and analogue input/output pins which are interfaced with different boards with expansion and other circuits.
10	**LCD**	An LCD is a fluid crystal display module that makes a transparent image. A very simple module widely used in DIYs and circuits is the 16 to2 LCD display. The 16 digits2 converts o into 2 such lines a showing of 16 signs per line.
11	**BUZZER**	A buzzer is like an alarm that send the audible signal. A snapshot for the different components is depicted in figure 6(a)-(g).

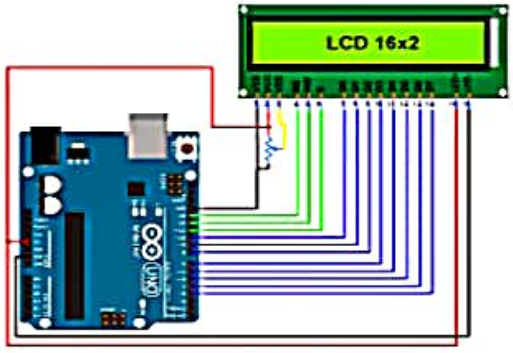
(a)MQ135 Sensors

(b) LPG Sensors

(c) Humidity sensor

(d)Temperature Sensors

(e) Arduino UNO Circuit



(f)LCD 16x2

(g) Buzzer

Figure 6 Components used in the proposed system

HARDWARE MODEL

Hardware Model to Preheat DHT11 Sensor Module

As discussed earlier, we need to preheat the DHT11 sensor so that it can work accurately. 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 VU 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 with a 12V DC via AC-DC adapter for 20 minutes.

STEP 4 : The setup was then disconnected.

Fig. 3.1 shown below describes the foresaid connections.

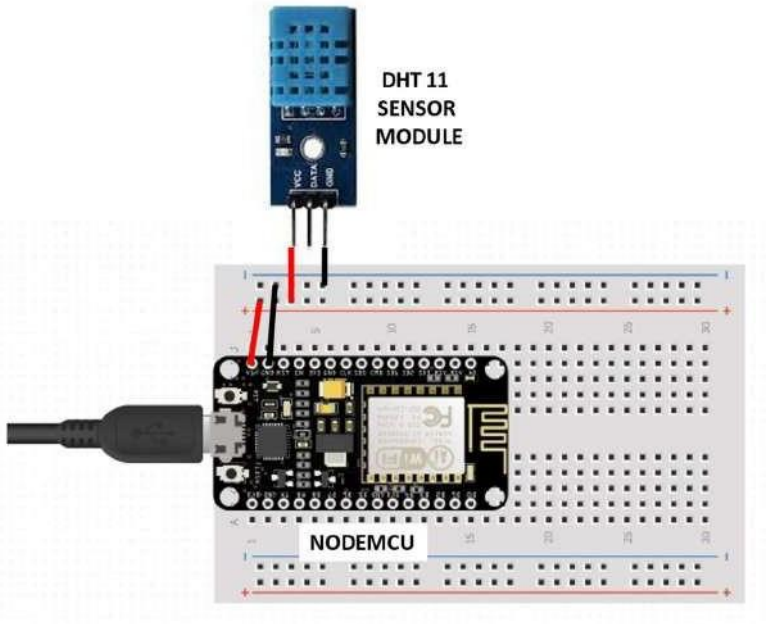


Fig. 3.1 (Circuit Diagram to Preheat the DHT11 sensor module)

Hardware Model to Preheat and Calibrate MQ-135 GasSensor Module ;

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 VU 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 with a 12V DC via AC-DC adapter for a day.

STEP 4 : The setup was then disconnected.

Fig. 3.2 shown below describes the foresaid connections.

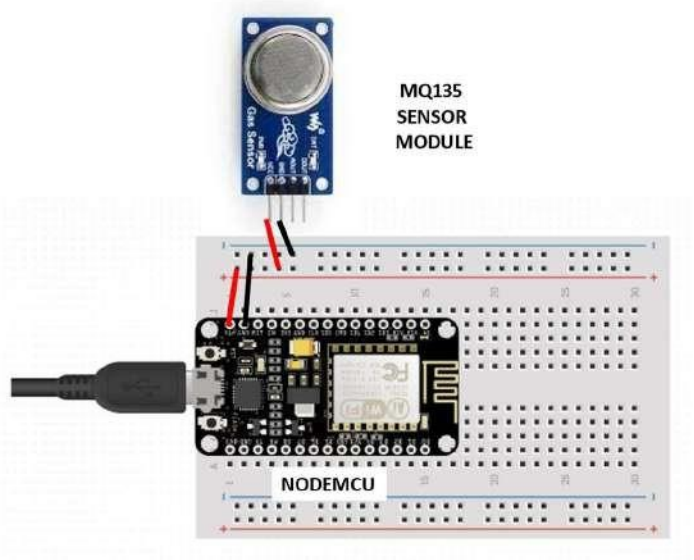


Fig. 3.2(Circuit Diagram to Preheat the MQ-135 Gas sensor module)

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

STEP 1 : The Vcc pin of the MQ-135 gas sensor module was connected with the VU 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 NodeMCUand the value of R_0in fresh air is collected from the serial monitor of the Arduino IDE.

