# IOT Based Air Pollution Monitoring System

A Project report submitted in partial fulfilment of the requirements for the degree of B. Tech in Information Technology By,

kalesha M(513221205007)

Under the professors and HOD

Department of Information Technology

THIRUMALAI ENGINEERING COLLEGE
KILAMBI KANCHIPURAM
The Right Place to Enrich Your Career

Department of Information Technology,
Thirumalai Engineering college kilambi.

CERTIFICATE

To whom it may concern

This is to certify that the project work entitled IOT Based Air Pollution Monitoring System is the bonafide work carried out by Kalesha M(513221205007), students of B.Tech in the Dept. of Information Technology, I affiliated to Anna University Chennai , India, during the academic year 2023-24, in partialfulfillment of the requirements for the degree of Bachelor of Information Technology and this project has not submitted previously for the award of any otherdegree, diploma and fellowship.

.

# ACKNOWLEDGEMENT

It is our great fortune that we have got an opportunity to carry out this project work under the supervision of Professor & HOD in the Department of Information and Technology Thirumalai Engineering College , Kilambi ,Tamilnadu , India. We express our sincere thanks and deepest sense of gratitude to our guide for his constant support, unparalleled guidance and limitless encouragement.

We wish to convey our gratitude to the Department of Information and Technology, TEC and the authority of TEC for providing all kinds of the infrastructural facility for the research work.

We would also like to convey our gratitude to all the faculty members and staffs of the Department of Information and Technology Engineering, TEC for their wholehearted cooperation to make this work turn into reality.

## List of Acronyms

| | |
|---|---|
| ○ DHT | Digital Humidity and Temperature |
| IoT | Internet of Things |
| ○ PPM | Parts Per Molecule |
| ○ PM | Particulate Matter |
| ○ CO | Carbon Monoxide |
| ○ $CO_2$ | Carbon Dioxide |
| ○ LED | Light Emitting Diode |
| ○ LPG | Liquid Petroleum Gas |
| ○ IDE | Integrated Development Environment |

## 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.

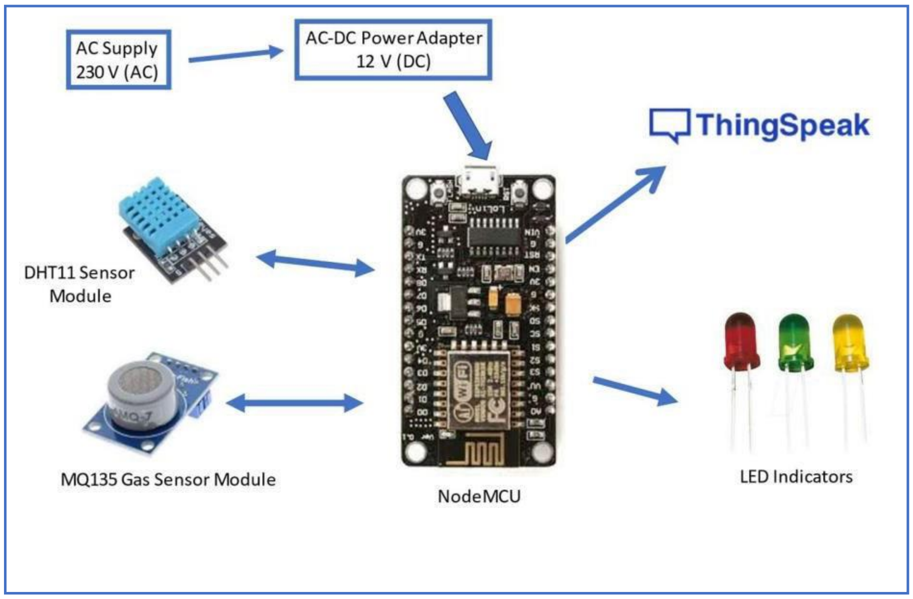


Fig.1. IoT based Air Pollution Monitoring System

Chapter1 INTRODUCTION

## 1.1 Aim of the Project

Air is getting polluted because of the release of toxic gases by industries, vehicle emissions and increased concentration of harmful gases and particulate matter in the atmosphere.

The level of pollution is increasing rapidly due to factors like industries, urbanization, increase in population, vehicle use which can affect human health. Particulate matter is one of the most important parameters having a significant contribution to the increase in air pollution. This creates a need for measurement and analysis of real-time air quality monitoring so that appropriate decisions can be taken in a timely period.

## 1.2 Literature Survey

The explanation of the Air Quality Index (AQI) and its standard ranges are described in [1]. From 0-100 ppm the atmosphere is safe for living. If the ppm level increases above 100 then it moves out of the safety zone. If the ppm value rises above 200 then it becomes extremely dangerous for human life.

The DHT11 sensor module is used to measure the temperature and the humidity of the surroundings [2]. The MQ-135 gas sensor is used to measure the air quality of the surroundings [3]. It can be calibrated with respect to fresh air, alcohol, carbon dioxide hydrogen and methane. In this project, it has been calibrated with respect to fresh air [9], [10].

In [4] the controlling action of NodeMCU has been described. This research has shown the uses of C++ as the programming language for scripting the software code. It has an inbuilt Wi-Fi module which allows the project to implement IoT easily. Arduino IDE is used to implement the coding part of the project [5], [8]. ThingSpeak cloud is used for the cloud service. It has a free version which requires a delay of 15 seconds to upload an entry in the cloud [6], [7]. As this project uses two sensors, both of them have internal heater elements and withdraw more power(P=V*I), so though both sensors are turned ON, their output voltage levels vary and show unpredictable values due to insufficient power drive. So, we used a separate power supply for the sensors as NodeMCU alone is not sufficient to drive two sensors [9].

# Chapter 2

# THEORY & DESCRIPTION OF THE COMPONENTS

## 2.1 What is IOT?

The Internet of Things (IoT) describes the network of physical objects—“things”—that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet. These devices range from ordinary household objects to sophisticated industrial tools.

The field has evolved due to the convergence of multiple technologies, including ubiquitous computing, commodity sensors, increasingly powerful embedded systems, and machine learning.

## 2.2Components 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

## 2.2 Brief Description of the Components

### NodeMCU V3

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.

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

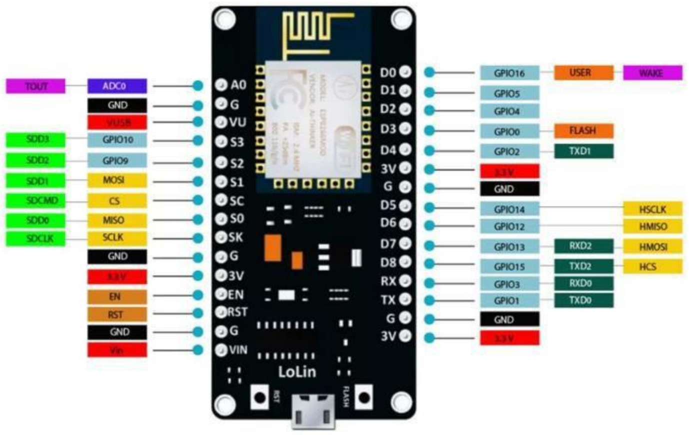


Fig. 2.1 (Pinout Diagram of NodeMCU V3)

### 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).

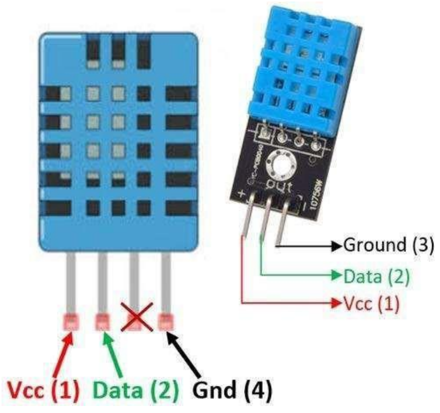


Fig 2.2 (Pinout Diagram of Humidity Measurement: The humidity sensing capacitor DHT11sensor)

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 andthen 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.

