Air Quality Monitoring.

NAAN MUDHALVAN

7328-SURYA ENGINEERING COLLEGE ,ERODE

TEAM MEMBERS,

V.DEVANATHAN	III-B.E(ECE)
M. GEETHA	III-B.E(ECE)
S.GOKUL	III-B.E(ECE)
T.GOKIL	III-B.E(ECE)
B.GOKULNATH	III-B.E(ECE)

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.

This paper presents real-time standalone air quality monitoring. Internet of Things (IoT)is nowadays finding profound use in each and every sector, plays a key role in our air quality monitoring system too. The setup will show the air quality in PPM on the webpage so that we can monitor it very easily.

In this IoT project, we can monitor the pollution level from anywhere using your computer or mobile.

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

Working Procedures

Node MCU 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 Think Speak cloud over the internet. We have also provided LED indicators to indicate the safety levels.

STEP 1. Firstly, the calibration of the 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 Node MCU.

STEP 5. Finally, the complete hardware circuit is implemented.
The software codes and the hardware circuits are described in the following chapters.

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:

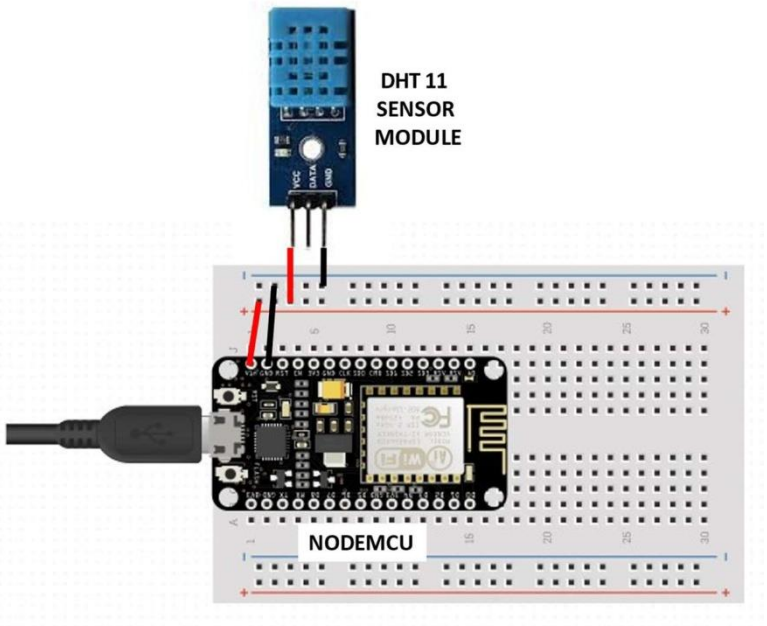
STEP 1 : The Vcc pin of the DHT11 sensor module was connected with the VU pin of Node MCU.

STEP 2 : The Gnd pin of the DHT11 sensor module was connected with the Gnd pin of Node MCU.

STEP 3 : The Node MCU is powered with a 12V DC via AC-DC adapter for 20 minutes.

STEP 4 : The setup was then disconnected.

Figure shown below describes the foresaid connections.