STEP 5 : The setup was then disconnected.

Fig. 3.3 shown below describes the foresaid connections.

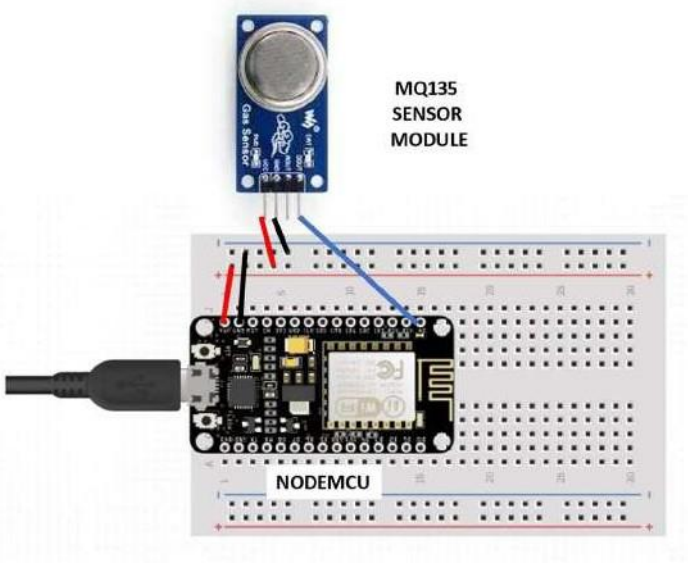


Fig. 3.3(Circuit Diagram to Calibrate the MQ-135 Gas sensor module)

Final Hardware Model ;

The following steps were performed to execute the project

STEP 1 : The Vcc pin of the MQ-135 gas sensor module and DHT11 sensor module was connected via Veroboard with an adapter delivering around 5V.

STEP 2 : The Gnd pin of the MQ-135 gas sensor module, DHT11 sensor module andthe cathode of the LED indicators was connected via Veroboard with the Gnd pin of the NodeMCU.

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

STEP 4 : The DATA pin of the DHT11 sensor module was connected with the D0 pinof the NodeMCU.

STEP 5 : The anode of the three LED indicators (green, yellow, and red) were connected to the D2, D3, and D4 pins of the NodeMCU respectively.

STEP 6 : The software code to execute the project was then uploaded to the NodeMCU.

STEP 7 : The setup was then powered with 9V DC via AC-DC adapter.

It can be now turned ON/OFF as per the requirements. Fig 3.4 represents the circuit diagram of the setup.