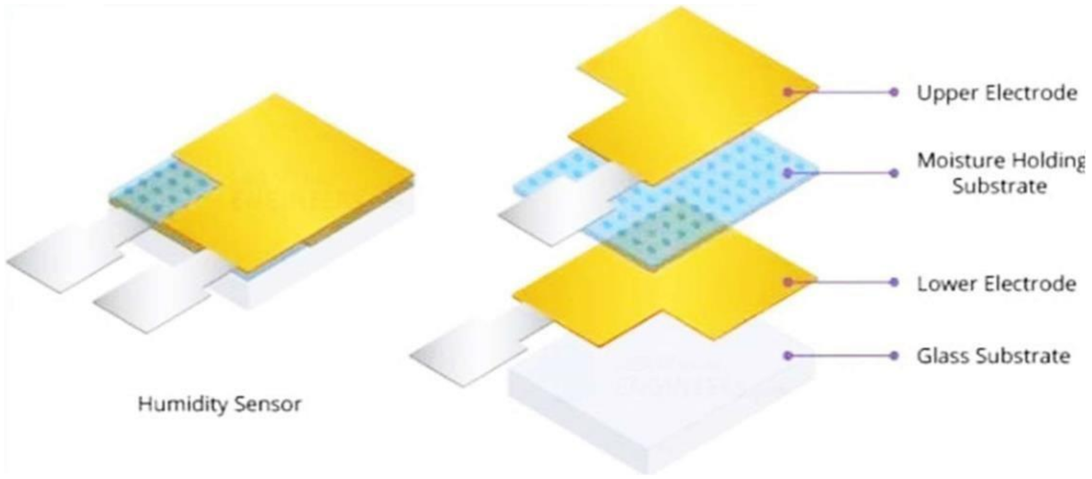


Fig 2.3(The structure of the humidity sensor)

## MQ-135 Gas Sensor Module

The material of MQ135 is $SnO_2$, 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 hasa 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

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 doublesided 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.

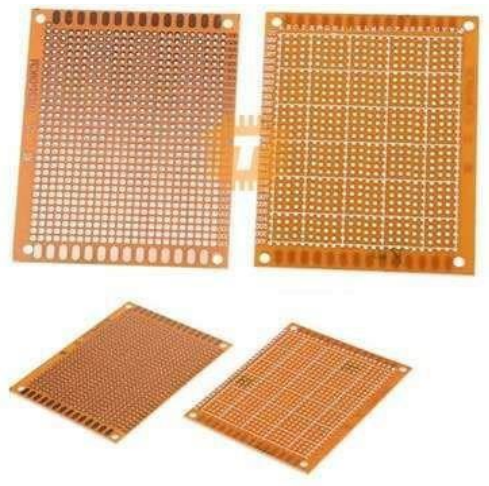


Fig 2.5 Veroboard

- 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 anelectrical component needs. The ACDC power

Fig 2.6 AC-DC Power Adapter

power source.

- 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 lightemitting 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

Fig 2.7LEDs

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

## Resistors

A resistor is a passivetwo-terminalelectrical component that implements electrical resistance as a circuit

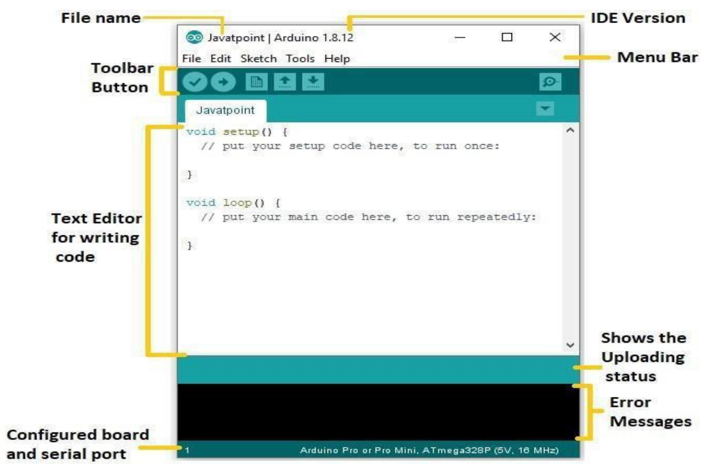
element. In electronic circuits, resistors are used to reduce current flow, adjust signal levels, to divide voltages, bias active elements, and terminate transmission lines, among other uses. High-power resistors that can dissipate many watts of electrical power as heat may be used as part of motor controls, in power distribution systems, or as test

Fig 2.8 Resistors

loads for generators. Fixed resistors have resistances that only change slightly with temperature, time or operating voltage.

###  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.'

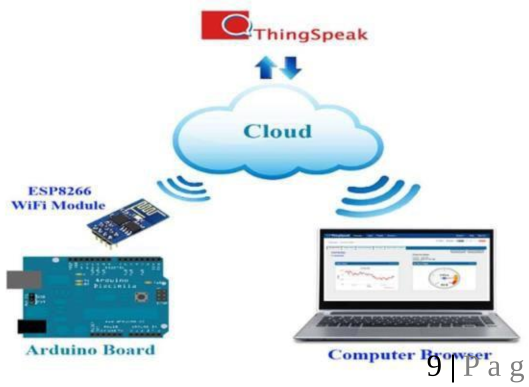

File name
IDE Version
Javatpoint | Arduino 1.8.12
File Edit Sketch Tools Help
Menu Bar
Toolbar Button
Javatpoint
Text Editor for writing code
Shows the Uploading status
Error Messages
Configured board and serial port
1
Arduino Pro or Pro Mini, ATmega328P (5V, 16 MHz)

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Fig 2.9Arduino IDE

### 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

Fig 2.10 ThingSpeak Cloud

from MathWorks, allowing ThingSpeak users to analyse and visualize uploaded data using MATLAB without requiring the purchase of a MATLAB license from MathWorks.

