Abstract: The term "internet of things" is used to describe a worldwide system of connected, self-aware devices that may gather data about their surroundings and share it with other networks (IOT). Since more people are breathing in polluted air due to growing rates of car usage, industrialization, and urbanisation, it poses a serious threat to public health. To combat this, a cutting-edge Internet of Things (IoT)-based air pollution monitoring system has been developed. This system is able to identify and quantify dangerous gases and compounds such carbon dioxide, nicotine, alcohol, benzene, nh3, and no2. Sensors connected to an esp32 microcontroller collect data on the surrounding environment and send it to the cloud in real time using the MQTT protocol. Accessible from anywhere with an Internet connection, the data is shown on both a website and an LCD. This system has the potential to provide employees real-time information about the source and location of any pollution. By providing precise data on air quality and issuing alerts when pollution levels reach over acceptable limits, an IoT-based air pollution monitoring system has the potential to decrease threats to public health.
Keywords: Internet of Things (IoT), Smart Devices, Air pollution, Air quality sensors, Microcontroller, MQTT protocol, Cloud platform, Remote monitoring, Real-time data.

1 Introduction

One of the most promising applications of the Internet of Things (IoT), which is changing the way people interact with technology, is in the field of air quality monitoring. Air pollution is the most pressing problem, because it has consequences for both the natural world and for human health. Rising populations, industrial activities, and transportation contribute significantly to air pollution worldwide. To better monitor air quality in real time and alert nearby workers and residents when pollution levels rise over safe thresholds, an Internet of Things (IoT)-based system is proposed. A microcontroller named ESP32 controls the device's temperature, humidity, and air quality sensors. The MQTT standard paves the way for information to be saved in the cloud. A web interface and an LCD screen provide remote monitoring capabilities. Nicotine, alcohol, benzene, NH3, and NO2 are just some of the compounds that the system might potentially identify. In theory, the monitoring system's findings may be useful in a variety of contexts. Possible benefits include more precise localization of polluted areas, tracking of the effectiveness of pollution-reduction programmes, and simpler access to reliable data on air quality Analyzing monitoring data allows us to calculate the daily air pollution level, allowing us to take the appropriate measures. The Internet of Things-based air pollution monitoring system is a viable alternative for businesses that must maintain tabs on pollution levels for purposes of employee safety because of its dependability and cheap cost. The technology might be used to protect not just commercial buildings, but also residential areas, parks, and other public areas. It's likely that Internet of Things-based air pollution monitoring devices hold the key to figuring out how to solve the massive issue of air pollution and its devastating impact on the ecosystem and human health. The ability of these technologies to identify dangerous substances in real time and provide relevant data has the potential to improve air quality and protect human health. It's important to get the word out about them and take steps to make them more affordable and accessible to the general population.

2 Objective

Air quality data are collected and analyzed as part of air pollution detection and monitoring to determine where and how much pollution is present. Our mission is to improve air quality and preserve human health by sharing accurate and timely information with politicians, environmental groups, and the general public. By keeping tabs on where pollution levels are highest, we can cut down on emissions from vehicles and companies. To further minimize pollution and encourage the use of eco-friendly technology, this data might also be used to draught laws and direct the creation of regulations. The identification and frequent monitoring of air pollution is crucial to achieving the goals of sustainable development, public health, and climate change mitigation.

3 Literature survey

Standard AQI values and more information are available here. Anywhere between 0 and 100 ppm is safe for human habitation. The dangerous threshold is reached if the ppm level rises over 100. When the concentration rises over 200 parts per million, it poses a significant risk to human health [1]. The DHT11 sensor module can detect both the temperature and humidity of its surroundings. The MQ-135 gas sensor is used to evaluate air quality. Oxygen-rich air, alcohol, carbon dioxide, Page hydrogen, and methane are just some of the gases and liquids that may be utilised to alter it. In this study, the quality of the surrounding air serves as a standard [2]. It has been shown that Node MCU exhibits dominant behaviour. This study demonstrates how useful a scripting language C++ may be for programmers. Because of its built-in Wi-Fi module, it's easy to incorporate IoT into preexisting software. The coding for this project is done in the Arduino IDE. The cloud service is provided by Thing Speak. There is a free version, but it adds 15 seconds to the time it takes to upload a file to the cloud [3]. Due to the power drive, the sensors' output voltage levels vary and display unpredictable behaviour even when both are switched ON; this is because the project employs two sensors, each of which incorporates internal heater components and hence consumes more power ($P=V*I$). Since the Node MCU can only power one sensor at a time, we needed to find a different way to power them [4].

4 System requirements

Hardware Components - LEDs that emit green, yellow, and red light, AC-DC adapters, resistors, and sensor modules like the Node MCU V3, the DHT11, and the MQ-135 [5].

Technology: Think Speak Cloud and the Arduino Software Development Kit. Node MCU V3 - The Node MCU V3 development board runs on open-source software and is based on the ESP8266. The built-in USB connector simplifies the process of creating and debugging IoT apps. It contains GPIO (general-purpose input/output) pins for easy connection to various gadgets and sensors.