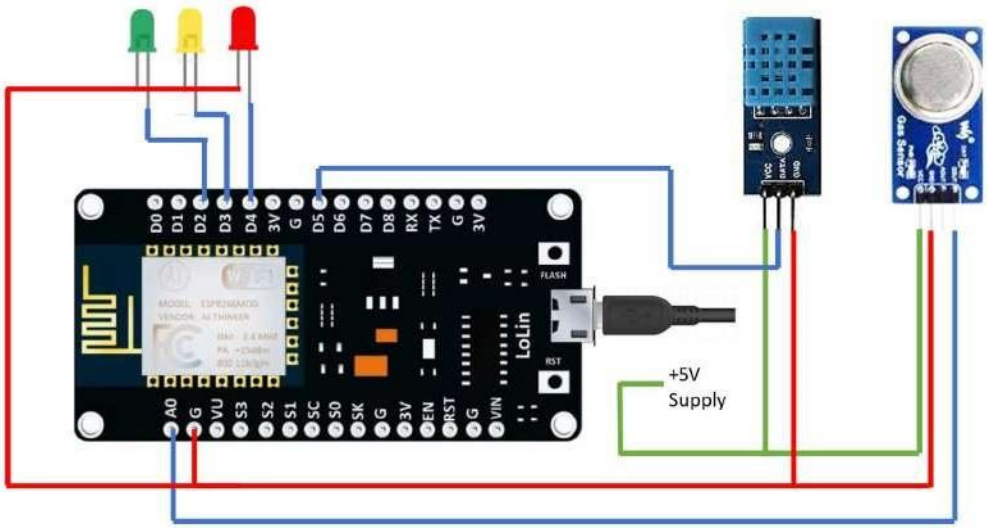
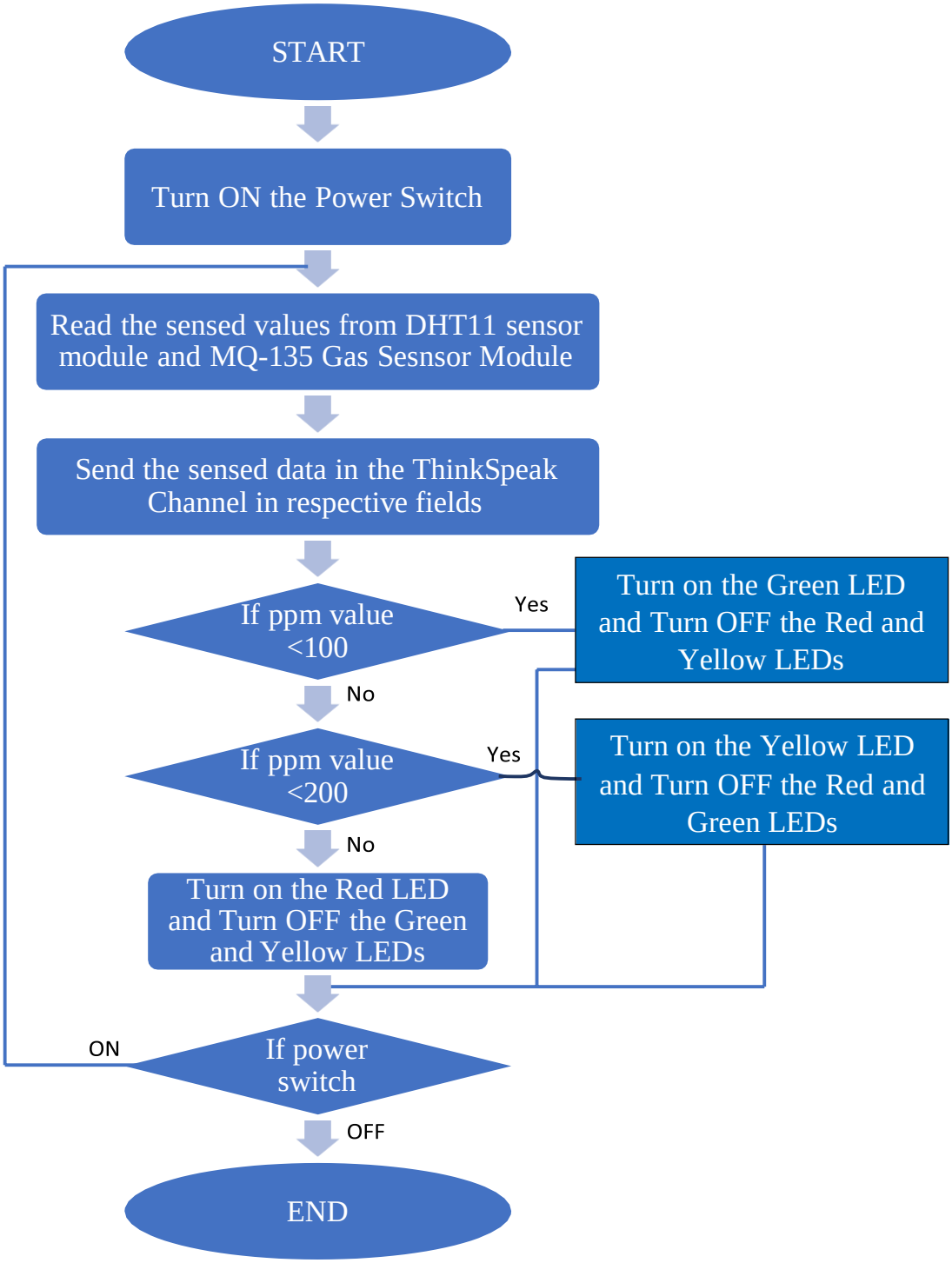


Fig. 3.4(Circuit Diagram of the setup)

SOFTWARE IMPLEMENTATION :

Working Algorithm or Flow Chart:



SOFTWARE CODE for Calibration of MQ135 Sensor:

```
void setup()
{
Serial.begin(9600); //Baud rate
pinMode(A0,INPUT);
}

void loop()
{
float sensor_volt; //Define variable for sensor
voltagefloat RS_air; //Define variable for sensor
resistance float R0; //Define variable for R0
float sensorValue=0.0; //Define variable for
 analog readings
Serial.print("Sensor Reading = ");
Serial.println(analogRead(A0));

for(int x = 0 ; x < 500 ; x++) //Start for loop
  {
sensorValue = sensorValue + analogRead(A0); //Add analog values of sensor 500 times
  }
sensorValue = sensorValue/500.0; //Take average of readings
sensor_volt = sensorValue*(5.0/1023.0); //Convert average to
voltageRS_air = ((5.0*1.0)/sensor_volt)-1.0; //Calculate RS in
fresh air
  R0 = RS_air/3.7; //Calculate R0

Serial.print("R0 = "); //Display "R0"
Serial.println(R0); //Display value of
R0delay(1000); //Wait 1 second

}
```