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

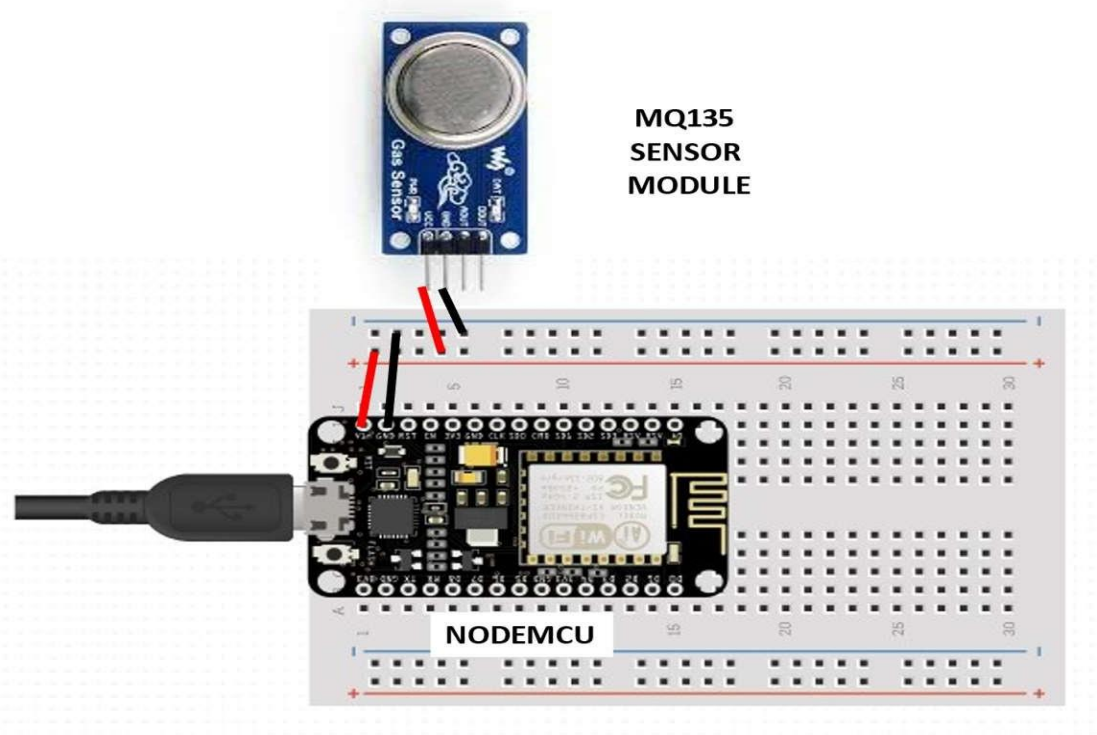
STEP 1 : The Vcc pin of the MQ-135 gas sensor module was connected with the VU pin of Node MCU

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

STEP 3 : The Node MCU is powered with a 12V DC via AC-DC adapter for a day.

STEP 4 : The setup was then disconnected

Figure shown below describes the foresaid connections.



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 VUpin of NodeMCU.

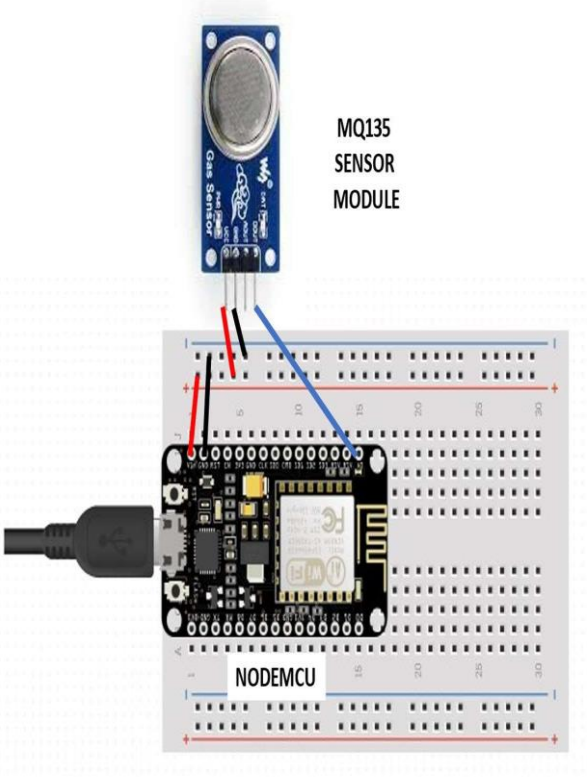
STEP 2 : The Gnd pin of the MQ-135 gas sensor module was connected with the Gndpin of Node MCU.

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

STEP 4 : The software code to calibrate the sensor is then uploaded to the Node MCUand the value of R0 in fresh air is collected from the serial monitor of the Arduino IDE.

STEP 5 : The setup was then disconnected.

Figure shown below describes the foresaid



connections.

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 Node MCU.