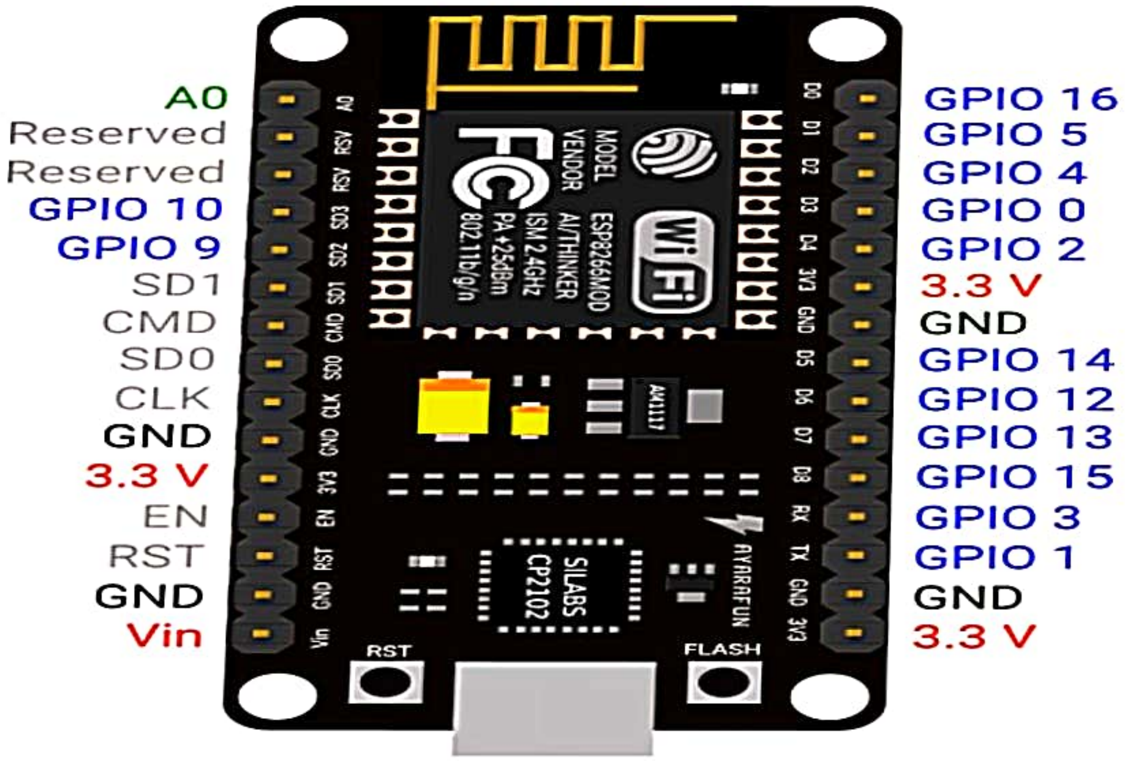


Figure 1 Node MCU V3

DHT11 Sensor Module -The DHT11 is a digitally output voltage-based temperature and humiditysensor. A thermistor and a capacitive humidity sensor are used to examine the surrounding air. The Vcc pin must be connected to 5V DC and ground (GND), as shown in Fig. 2. It is easier to interpret the sensor's output voltage when using the Data pin in digital mode. Comparing Dry and Wet Conditions The humidity sensing capacitor is shown in Figure 2; it comprises of two electrodes and a dielectric substrate that may trap moisture between them. Humidity has an effect on the value of capacitance. The IC tracks the resistance and translates the data into a digital format as it fluctuates. A thermistor with a negative temperature coefficient is used to measure temperatures using the DHT11 sensor. The resistance of this thermistor drops as its temperature increases. The sensor's semiconductor ceramics or polymer construction enables it to measure resistance changes across a broad dynamic range.

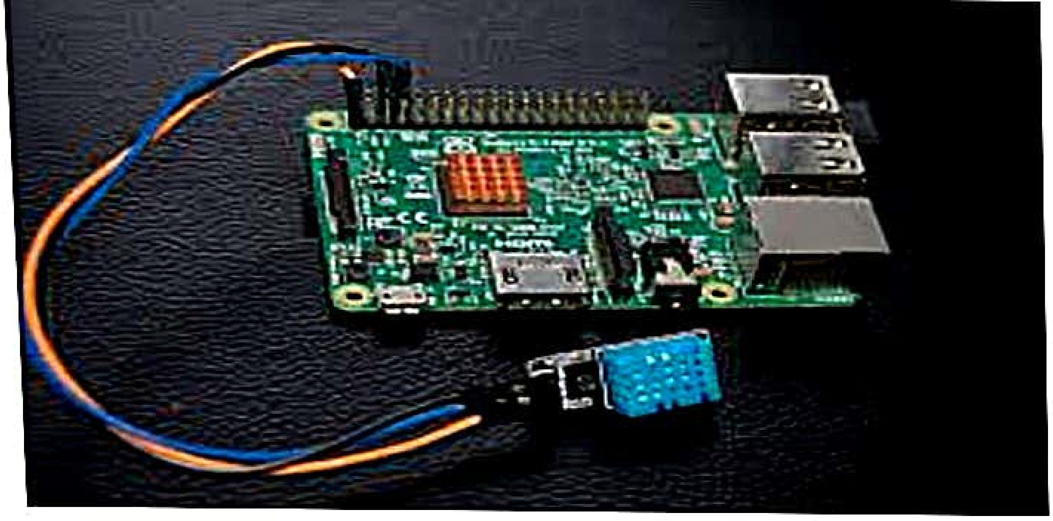


Figure 2 DHT11 Sensor Module

MQ-135 Gas sensor Module- Ammonia, nitrogen oxides, sulphor dioxide, benzene, smoke, and carbon dioxide are just some of the gases that may be detected by the MQ-135 gas sensor module. The idea is based on the observation that the resistance changes when the target gas is present. The module's analogue output corresponds to the observed concentration of gas. Among the various applications it has found are in security systems, air quality monitoring, and pollution management.

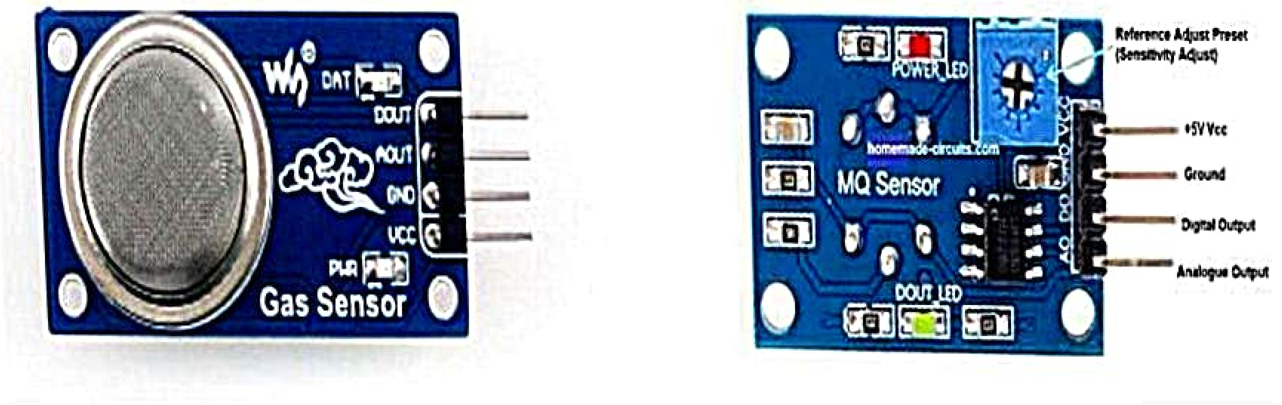


Figure 3 MQ-135 Gas sensor Module

Veroboard- Veroboard (KS100) is a kind of PCB used for electrical circuit prototyping. Connections may be made by trimming the copper strips that separate the rows of copper pads. Because of how easily it can be tailored to meet a variety of aesthetic needs, it is often used for quick jobs.