Execution of the Main Program :

```
#include
<ESP8266WiFi.h>
#include <DHT.h>
#include <ThingSpeak.h>

DHT dht(D5, DHT11);
#define LED_GREEN D2
#define LED_YELLOW
D3#define LED_RED D4
#define MQ_135 A0
int ppm=0;
float m = -0.3376; //Slope
float b = 0.7165; //Y-
Intercept
float R0 = 3.12; //Sensor Resistance in fresh air from previous
codeWiFiClient client;

long myChannelNumber = 123456; // Channel id

const char myWriteAPIKey[] = "API_Key";

void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
pinMode(LED_GREEN,OUTPUT);
pinMode(LED_YELLOW,OUTPUT);
pinMode(LED_RED,OUTPUT);
pinMode(MQ_135, INPUT);
WiFi.begin("WiFi_Name", "WiFi_Password");
while(WiFi.status() != WL_CONNECTED)
  {
delay(200);
Serial.print(".");
  }
Serial.println();
Serial.println("NodeMCU is connected!");
Serial.println(WiFi.localIP());
dht.begin();
ThingSpeak.begin(client);
}
```

```
void loop() {
  float sensor_volt; //Define variable for sensor
  voltagefloat RS_gas; //Define variable for sensor
  resistance float ratio; //Define variable for ratio
  int sensorValue;//Variable to store the analog values from
  MQ-135float h;
  float t;
  float ppm_log; //Get ppm value in linear scale according to the the
  ratio valuefloat ppm; //Convert ppm value to log scale
  h = dht.readHumidity();
delay(4000);
  t =
dht.readTemperature();
delay(4000);


sensorValue = analogRead(gas_sensor); //Read analog values of
sensor sensor_volt = sensorValue*(5.0/1023.0); //Convert analog
values to voltageRS_gas = ((5.0*1.0)/sensor_volt)-1.0; //Get value of
RS in a gas
  ratio = RS_gas/R0; // Get ratio RS_gas/RS_air
ppm_log = (log10(ratio)-b)/m; //Get ppm value in linear scale according to the ratio
  valueppm = pow(10, ppm_log); //Convert ppm value to log scale

Serial.println("Temperature: " +
(String) t);Serial.println("Humidity: " +
(String) h);
Serial.println("Our desired PPM = "+ (String) ppm);

ThingSpeak.writeField(myChannelNumber, 1, t, myWriteAPIKey);
delay(20000);
ThingSpeak.writeField(myChannelNumber, 2, h,
myWriteAPIKey);delay(20000);
ThingSpeak.writeField(myChannelNumber, 3, ppm,
myWriteAPIKey);delay(20000);

  if(ppm<=100)
  {
digitalWrite(LED_GREEN,HIGH);
digitalWrite(LED_YELLOW,LOW);
digitalWrite(LED_RED,LOW);
  }
  else if(ppm<=200)
  {
```

```
digitalWrite(LED_GREEN,LOW);
digitalWrite(LED_YELLOW,HIGH);
digitalWrite(LED_RED,LOW);
  }
  else
  {
digitalWrite(LED_GREEN,LOW);
digitalWrite(LED_YELLOW,LOW);
digitalWrite(LED_RED,HIGH);
  }
delay(2000);}
```

AIR POLLUTION MONITORING

(Using Web Development)

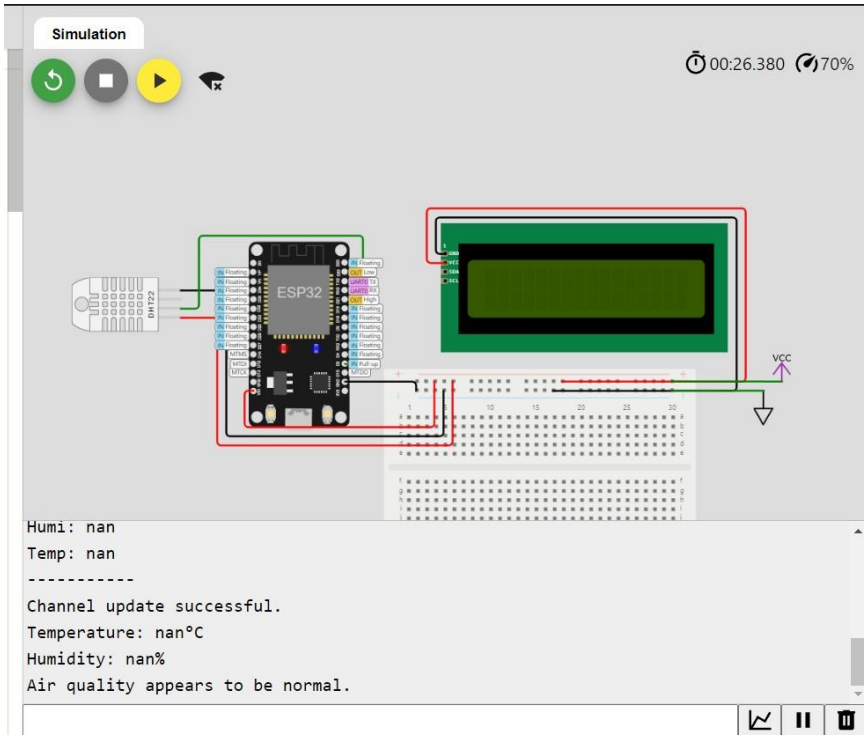
CODE:

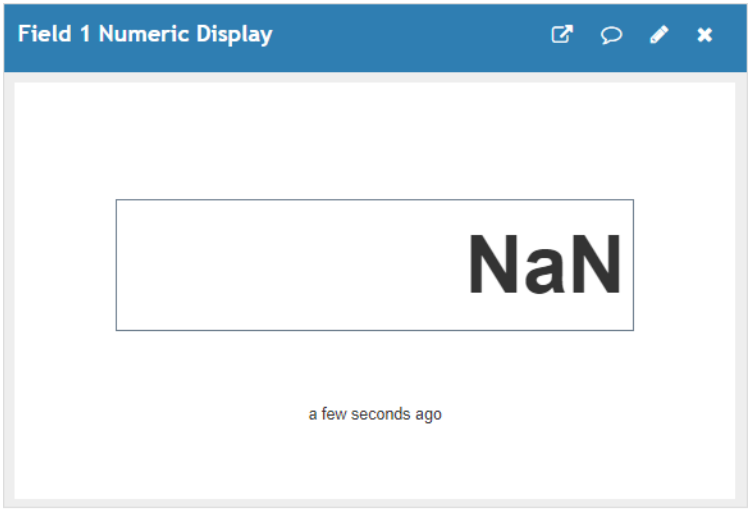
```
#include <DHT.h>
#include <WiFi.h>
#include <ThingSpeak.h>
#include "DHTesp.h"

#define DHT_PIN 2 // Replace with the GPIO pin connected to the DHT22 sensor#define
LED_GREEN_PIN 21 // Replace with the GPIO pin connected to the greenLED bulb
#define LED_RED_PIN 22 // Replace with the GPIO pin connected to the red LEDbulb

char ssid[] = "Wokwi-GUEST";
char pass[] = "";
WiFiClient client;

unsigned long myChannelNumber = 2308799;
const char *myWriteAPIKey =
"Y5D386LU3W5X66Y2";int statusCode;
DHTesp dhtSensor;

int ledGreen =
LED_GREEN_PIN;int ledRed =
LED_RED_PIN;

struct Data {
  float temperature;
  float humidity;
};

Data data;   // Declare a variable to store the datavoid setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA); ThingSpeak.begin(client);
  dhtSensor.setup(DHT_PIN, DHTesp::DHT22);


  pinMode(ledGreen,
  OUTPUT);pinMode(ledRed,
  OUTPUT);
}
```

```
void connectToCloud() {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.print("Attempting to connect"); while
    (WiFi.status() != WL_CONNECTED) {
      WiFi.begin(ssid, pass);
      for (int i = 0; i < 5; i++) {
        Serial.print("."); delay(1000);
      }
    }
    Serial.println("\nConnected.");
  }
}

void computeData() {
  TempAndHumidity sensorData = dhtSensor.getTempAndHumidity();
  data.temperature = sensorData.temperature;
  data.humidity = sensorData.humidity;
  Serial.println(" ------------------- "); Serial.println("Humi: " +
  String(data.humidity)); Serial.println("Temp: " +
  String(data.temperature));Serial.println("");
}

void writeData() { ThingSpeak.setField(1,
  data.humidity);
  ThingSpeak.setField(2, data.temperature);
  statusCode = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);if
  (statusCode == 200)
    Serial.println("Channel update successful.");else
    Serial.println("Problem Writing data. HTTP error code: " +
String(statusCode));
  delay(15000);   // Data to be uploaded every 15 seconds
}

void loop() {
  connectToCloud();
  computeData();
  writeData();

  // Read temperature and humidity
  float temperature = dhtSensor.getTemperature();float
  humidity = dhtSensor.getHumidity();

  // Print the results
  Serial.print("Temperature: ");
  Serial.print(temperature);
```

```
    Serial.println("°C");
    Serial.print("Humidity: ");
    Serial.print(humidity);
    Serial.println("%");

    // Add a condition for air quality
    if (temperature > 25.0 && humidity > 70.0) {
      Serial.println("Air quality might be poor (high temperature andhumidity).");
      digitalWrite(ledRed, HIGH);
      digitalWrite(ledGreen, LOW);
    } else if (temperature < 20.0 && humidity < 30.0) {
      Serial.println("Air quality might be poor (low temperature andhumidity).");
      digitalWrite(ledRed, HIGH);
      digitalWrite(ledGreen, LOW);
    } else {
      Serial.println("Air quality appears to be normal.");
      digitalWrite(ledRed, LOW);
      digitalWrite(ledGreen, HIGH);
    }

    delay(10000);  // Wait for some time before the next measurement (10seconds)
}
```

OUTPUT:




Field 1 Numeric Display
NaN
a few seconds ago

RESULTS ;

The working of the designed prototype has been investigated for the 5 sets of experiments as described in the following sections

EXPERIMENT 1:

Aim:

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

Experimental Condition:

The experiment was performed on a warm sunny day in a localoutdoor area.