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

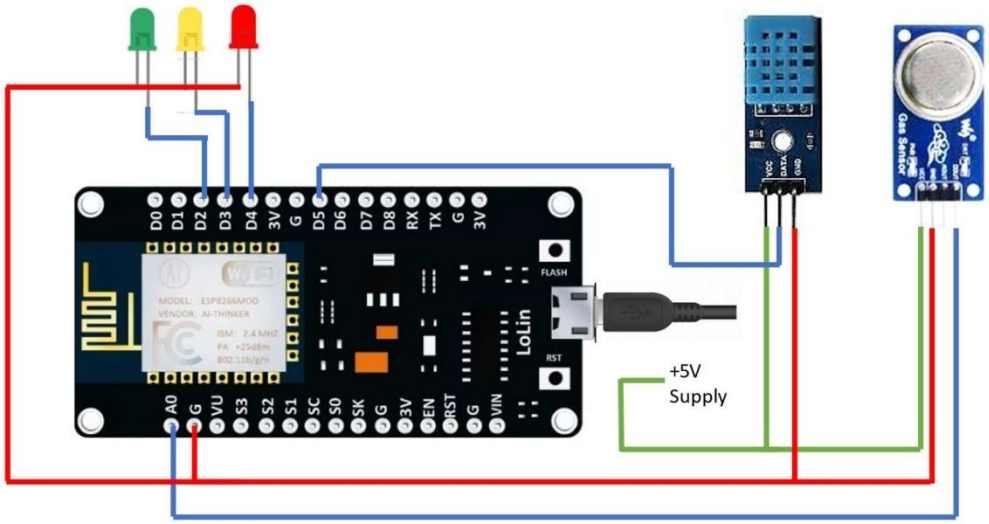
STEP 4 : The DATA pin of the DHT11 sensor module was connected with the D0 pin of the Node MCU.

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 Node MCU.

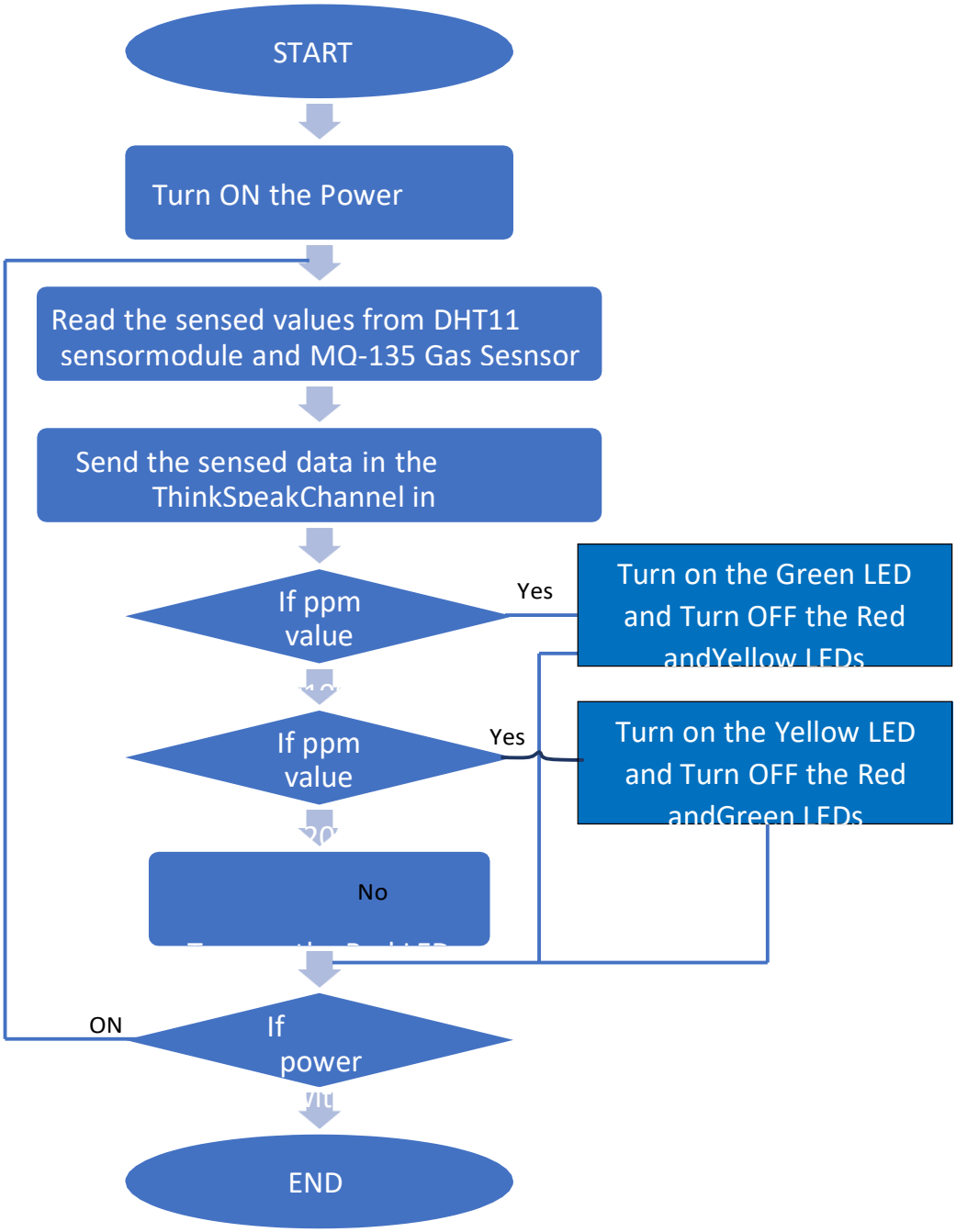
STEP 7 : The setup was then powered with 9V DC via AC-DC adapter.