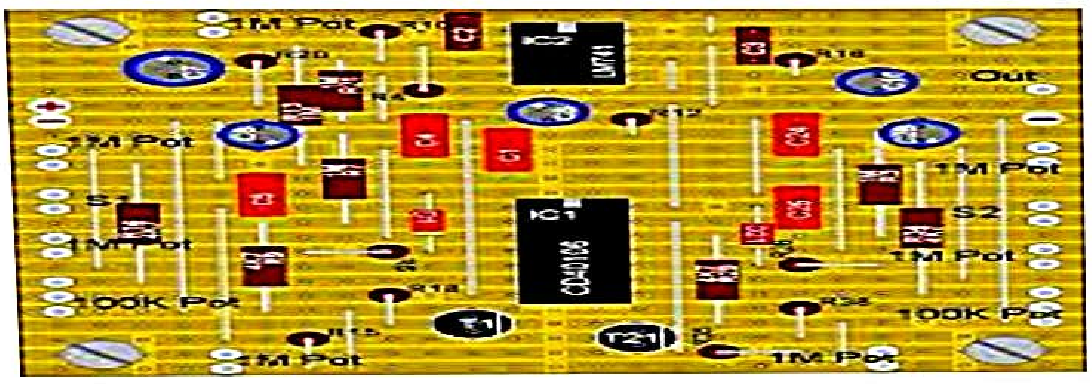


Figure 5 Veroboard

Breadboard- A breadboard is a tool used to prototype electrical circuits without solder. The board's slotted shape makes it easy to connect the different electrical pieces by providing a snug fit for the component leads. Breadboards are useful for quick development and testing of circuits since they do not need soldering or other permanent connections.

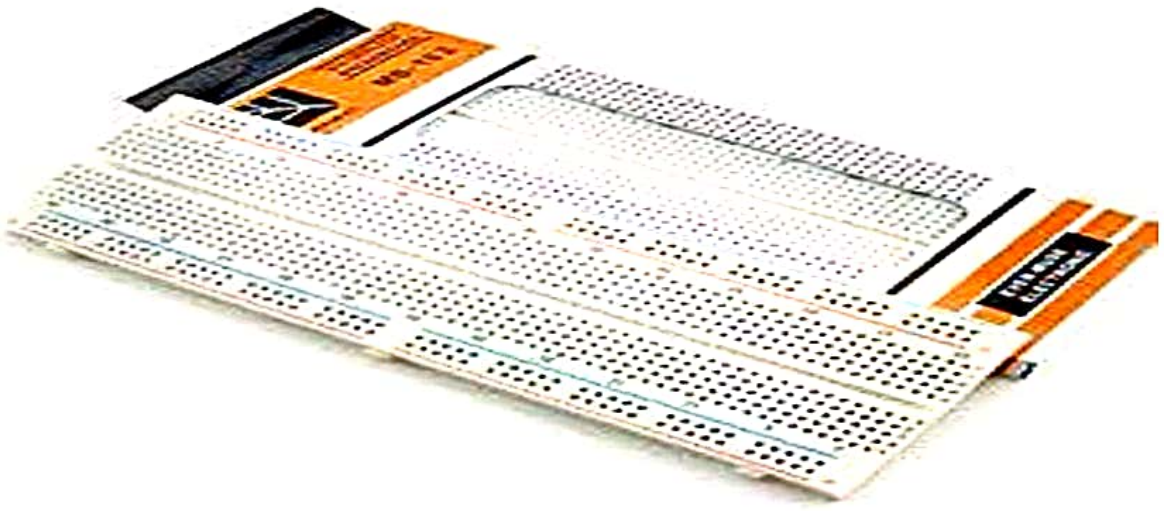


Figure 6 Breadboard

AC-DC Adapters –A breadboard, a solderless tool, is used to prototype electrical circuits. The electrical components may be connected quickly and easily by simply inserting the component leads into the appropriate holes on the board. Because they don't need solder or other permanent connections, breadboards are ideal for quickly prototyping and testing circuits.

Figure 7 AC-DC Adapters

LED emitting green, yellow and red colors Green, yellow, and red LEDs are often used as status indicators in electronics, with each color serving a somewhat different purpose. In many cases, a green light indicates that everything is running well, whereas a yellow light serves as a warning or indicates that there may be a problem. Indicators of electricity and traffic utilize these hues often as visual status indicators.

Figure 8 LED light

Resistor – In an electrical circuit, a resistor limits how much current can pass. Its major function is to protect electrical components from being damaged by electrical current. The ohm is a universal unit of resistance measurement. Electrical circuits often use voltage dividers, current limits, and signal conditioners.

Figure 9 Resistor

Arduino IDE - This open-source software may be used to programme the Arduino board. For building and uploading programmers to the board, it provides a user interface. The IDE supports a simplified version of C++ and includes a library of pre-written scripts for commonly performed tasks. A serial monitor is also provided, which may be used to investigate data sent by the board and troubleshoot the code.

Thing Speak Cloud – With the help of the open-source, cost-free application Thing Speak, gadgets may communicate with one another online. Developers used Ruby to construct it. It offers an application programming interface (API) that mobile devices and social networking sites may utilise to access, retrieve, and record data more easily. Thing Speak was made available by io Bridge in 2010 to assist with IoT applications. Customers may analyse and display data by using Thing Speak integrated within MathWorks' C++ without purchasing a separate copy of C++.