Observations in ThingSpeak Cloud:

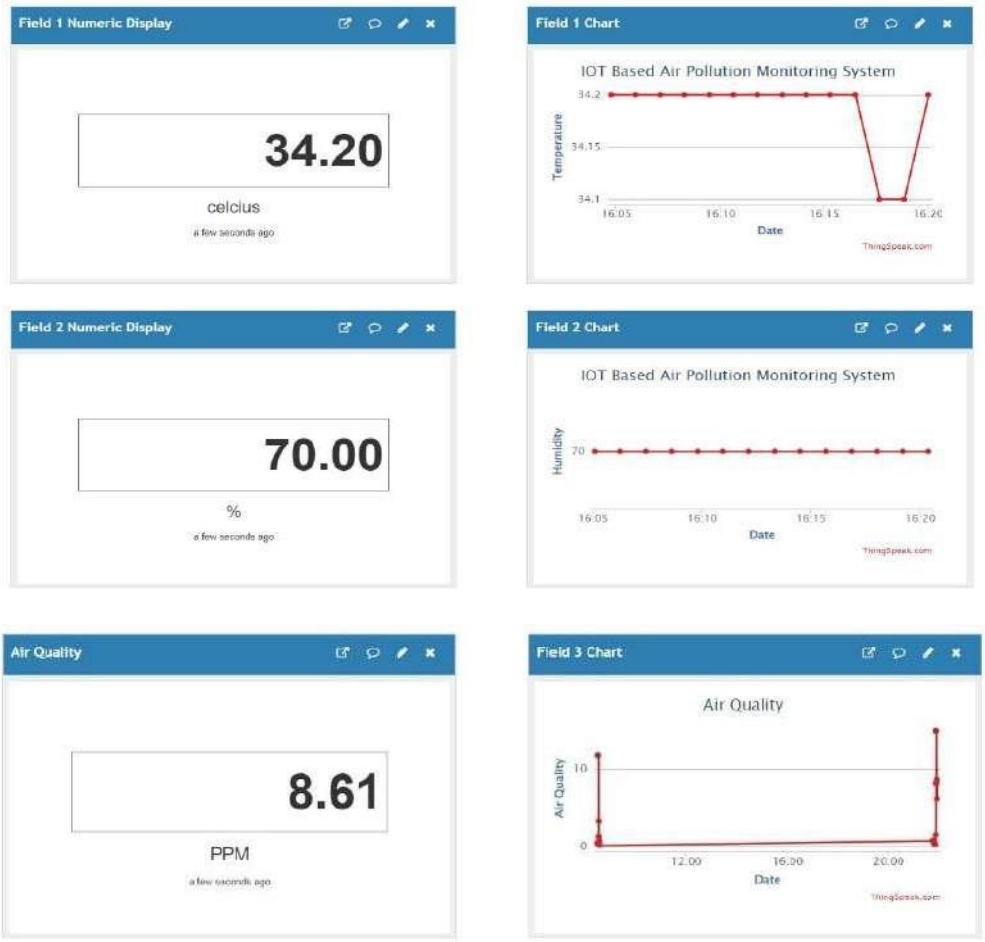


Fig: 5.1 Observations for Experiment 1

conclusion:

We have taken the reference from the Samsung mobile weather app for verifying the values. It matched with a +1.20 error with the temperature data, +5 error with the humidity data and +0.11 error with the PPM data. Hence, we can conclude that the setup has measured the temperature and humidity around the setup area successfully.

EXPERIMENT 2:

Aim:

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

Experimental Condition:

The experiment was performed indoor in the presence of alcoholic gases. Drops of an alcoholic mixture (hand sanitiser) were used to produce alcoholic vapours.

Observations in ThingSpeak Cloud:

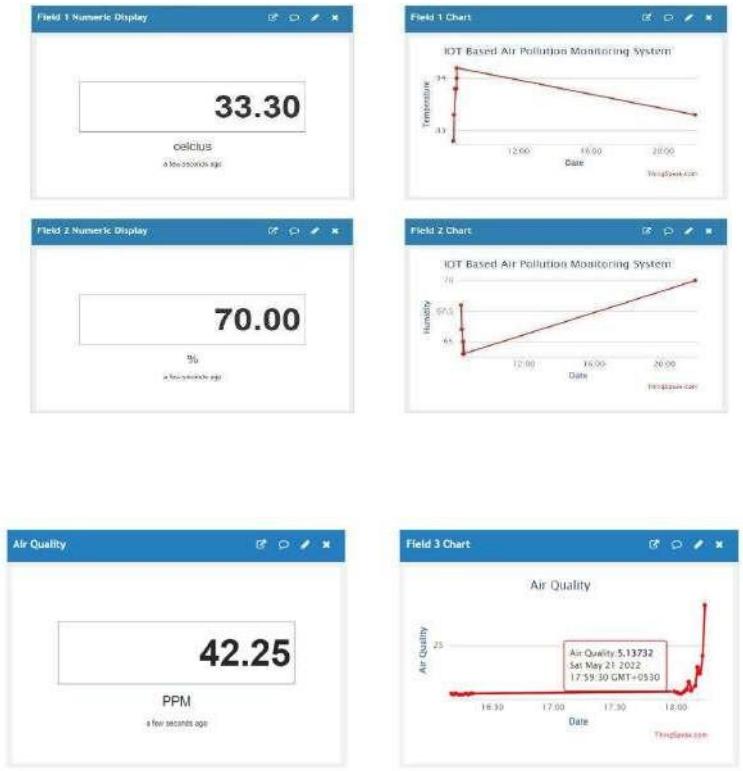


Fig: 5.3 Observations for Experiment 2

Conclusion:

We can observe from the results that the presence of alcohol vapours near the setup can be easily detected by the system. We have taken the reference from the Samsung mobile weather app for verifying the values. It matched with a +1.30 error with the temperature data, +5 error with the humidity data and +0.25 error with the PPM data. Hence, it can be concluded that we can detect the presence of alcoholic vapours with the help of this monitoring system.

EXPERIMENT 3:

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 ThinkSpeak Cloud

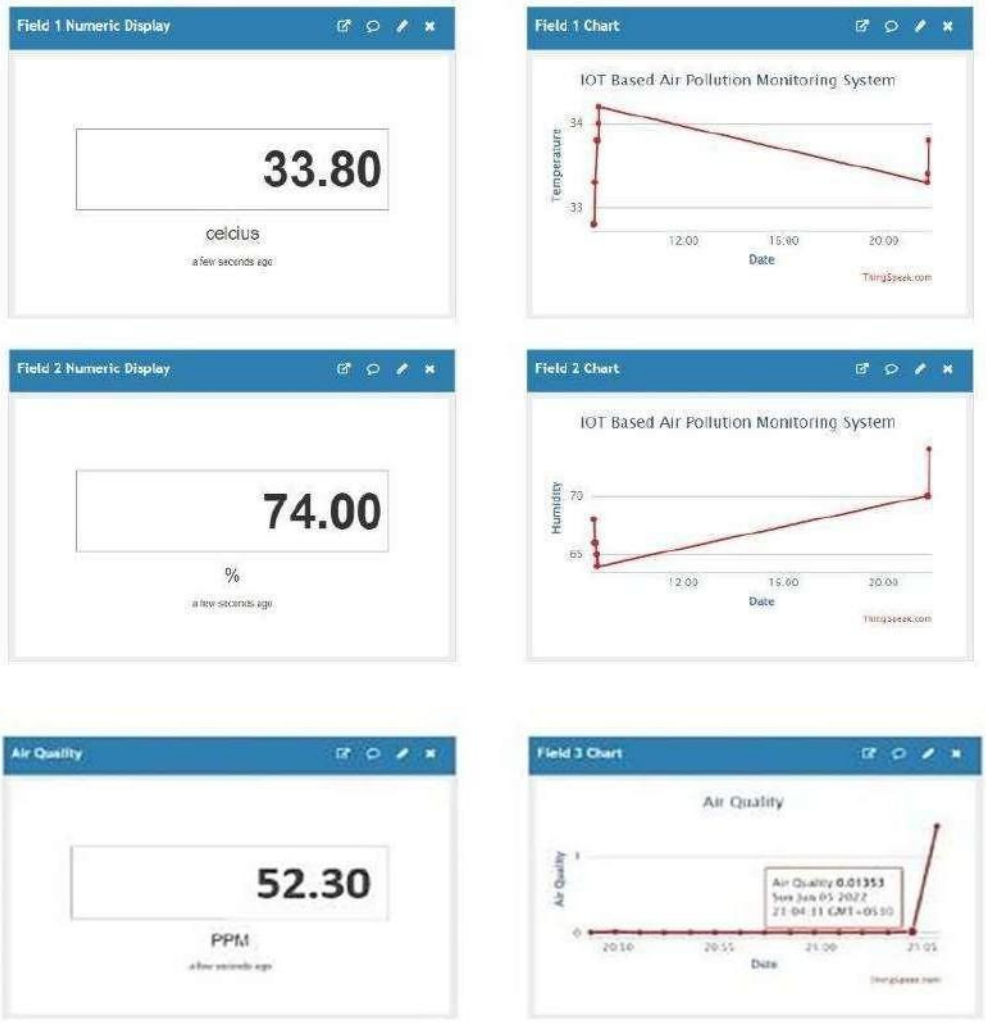


Fig: 5.5 Observations for Experiment 3

Conclusion:

We can observe from the results that the presence of smoke near the setup can be easily detectedby the system. We have taken the reference from the Samsung mobile weather app for verifyingthe values. It matched with a +1.80 error with the temperature data, +4 error with the humidity data and -0.7 error with the PPM data. Hence, it can be concluded that we can detect the presence of smoke with the help of this monitoring system.