## 2.3 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.

# Chapter 3

# HARDWARE MODEL

## 3.1 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.

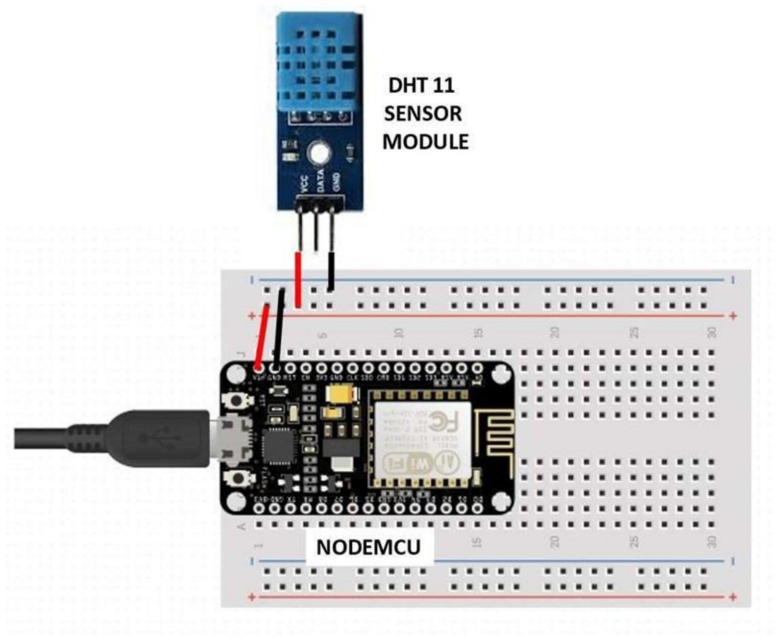


Fig. 3.1 (Circuit Diagram to Preheat the DHT11 sensor module)

## 3.2 Hardware Model to Preheat and Calibrate MQ-135 Gas Sensor 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.

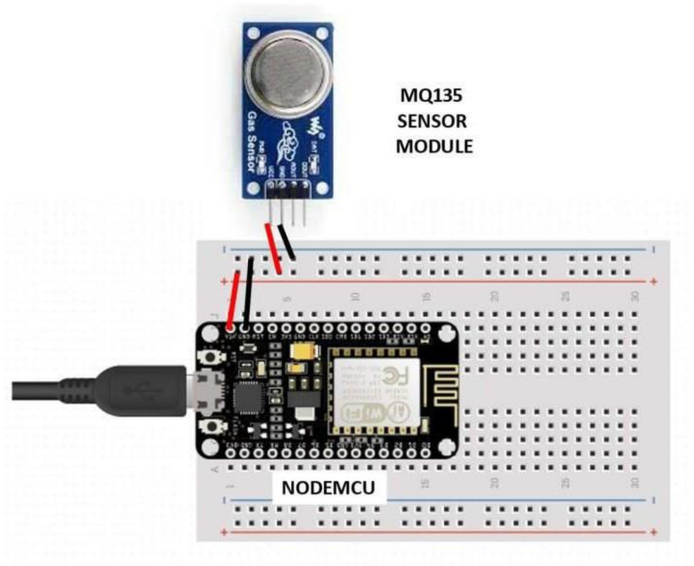


Fig. 3.2(Circuit Diagram to Preheat the MQ-135 Gas sensor module)

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.

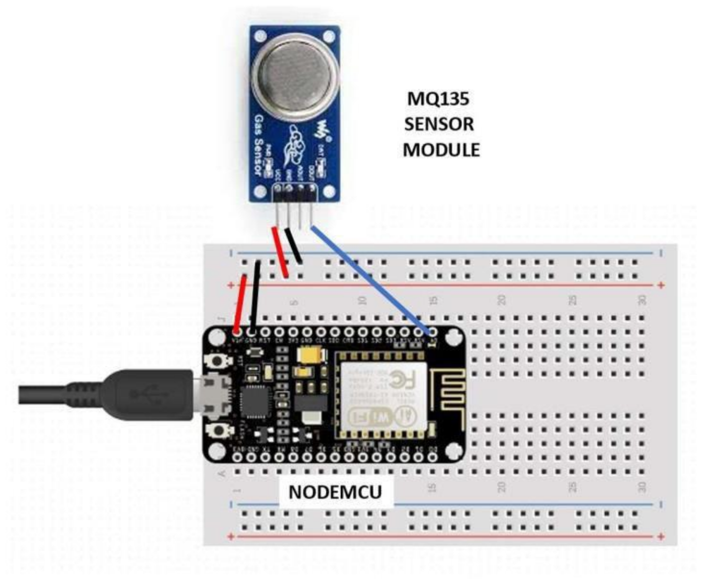
STEP 3 : The analog DATA pin of the MQ-135 gas sensor module was connected with the A0 Pin of the NodeMCU.

STEP 4 : The software code to calibrate the sensor is then uploaded to the NodeMCU and the value of $R_0$in 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)



## 3.3 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 and the 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 with the A0 Pin of the NodeMCU.