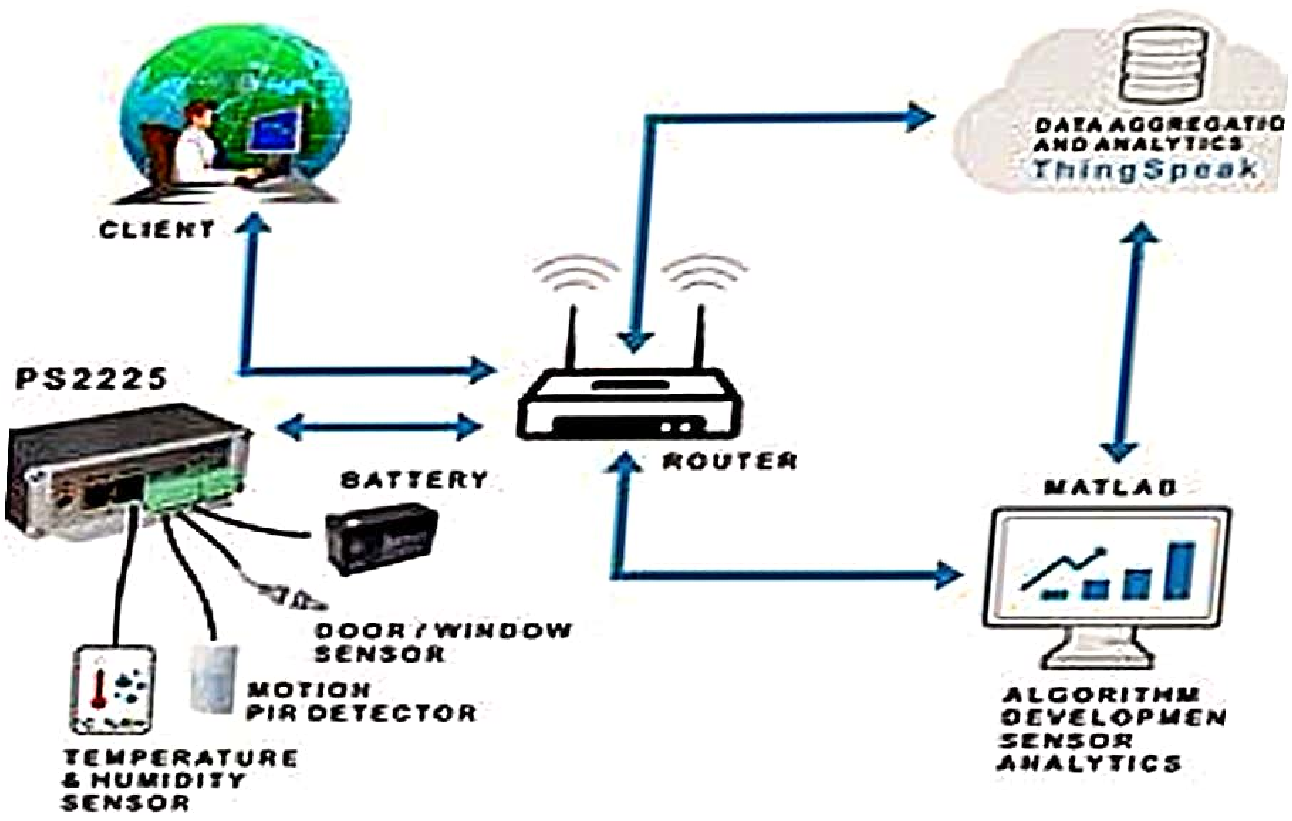


Figure 10 Thing Speak Cloud

5 Working Procedure and Hardware Model

Sensor data is read by the system's central controller, a Node MCU, and then displayed through LEDs. The MQ-135 is a part-per-million air quality sensor, while the DHT11 is a meteorological station. LEDs indicate the current security status and data is sent to the Thing Speak server. After the MQ-135 is calibrated for 24 minutes, the DHT11 is heated for 10 minutes. When the complete working code for the Node MCU is available, the hardware circuit may be constructed from the necessary components [6-8].

The DHT11 and MQ-135 gas sensor modules are calibrated and warmed up with the help of the hardware model. After powering the Node MCU with 12V DC for 20 minutes, the DHT11 sensor is preheated by connecting its Vcc and Gnd pins to the VU and Gnd pins of the Node MCU, respectively.

Connect the MQ-135 gas sensor module's Vcc, Gnd, and analogue DATA wires to Node MCU's VU, GND, and A0 pins, then run Node MCU on 12V DC for a day to bring it up to temperature and calibrate it. Both modules have their Vcc pins connected to the 5V adaptor on the Veroboard, and their LEI indication cathodes connected to the Node MCU's D2, D3, and D4 pins, as well as the MQ-135' analogue DATA line's A0 pin, the DHT11's DATA pin, and the Node MCU's Gnd pin. An AC-DC converter supplies the required 9V DC, and the Node MCU is pre-loaded with the necessary instructions [8-10].

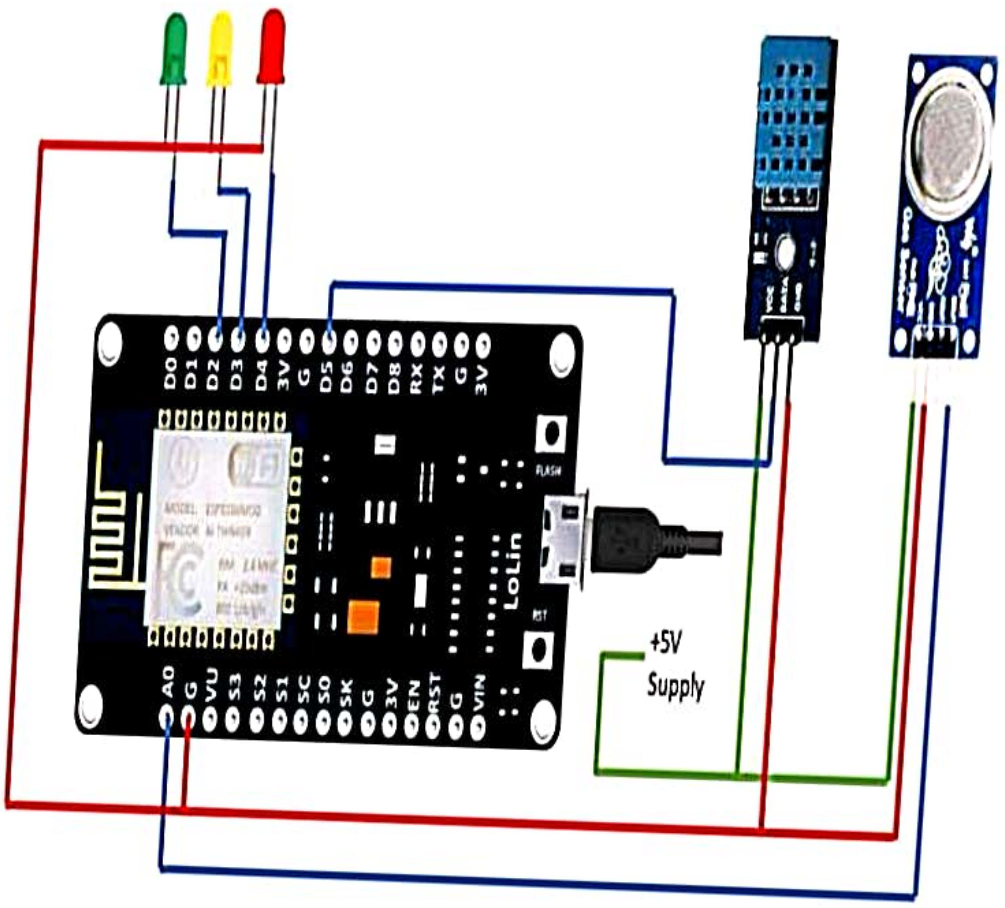


Figure 11 Hardware Model

6 Calibration of MQ-135 Gas Sensor Module

Calibration theory states that exposing the sensor to a clean air environment is crucial. Develop a formula to reduce the sensor's voltage reading to microvolts (parts per million). The following conclusion may be drawn from these data points: In theory, the most important step is to calibrate the sensor in ambient air. Determine a percentage value (ppm) for the sensor's voltage measurement (parts per million). With these figures, we may learn [11, 12]:

7 SOFTWARE IMPLEMENTATION

Algorithms

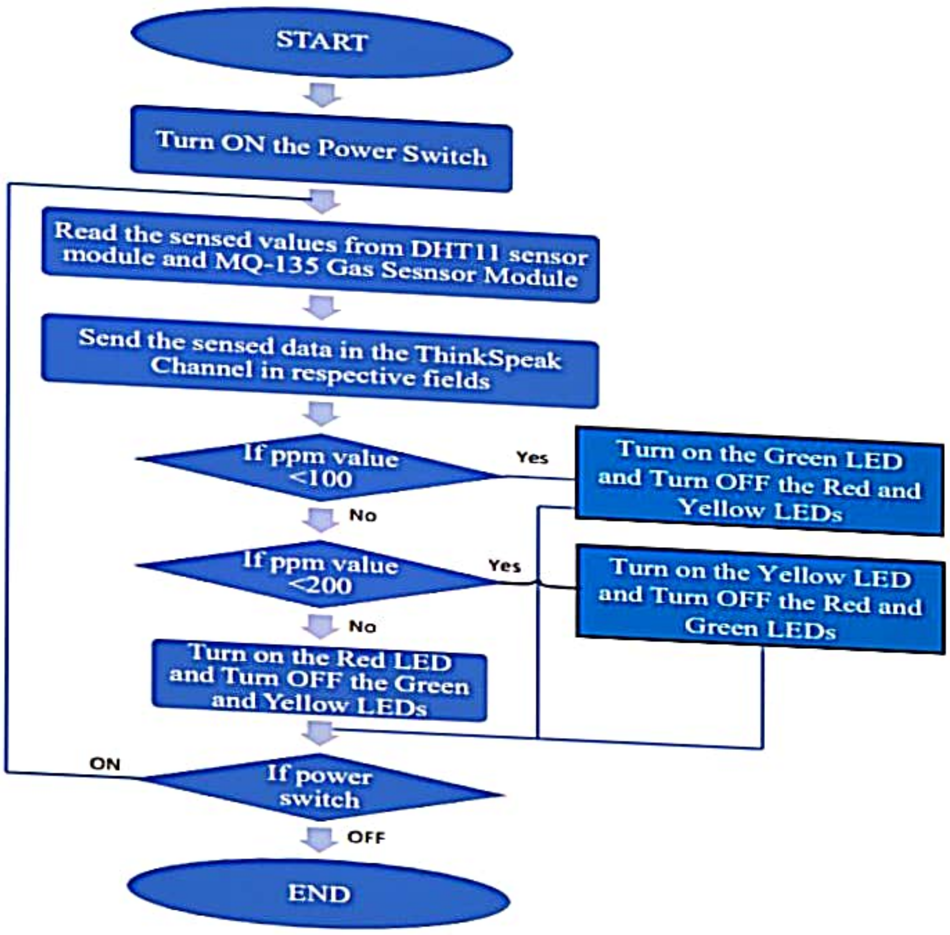

START
Turn ON the Power Switch
Read the sensed values from DHT11 sensor module and MQ-135 Gas Sesnsor Module
Send the sensed data in the ThinkSpeak Channel in respective fields
If ppm value <100
Yes
Turn on the Green LED and Turn OFF the Red and Yellow LEDs
No
If ppm value <200
Yes
Turn on the Yellow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

Figure 12 Flow Chart of Software Implementation

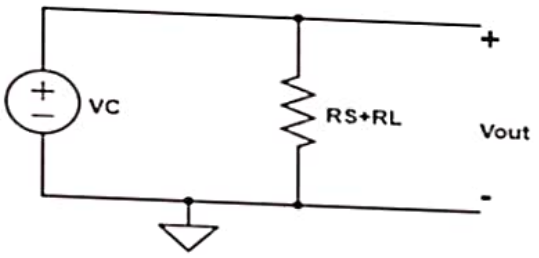


Figure 13 Calibration of MQ-135 Gas Sensor Module

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

$I = V/R$ (1)

From Fig 11, eqn. 1 is equivalent to

$I = Vc/RS + RL$ (2)

From Fig 10, we can obtain the output voltage at the load resistor using the value obtained for I and Ohm's Law at a constant temperature

$V = I \, x \, R$ (3)

$VRL = [VC/(RS + RL)] \, x \, RL$ (4)

$$VRL = [(VC * RL)/(RS + RL) \tag{5}$$

So now we solve for RS:

$$VRL \; x \; (RS + RL) = VC \; x \; RL) \tag{6}$$

$$(VRL \; x \; RS) + (VRL \; x \; RL) = VC \; x \; RL \tag{7}$$

$$VRL \; x \; RS = (VC * RL) - (VRL * RL) \tag{8}$$

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

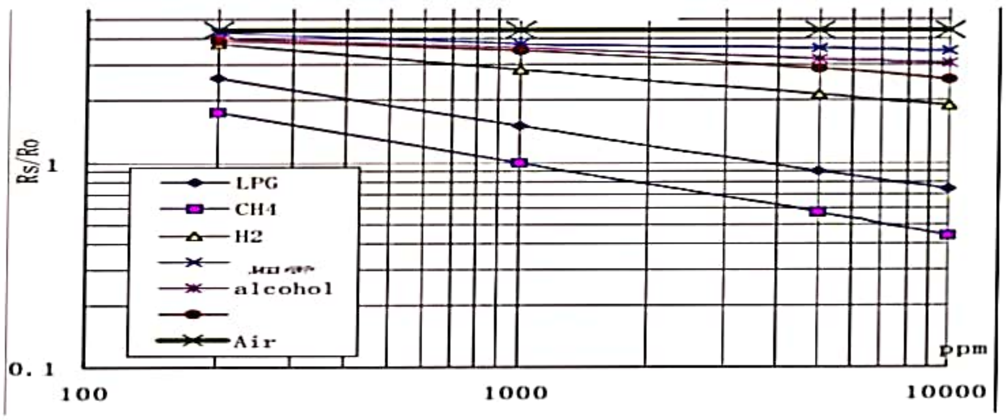


Figure 14 Graph representing ratio vs ppm variations

$$RS = \{(VC * RL)VRL\} - RL \tag{10}$$

Eqn. 9 helps us to find the internal sensor resistance for fresh air [13, 14].

