TOPIC:

SENSOR STATION:

MEASURING AIR QUALITY INDEX

Made By:
Ishani Sharma(2101AI16) & Pragya Harsh(2101AI23)

Submission Date: 02/12/2022

Motivation:

About Air Pollution:

Air pollution has become a common phenomenon everywhere. Specially in the urban areas, air pollution is a real-life problem. A lot of people get sick only due to air pollution. In the urban areas, the increased number of petrol and diesel vehicles and the presence of industrial areas at the outskirts of the major cities are the main causes of air pollution. The problem is seriously intensified in the metropolitan cities. Also, the climate change is now apparent. The governments all around the world are taking every measure in their capacity. Many European countries have aimed to replace petrol and diesel vehicles with the electric vehicles by 2030. Even India has aimed to do so by 2025. The use of coal for electricity generation is now going to be a thing of past. The nations are now focusing to generate energy from nuclear reactors and the renewable resources like solar energy, wind energy and hydroelectric power.

About Project:

It is now important to monitor air pollution in real time in most of the urban areas.
This project is aimed at developing an IOT device which can monitor air pollution in real time and log data to a remote server. Remote monitoring was facilitated using classical modes in the past, which has some pitfalls like limited memory, processing speed and complex programming strategies. By using Internet of Things and recording sensor data to a remote server, the limitations of memory in the monitoring devices and manual collection of data from the installed devices can be overcome. The IOT also helps monitoring the data in real time.

Introduction:

Now, in our project, we have tried to build a model that could help in minimizing one of the most challenging issues of air pollution.
The air pollution monitoring device developed in this project is based on Arduino UNO, which connects to the ThingSpeak App using the Wi-Fi Module. The sensor used for monitoring air pollution is the MQ-135 gas sensor. In this project, we have also integrated a DHT11 Temperature and Humidity Sensor. The sensor data is also displayed on the serial monitor interfaced with the monitoring IoT device.

Description:

In our project, the Arduino Board connects to the ThingSpeak App using the ESP8266 Wifi Module. As the cities usually have Wi-Fi hotspots at most of the places, so the device can be easily installed near any hotspot for its operation. The ThingSpeak is a popular IOT platform.
The MQ-135 gas sensor monitors the level of gases like Carbon dioxide, Carbon monoxide, Ammonia, Acetone, Toulene, Sulphide which major contributers to air pollution.
The DHT11 Temperature and humidity sensor measures the temperature and humidity of the environment.
Monitoring temperature along with the major polluting gases could also help in investigating about global warming, another major environmental problem of the era.
Thus, the sending the sensed data to the ThingSpeak server using the Wi-Fi module is managed by the Arduino Sketch.
The ThingSpeak App helps in analysing the sensed data in various different ways, like plotting the data set in the form of a graph that makes the information more illustrative.
The Arduino sketch is written, compiled, and loaded to the Arduino board using Arduino IDE.

COMPONENTS REQUIRED:

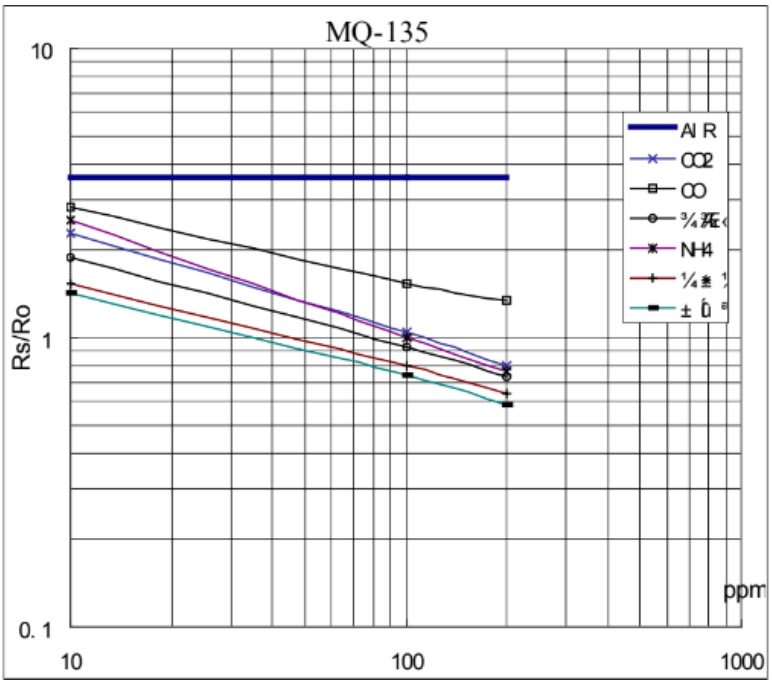
1. ***ARDUINO UNO:***
   Arduino UNO is one of the most popular prototyping boards. It is small in size and packed with rich features. The board comes with built-in Arduino boot loader. It is an Atmega 328 based controller board which has 14 GPIO pins, 6 PWM pins, 6 Analog inputs and on board UART, SPI and TWI interfaces. In this IOT device, 9 pins of the board are utilized. There are six pins used to interface the character LCD. There are two pins utilized to interface the ESP8266 Wi-Fi Module and an analog input pin is used to connect with the MQ-135 sensor