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

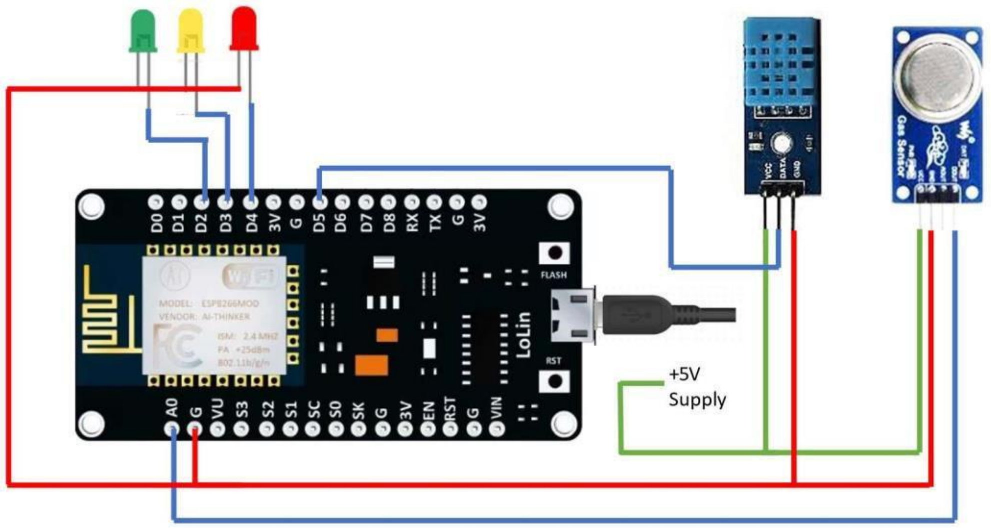
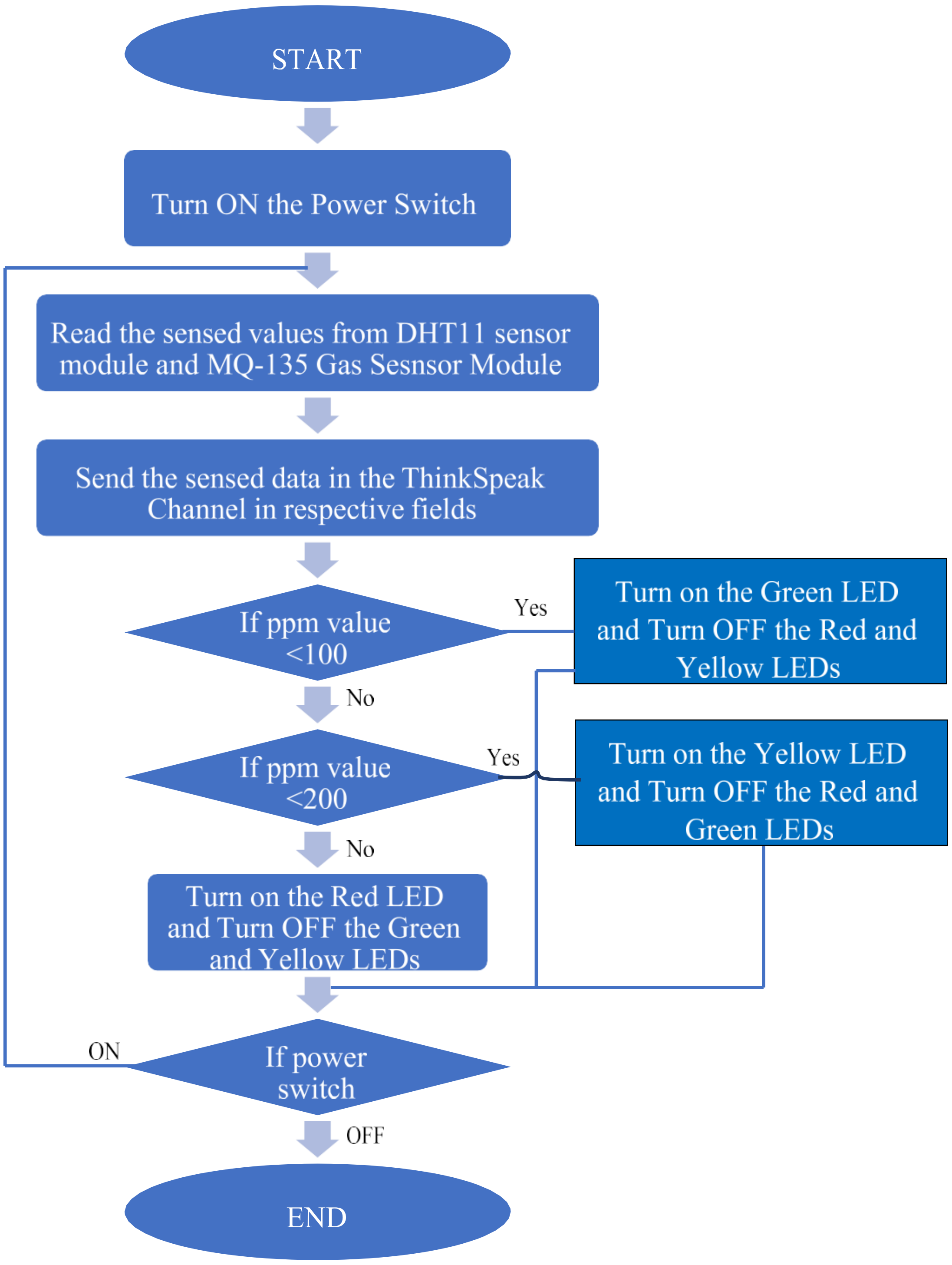


Fig. 3.4(Circuit Diagram of the setup)

# Chapter 4

SOFTWARE IMPLEMENTATION

## 4.1 Working Algorithm


START
Turn ON the Power Switch
Read the sensed values from DHT11 sensor module and MQ-135 Gas Sesnsor Module
Send the sensed data in the ThinkSpeak Channel in respective fields
If ppm value <100
Yes
Turn on the Green LED and Turn OFF the Red and Yellow LEDs
No
If ppm value <200
Yes
Turn on the Yellow LED and Turn OFF the Red and Green LEDs
No
Turn on the Red LED and Turn OFF the Green and Yellow LEDs
ON
If power switch
OFF
END

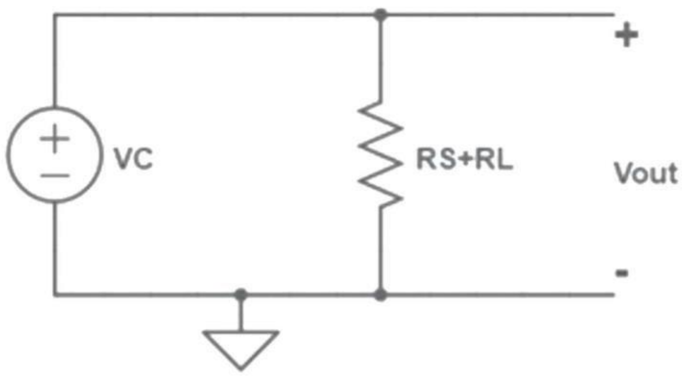


Fig. 4.1 (Internal Circuit diagram of MQ-135 sensor)

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

$$I = \qquad (1)$$

From Fig 4.1, eqn. 1 is equivalent to

$$I = \underline{\qquad} \qquad (2)$$

From Fig 4.1, 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 x R.

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

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

So now we solve for RS:

$$VRLx\ (RS + RL) = VCx\ RL \qquad (5)$$

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

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

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

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

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

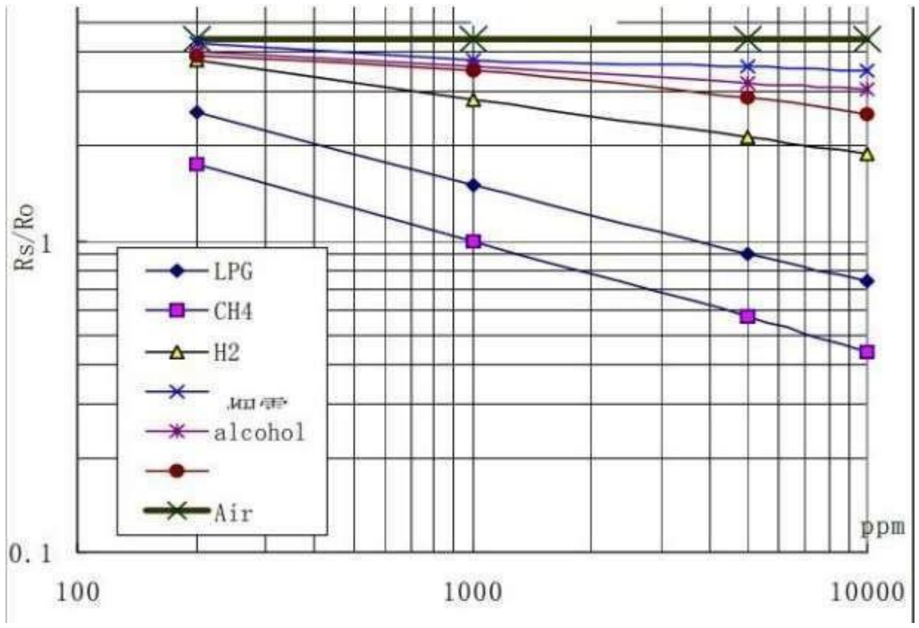


Fig. 4.2 (Graph representing ratio vs ppm variations)

From the graph shown in fig 4.2, we can see that the resistance ratio in fresh air is a constant:

$$RS / R0 = 3.6 \tag{10}$$

Value 3.6 which is mentioned ineqn. 10 is depicted in the datasheet shown in Fig 4.2. 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. Thenwe 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-logscale. From above Figure 4.2, we try to derive the following calculations.

$$y = mx + b \tag{11}$$

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

$$\log_{10}y = m * \log_{10}x + b \tag{12}$$

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:

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

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

$$m = \log(\_\_) / \log( ) \quad (14)$$

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

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

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

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)

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

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

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

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

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

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:

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

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.