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



SOFTWARE IMPLEMENTATION

Working Algorithm



♦ *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
readingsSerial.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

```
5   #include<ESP8266Wifi.h>
    #include <DHT.h>
6   #include <ThingSpeak.h>
7
8   DHT dht(D5, DHT11);
9   #define LED_GREEN D2
    #define LED_YELLOW
    D3#define LED_RED D4
    #define MQ_135 A0
10  int ppm=0;
11  float m = -0.3376; //Slope
    float b = 0.7165; //Y-
    Intercept
```

```
12  float R0 = 3.12; //Sensor Resistance in fresh air from previous code

    WiFiClient client;

13  long myChannelNumber = 123456; // Channel id
14  const char myWriteAPIKey[] = "API_Key";
15
16  void setup() {
17  // put your setup code here, to run once:
18  Serial.begin(9600);
    pinMode(LED_GREEN,OUTP
    UT);
    pinMode(LED_YELLOW,OU
    TPUT);
    pinMode(LED_RED,OUTPUT
    );
19  pinMode(MQ_135, INPUT);
    WiFi.begin("WiFi_Name", "WiFi_Password");
    while(WiFi.status() != WL_CONNECTED)
20  {
21  delay(200);
    Serial.print("
    .");
22  }
23  Serial.println();
24  Serial.println("NodeMCU is connected!");
    Serial.println(WiFi.localIP());
25  dht.begin();
    ThingSpeak.begin(clie
    nt);
26  }
27
28  void loop() {
29  float sensor_volt; //Define variable for sensor
    voltagefloat RS_gas; //Define variable for
    sensor resistance float ratio; //Define variable
    for ratio
30  int sensorValue;//Variable to store the analog values from MQ-
    135float h;
31  float t;
32  float ppm_log; //Get ppm value in linear scale according to the the ratio
    valuefloat ppm; //Convert ppm value to log scale
33  h =
    dht.readHumidity();
    delay(4000);
34  t = dht.readTemperature();
    delay(4000);
```

```
35
36
37  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
38  ratio = RS_gas/R0; // Get ratio RS_gas/RS_air
39  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
40
41  Serial.println("Temperature: " +
    (String) t);Serial.println("Humidity:
    " + (String) h);
42  Serial.println("Our desired PPM = "+ (String) ppm);
43
44  ThingSpeak.writeField(myChannelNumber, 1, t,
    myWriteAPIKey);delay(20000);
45  ThingSpeak.writeField(myChannelNumber, 2, h,
    myWriteAPIKey);delay(20000);
46  ThingSpeak.writeField(myChannelNumber, 3, ppm,
    myWriteAPIKey);delay(20000);
47
48  if(ppm<=100)
49  {
50  digitalWrite(LED_GREEN,HI
    GH);
    digitalWrite(LED_YELLOW,L
    OW);
    digitalWrite(LED_RED,LOW);
51  }
52  else if(ppm<=200)
53  {
54  digitalWrite(LED_GREEN,LO
    W);
    digitalWrite(LED_YELLOW,
    HIGH);
    digitalWrite(LED_RED,LOW);
55  }
56  else
57  {
58  digitalWrite(LED_GREEN,LO
    W);
    digitalWrite(LED_YELLOW,L
    OW);
    digitalWrite(LED_RED,HIGH)
    ;
```

```
59    }
60    delay(2000);}
```

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, CO_2, 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.