EXPERIMENT 4:

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 ThingSpeak Cloud:

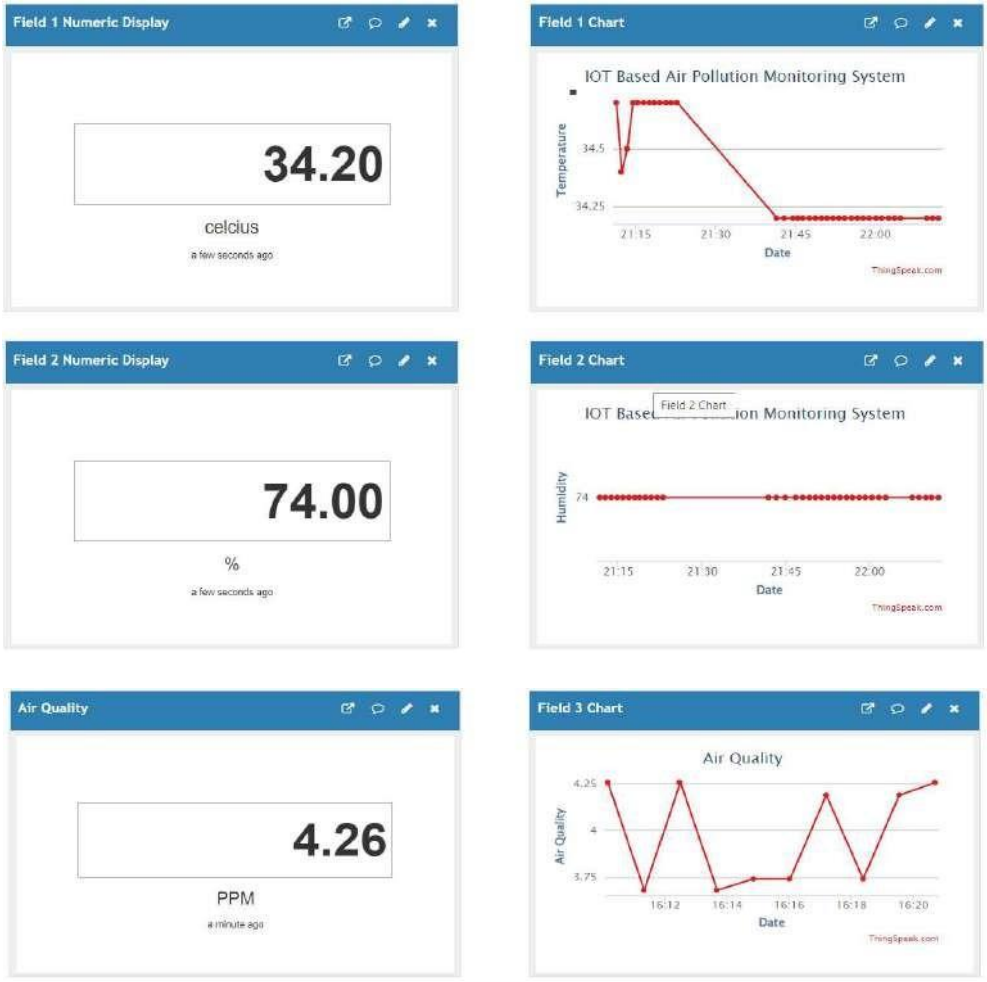


Fig: 5.7 Observations for Experiment

Conclusion:

We have taken the reference from the Samsung mobile weather app for verifying the values. It matched with a +1.20 error with the temperature data, +5 error with the humidity data and -0.08 error with the PPM data. Hence, we can conclude that the setup has measured the temperature and humidity around the setup area successfully.

EXPERIMENT 5:

Aim:

To demonstrate the working of the system in an air-conditioned indoor atmosphere.

Experimental Condition:

The experiment was performed at room temperature.

Observations in ThingSpeak Cloud:

Fig;5.8 observation of Experiment 5

Conclusion:

We have taken the reference from the Samsung mobile weather app for verifying the values. It matched with a +0.6 error with the temperature data, +2 error with the humidity data and -0.03 error with the PPM data. Hence, we can conclude that the setup has measured the temperature and humidity around the setup area successfully.

Experimental Results

Expt. No.	Temperature (in celsius)			Humidity (in %)			Air Quality (in ppm)		
	Project Reading	Samsung Weather App Reading	Error	Project Reading	Samsung Weather App Reading	Error	Project Reading	Samsung Weather App Reading	Error
1	34.2	33	1.2	70	65	5	8.61	8.5	0.11
2	33.3	32	1.3	70	65	5	42.25	42	0.25
3	33.8	32	1.8	74	70	4	52.3	53	-0.7
4	34.2	33	1.2	74	69	5	4.26	4.34	-0.08
5	22.6	22	0.6	59	57	2	0.67	0.7	-0.03