 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
voltage    float 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 voltage
RS_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 R0
delay(1000); //Wait 1 second

}
```

## 4.2 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 code

WiFiClient 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 voltage    float
RS_gas; //Define variable for sensor resistance     float ratio; //Define
variable for ratio  int sensorValue;//Variable to store the analog values from
MQ-135    float h; float t; float ppm_log; //Get ppm value in linear scale
according to the the ratio value     float 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
voltage  RS_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 value
ppm = 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);}
```

# Chapter 5

# 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 local outdoor area.

Observations in ThingSpeak Cloud:

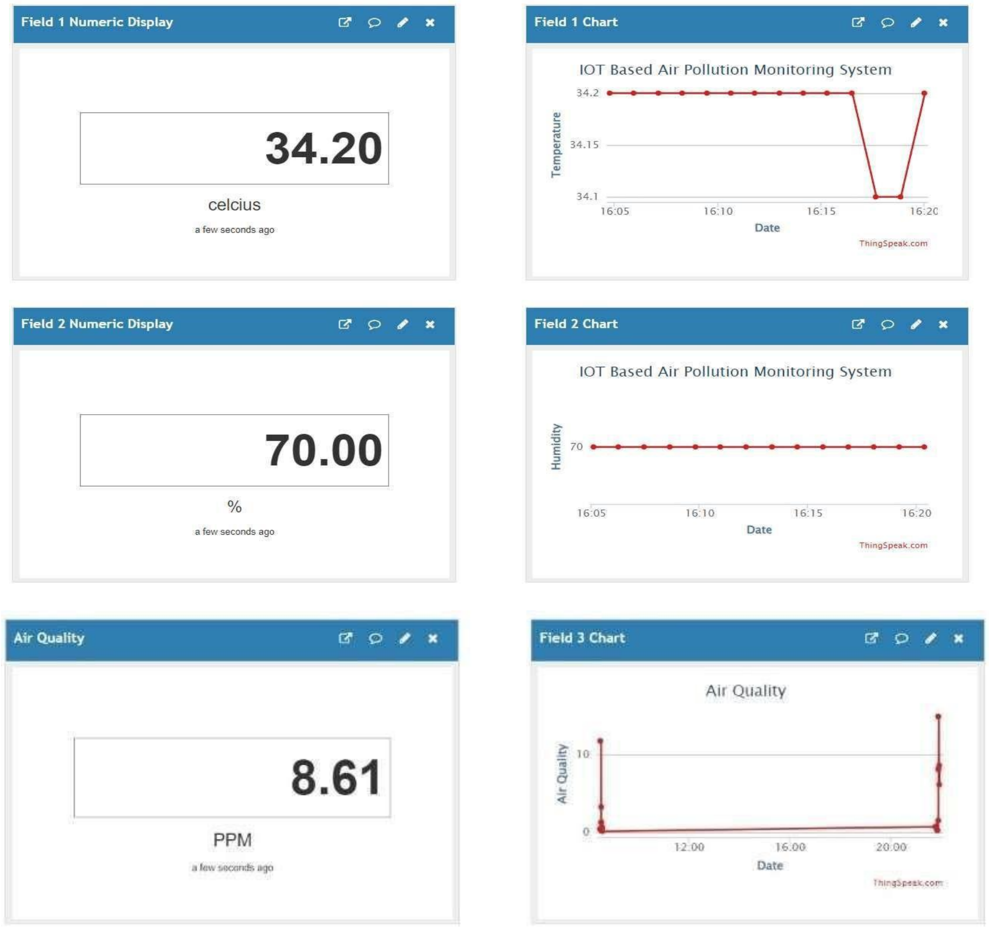


Fig: 5.1 Observations for Experiment 1

## Setup:

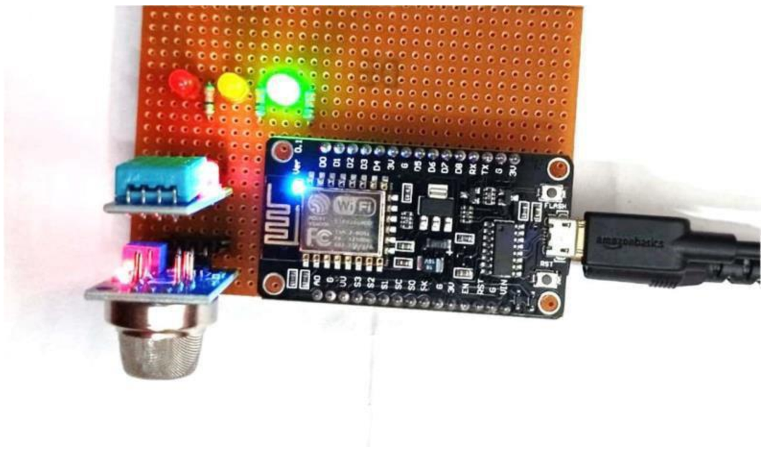


Fig: 5.2 Setup for Experiment 1

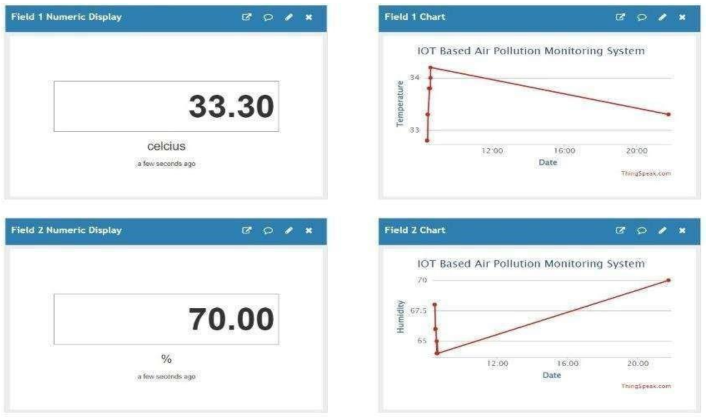
**Conclusion:** We have taken the reference from the Samsung mobile weather app for verifying the values. It matched with a +1.20 error with the temperature data, +5 error with the humidity data and +0.11 error with the PPM data. Hence, we can conclude that the setup has measured the temperature and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