From the graph shown in fig 11, we can see that the resistance ratio in fresh air is a constant:

$$RSR0 = 3.6 \tag{11}$$

Fig. 11 The result of Equation (Eq.) 3.6 is shown in the datasheet. The RS measurement taken in free space is used as a baseline from which R0 is determined. The average analogue reading from the sensor will be converted into a voltage digitally. The RS formula will then be used to determine R0. Assume for the moment that all lines are straight. Because of this linear connection, the formula for both the ratio and the concentration may be combined into a single expression. This allows you to determine the concentration of a gas at any ratio, regardless of whether or not that ratio is shown on the graph. We'll utilise the line equation, but this time using logarithms instead of degrees. Using the data in Figure 11, we attempt to get the following values.

$$log10y = m * log10x + b \tag{12}$$

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:

$$m = \{logy - log(y0)\} / \{logx - log(x0)\} \tag{13}$$

If we apply the logarithmic quotient rule, we get the following: Now we substitute the values for x, x0, y, and $y0$: $m = log(0.75/2.6) \,/\, log(10000/200)$ (14)

$$m = -0.318 \quad (15)$$

We can compute the y-intercept now that we know m. In order to achieve this, we must choose one point from the graph (once again from the CO2 line). As for us, we went with (5000, 0.9)

$$log(y) = m * log(x) + b \quad (16)$$

$$b = log(0.9) - (-0.318) * log(5000) \quad (17)$$

$$b = 1.13 \quad (18)$$

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

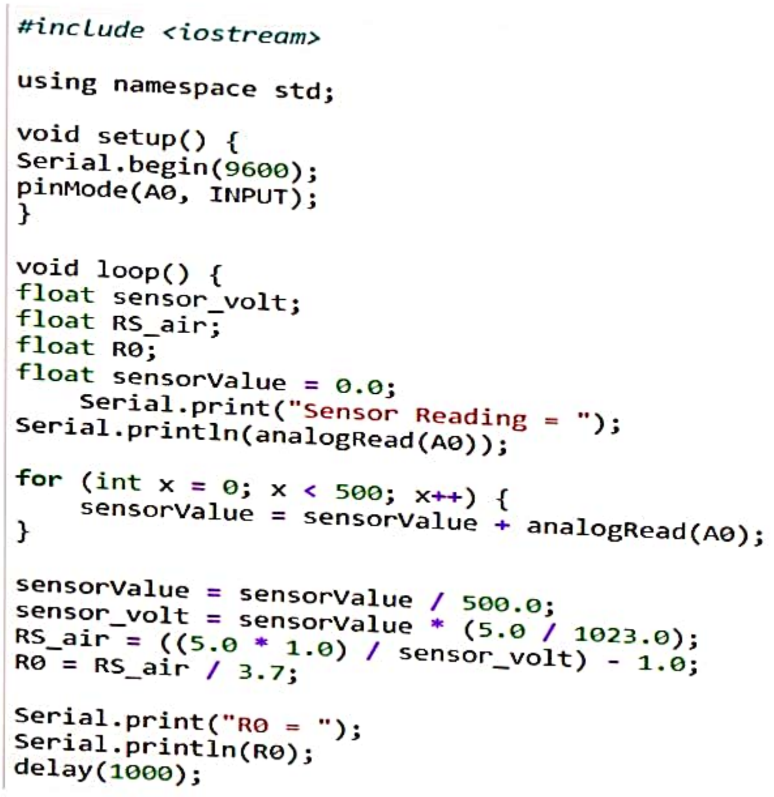
$$log(x) = \{log(y) - b\} \,/\, m \quad (19)$$

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:

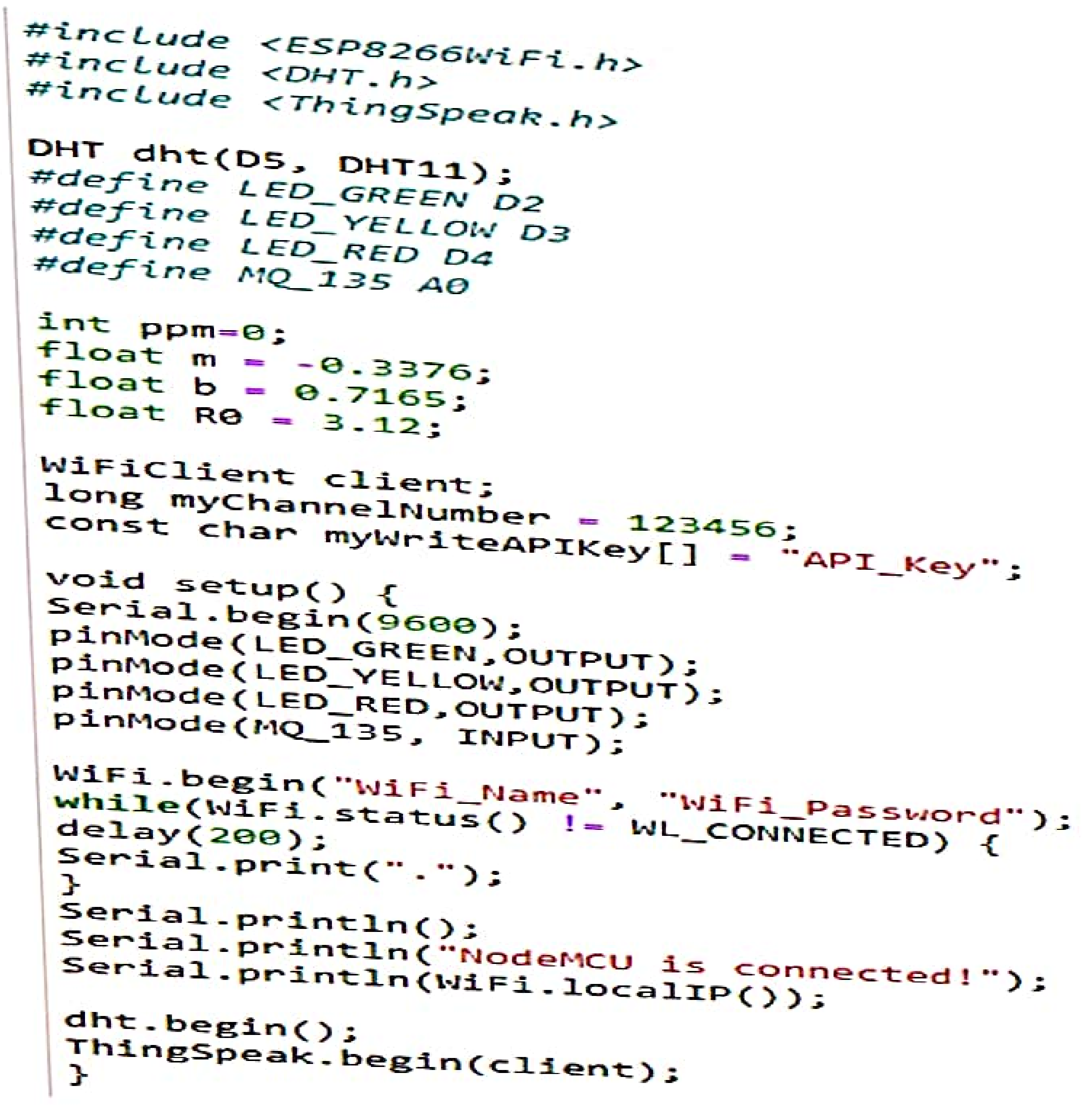
$$x = 10 \wedge [\{log(y) - b] \,/\, m \quad (20)$$

We will be able to convert the sensor output numbers into PPM using equations 9 and 20. (Parts per Million). Now, we created the code and properly connected the Node MCU after flashing it.