### Observations in ThingSpeak Cloud:



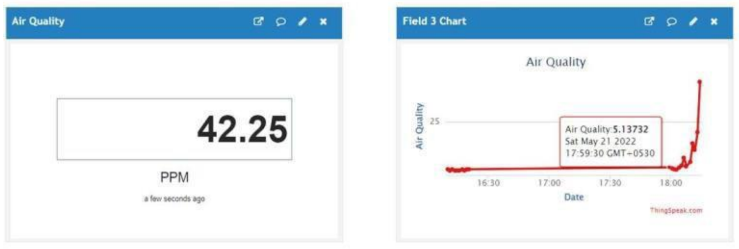


Fig: 5.3 Observations for Experiment 2

## Setup:

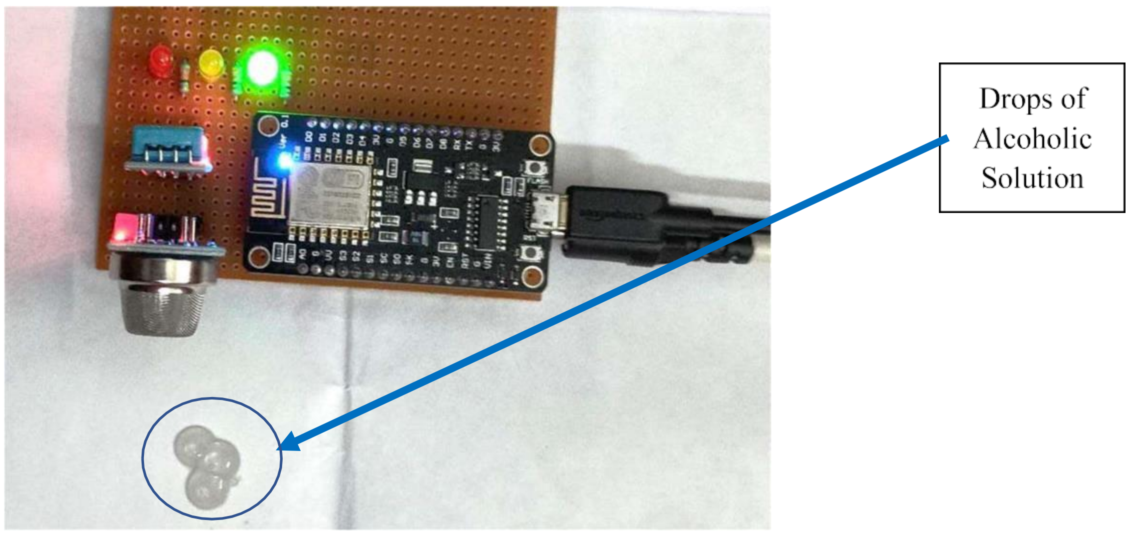


Fig: 5.4 Setup for Experiment 2

## 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

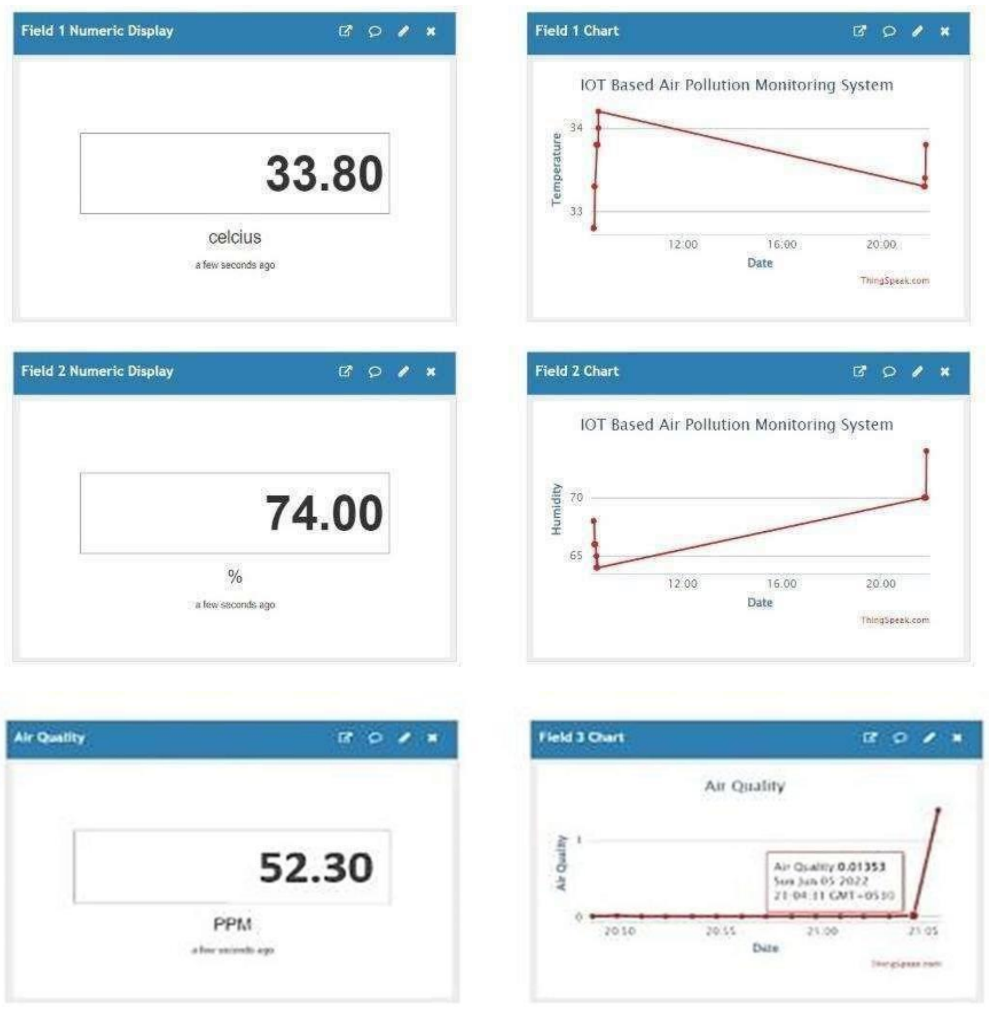


Fig: 5.5 Observations for Experiment 3

## Setup:

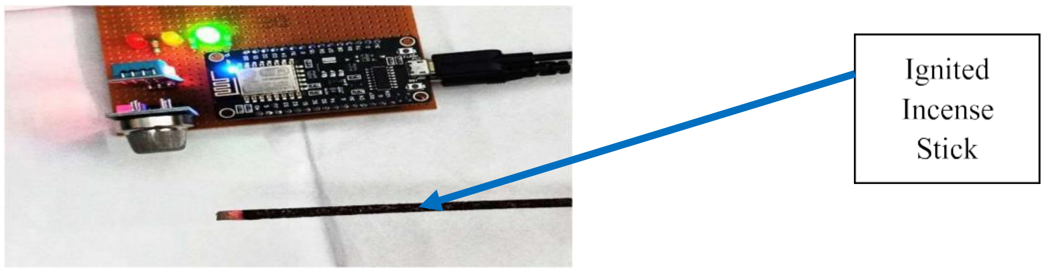


Fig: 5.6 Setup for Experiment 3

Conclusion:

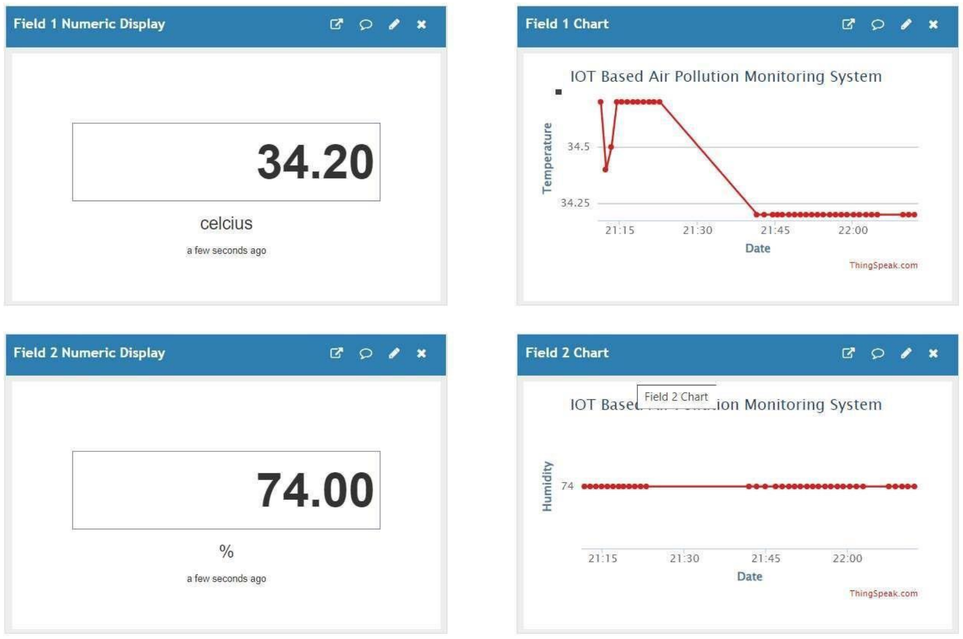
We can observe from the results that the presence of smoke near the setup can be easily detected by the system. We have taken the reference from the Samsung mobile weather app for verifying the values. It matched with a +1.80 error with the temperature data, +4 error with the humidity data and -0.7 error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

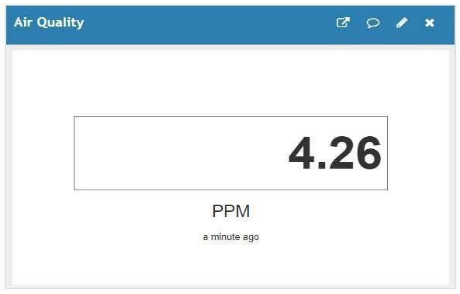
## EXPERIMENT 4:

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:





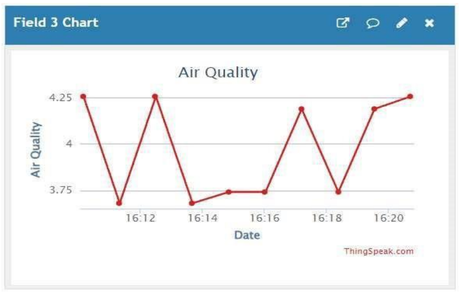


Fig: 5.7 Observations for Experiment 4

## Setup:

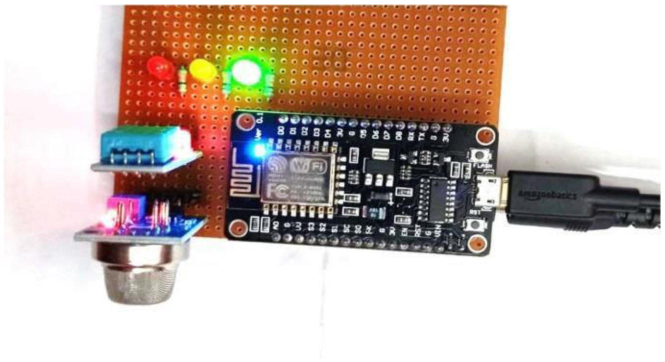


Fig: 5.8 Setup for Experiment 4

## 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:

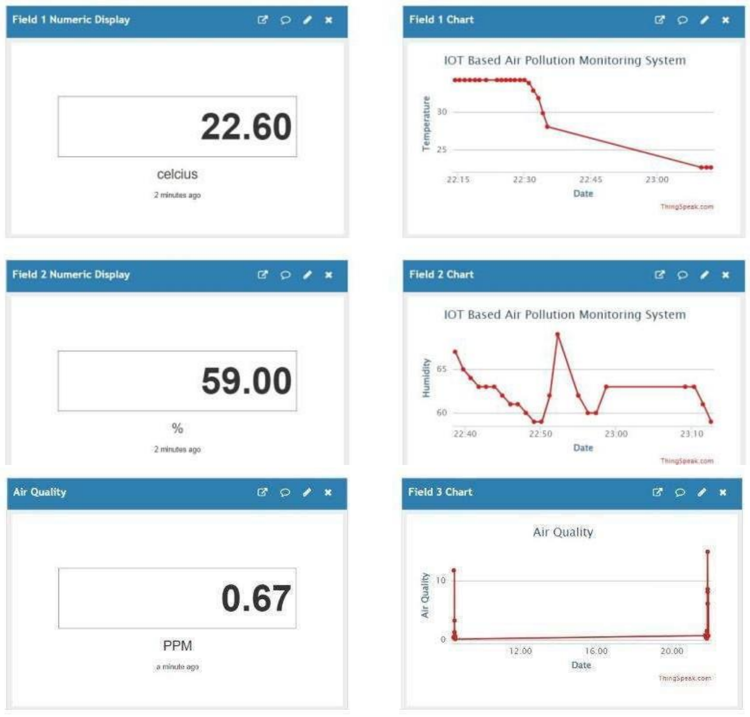


Fig: 5.9 Observations for Experiment 5

## Setup:

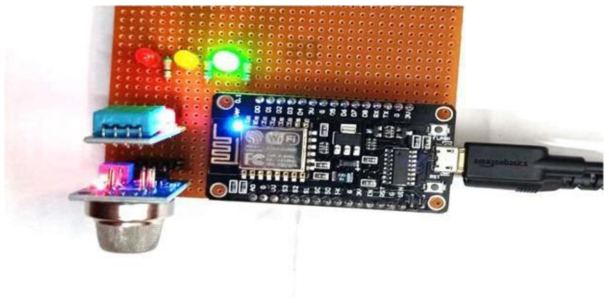


Fig: 5.10 Setup for Experiment 5

## 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 hasmeasured the temperature and humidity around the setup area successfully.

Table 5.1: Experimental Results

| Expt. | Temperature (in celsius) | Humidity (in %) | Air Quality (in ppm) |
|---|---|---|---|

| No. | 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 |

# Chapter 6

# CONCLUSION

In this project IoT based on measurement and display of Air Quality Index (AQI), Humidity and Temperature of the atmosphere have been performed. From the information obtained from the project, it is possible to calculate Air Quality in PPM. The disadvantage of the MQ135 sensor is that specifically it can't tell the Carbon Monoxide or Carbon Dioxide level in the atmosphere, but the advantage of MQ135 is that it is able to detect smoke, CO, CO2, $NH_{4}$, etc harmful gases.

After performing several experiments, it can be easily concluded that the setup is able to measure the air quality in ppm, the temperature in Celsius and humidity in percentage with considerable accuracy. The results obtained from the experiments are verified through Google data. Moreover, the led indicators help us to detect the air quality level around the setup. However, the project experiences a drawback that is it cannot measure the ppm values of the pollutant components separately. This could have been improved by adding gas sensors for different pollutants. But eventually, it would increase the cost of the setup and not be a necessary provision to monitor the air quality. Since it's an IOT-based project, it will require a stable internet connection for uploading the data to the ThinkSpeak cloud. Therefore, it is possible to conclude that the designed prototype can be utilized for air quality, humidity and temperature of the surrounding atmosphere successfully.

# REFERENCES

[ 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 ] Pasha, S. (2016). ThingSpeak based sensing and monitoring system for IoT with Matlab Analysis. International Journal of New Technology and Research, 2(6).

[ 8 ] 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.

[ 9 ] IoT based Air Quality monitoring system using MQ135 & MQ7 with MachineLearning analysis by Kinnera Bharath Kumar Sai M.Tech CSE VIT University, Vellore Subhaditya Mukherjee B.Tech CSE VIT University, Vellore Dr. Parveen Sultana H

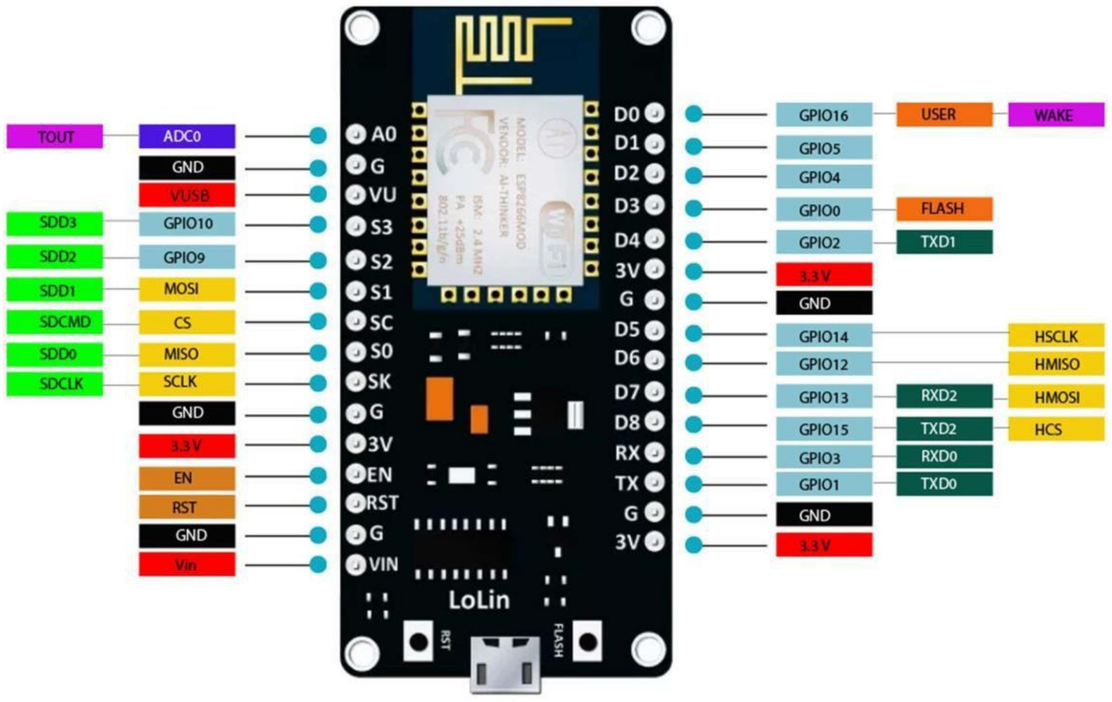
Associate Professor Department of CSE, VIT University.

[ 10 ] https://www.codrey.com/electronic-circuits/how-to-use-mq-135-gas-sensor

# Appendix

## A.1 PIN DESCRIPTION OF NODEMCU

Pinout diagram of the NodeMCU:



Description:

| Pin Category | Name | Description |
|---|---|---|
| Power | Micro-USB, 3.3V, GND, Vin | Micro-USB: NodeMCU can be powered through the USB port<br><br>3.3V: Regulated 3.3V can be supplied to this pin to power the board<br><br>GND: Ground pins<br><br>Vin: External Power Supply |

| | | |
|---|---|---|
| Control Pins | EN, RST | The pin and the button reset the microcontroller |
| Analog Pin | A0 | Used to measure analog voltage in the range of 0-3.3V |
| GPIO Pins | GPIO1 to GPIO16 | NodeMCU has 16 general purpose input-output pins on its board |
| SPI Pins | SD1, CMD, SD0, CLK | NodeMCU has four pins available for SPI communication. |
| UART Pins | TXD0, RXD0, TXD2, RXD2 | NodeMCU has two UART interfaces, UART0 (RXD0 & TXD0) and UART1 (RXD1 & TXD1). UART1 is used to upload the firmware/program. |
| I2C Pins | | NodeMCU has I2C functionality support but due to the internal functionality of these pins, you have to find which pin is I2C. |

## A.2 DESCRIPTION OF SOFTWARE LIBRARIES USED

### ESP8226WiFi Library

The ESP8266WiFi library provides a wide collection of C++ methods (functions) and properties to configure and operate an ESP8266 module.

Commands used are as follows:

- WiFi.begin(“ WiFi Name”, “WiFiPassword”); | Command to connect with WiFi network.
- WiFi.status(); | To check the status of the connection.
  If it returns – WL_CONNECTED | WiFi is connected
  If it returns – WL_IDLE_STATUS | WiFi is connected but no internet found
  If it returns – WL_CONNECT_FAILED | WiFi is not connected

### DHT11 sensor Library

The DHT sensor library provides a wide collection of C++ methods (functions) and properties to configure and operate the DHT11 sensor module.

The commands used are as follows:

- DHT dht(D5, DHT11); | Set the pin for reading data.
- dht.begin();| Command to connect with DHT11 sensor module.
- dht.readTemperature(); | Returns the value of the temperature in Celsius.
- dht.readHumidity(); | Returns the value of humidity in percentage.

### ThinkSpeak Library

The ThinkSpeak library provides a wide collection of C++ methods (functions) and properties to configure and operate the ThinkSpeak cloud.

The commands used are as follows:

- ThingSpeak.writeField(myChannelNumber, 1, t, myWriteAPIKey); । To upload data in the ThinkSpeak Field.

## A.3 COST ESTIMATION OF THE PROJECT

For making the project we have used the following components (as mentioned in Table 2). As per the pricing on the online websites for electronic components, we have formulated a cost estimation.

Table 2: Cost Estimation of the Project

| Components | Price (in Rs) |
|---|---|
| NodeMCU V3 | 288 |
| DHT11 Sensor Module | 120 |
| MQ135 Gas Sensor Module | 135 |
| Connecting Wires | 60 |
| LEDs (Red, Green & Yellow) | 9 |
| AC-DC Power Adapter | 120 |
| Female PCB Berg Terminal and cable | 80 |
| Veroboard | 100 |
| Breadboard | 70 |
| Total | 982 |