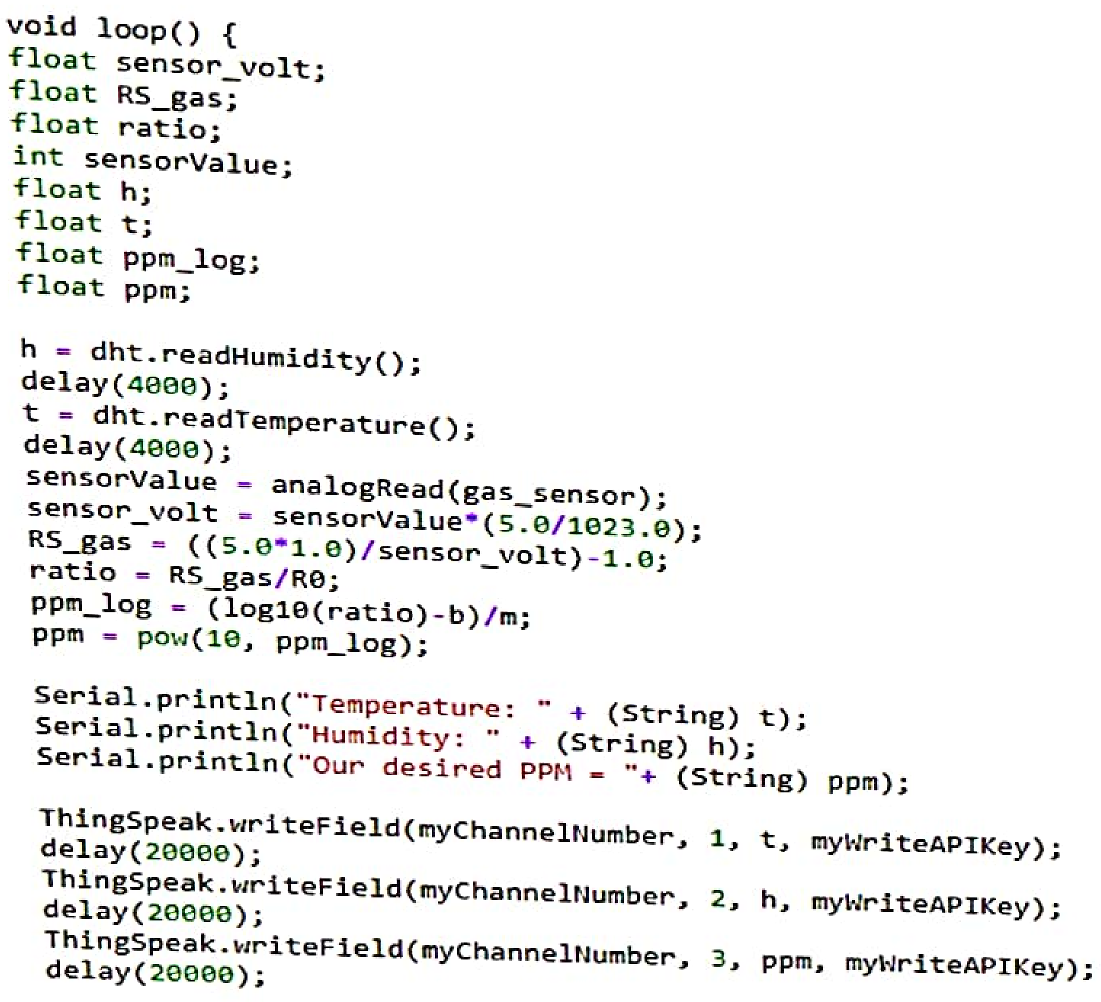
8 SOFTWARE CODE for Calibration of MQ135 Sensor



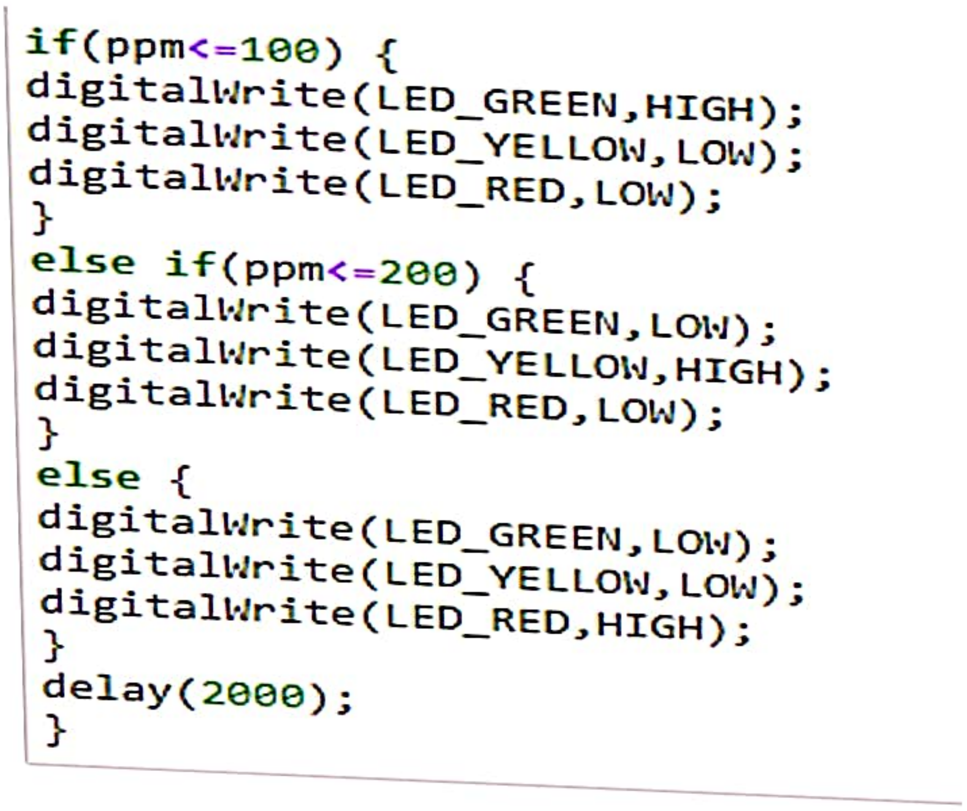
```
#include <iostream>

using namespace std;

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);
```



```
#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;
float b = 0.7165;
float R0 = 3.12;

WiFiClient client;
long myChannelNumber = 123456;
const char myWriteAPIKey[] = "API_Key";

void setup() {
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;
float RS_gas;
float ratio;
int sensorValue;
float h;
float t;
float ppm_log;
float ppm;

h = dht.readHumidity();
delay(4000);
t = dht.readTemperature();
delay(4000);
sensorValue = analogRead(gas_sensor);
sensor_volt = sensorValue*(5.0/1023.0);
RS_gas = ((5.0*1.0)/sensor_volt)-1.0;
ratio = RS_gas/R0;
ppm_log = (log10(ratio)-b)/m;
ppm = pow(10, ppm_log);

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);
}
```

9 Results

The MQ135 sensor measures temperature, humidity, and air quality for the project. When the data was compared to that from a mobile weather app, it was found to be inaccurate by a total of 0.06 degrees, 2.0 percent, and 0.03 ppm for temperature, humidity, and air quality, respectively. The system's accurate readings of ambient temperature and humidity provide evidence of this.

Table 1. Representing MQ135 sensor to detect the temperature, humidity, and air quality

Exp. No.	**Temperature (in celsius)**			**Humidity (in %)**			**Air Quality (in ppm)**		
	Project Reading	**App Reading**	**Error**	**Project Reading**	**App Reading**	**Error**	**Project Reading**	**App Reading**	**Error**
1	31.2	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

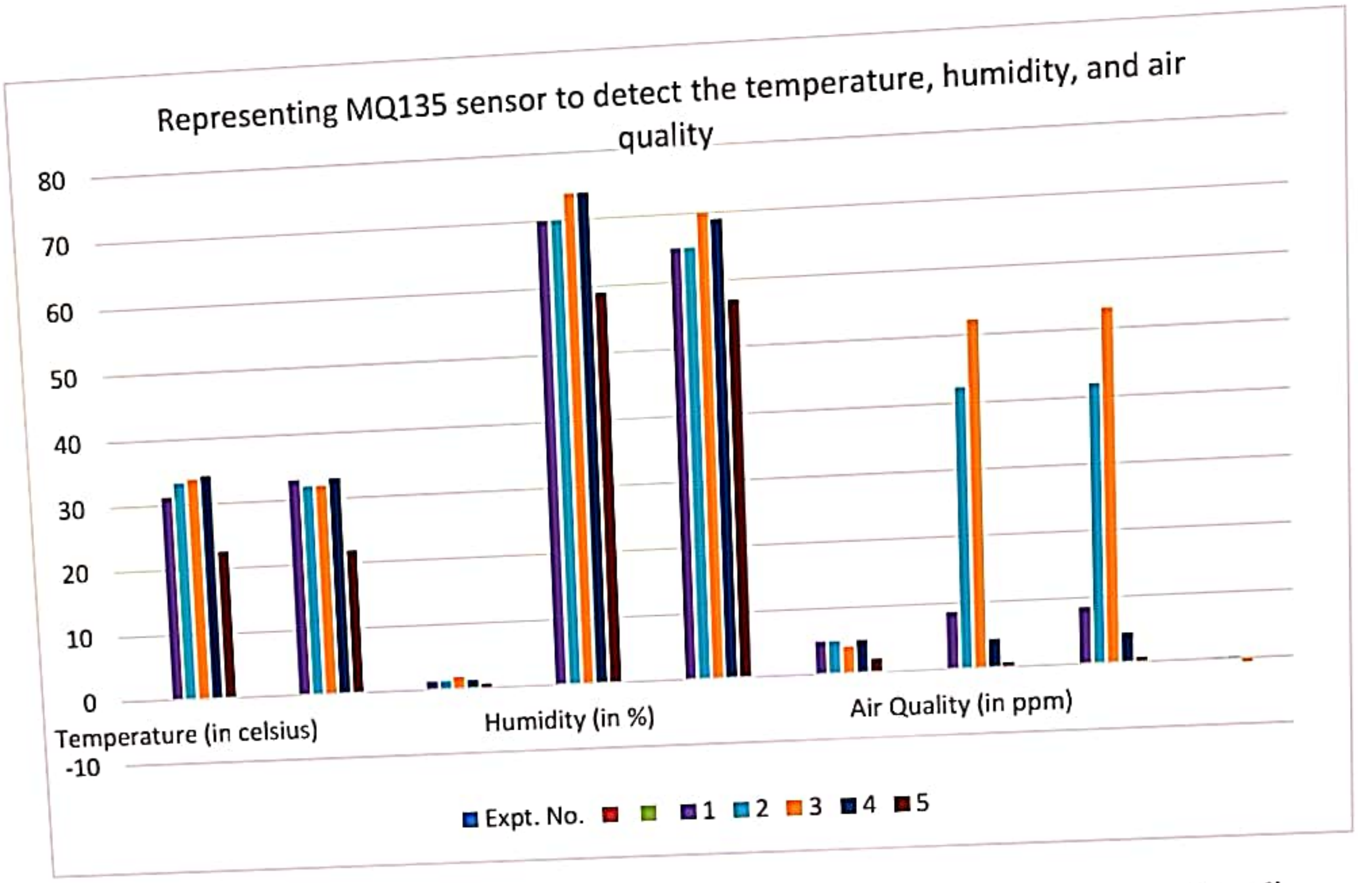


Figure 15 Representing MQ135 sensor to detect the temperature, humidity, and air quality

10 Conclusion

The current temperature, humidity, and Air Quality Index are just some of the metrics that this IoT gadget monitors and displays (AQI). The system uses a MQ135 sensor to detect and measure airborne pollutants such as smoke, CO, CO2, NH4, and others in parts per million (ppm). The instrument provides very accurate readings of both temperature and relative humidity (in Celsius). LED lights on the gadget display the air quality around the setup, with Google data providing further confirmation. By including gas sensors for various pollutants, the system would be able to track the ppm concentrations of each pollutant separately.

The sensor does a great job monitoring the local environment for things like temperature, humidity, and air quality, but it needs a constant internet connection to upload its findings to the Thing Speak cloud. In sum, the system offers a reliable and low-cost means of monitoring environmental factors including temperature, humidity, and air quality.