1. ***NodMCU ESP8266 Wi-Fi Module:***

The ESP8266 Wi-Fi Module is used to connect with any available internet hotspot and transfer sensor data to ThingSpeak Platform via Wi-Fi. The ESP8266 Wi-Fi Module is a self contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to a Wi-Fi network. The ESP8266 is capable of either hosting an application or off loading all Wi-Fi networking functions from another application processor. Each ESP8266 module comes pre-programmed with an AT command set firmware. So, one can simply hook this up to an Arduino device. Here it uploads the monitoring data to the cloud. The module comes available in two models – ESP-01 and ESP-12. ESP-12 has 16 pins available for interfacing while ESP-01 has only 8 pins available for use.

2. ***MQ-135 Sensor:***

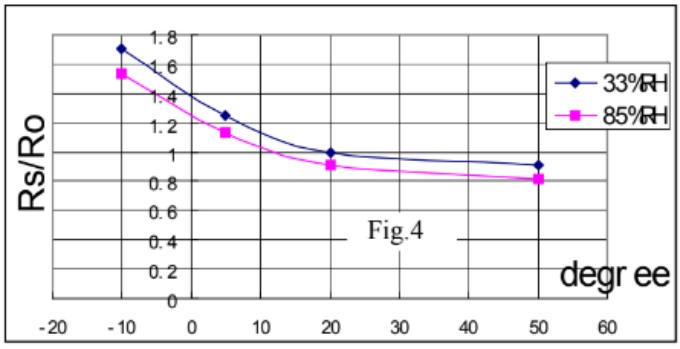
MQ-135 is a gas sensor which is used to measure the concentration of combustible gases. It has lower conductivity in clean air while its conductivity increases with the presence of the combustible gases in the air. The sensor is highly sensitive to gases like Ammonia, Sulphide and Benzene steam. The sensor can detect the concentration of combustible gases in range from 100 PPM to 1000 PPM.



Sensitivity characteristics of the MQ-135 for several gases.

Temp: 20 degrees celcius
Humidity: 65%
O2 concentration 21%
RL=20kΩ
Ro: sensor resistance at 100ppm of
NH3 in the clean air.
Rs: sensor resistance at various concentrations of gases.

From the sensitivity curve of the sensor, it can be seen that the resistance of the sensor decreases as the concentration of the target gas is increased in PPM while for clean air its resistance remains constant. In the graph, the Rs is the resistance in target gas and Ro is the resistance in clean air. The graph is shown for Carbon dioxide, Carbon Monoxide and Ammonia. The sensitivity of this sensor can be adjusted and calibrated to detect specific concentration level of a target gas. The sensor has four terminals – Ground, VCC, Digital Out and Analog Out. The VCC and Ground terminals of the sensor are connected to the common VCC and Ground. The Analog Output pin of the sensor is connected to the A0 pin of the Arduino. The analog output voltage from the sensor can be assumed directly proportional to the concentration of CO2 gas in PPM under standard conditions. The analog voltage is sensed from the sensor and converted to a digital value in range from 0 to 1023 by the inbuilt ADC channel of the controller. The digitized value is hence equal to the gas concentration in PPM.



Dependence of the MQ-135 on temperature and humidity.

Ro: sensor resistance at 100ppm of NH3 in air

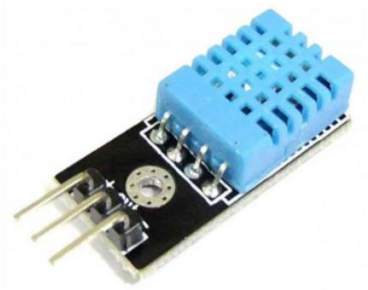
at 33%RH and 20 degrees .

Rs: sensor resistance at 100ppm of NH3

at different temperatures and humidity.

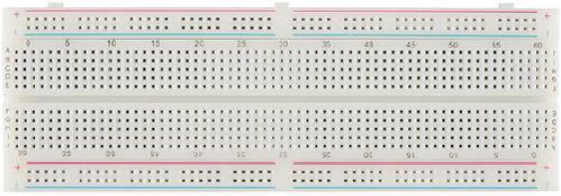
3. *DHT11 Sensor:*

The DHT-11 Digital Temperature and Humidity Sensor is a basic, low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air and spits out a digital signal on the data pin (no analog input pins needed).



4. *Breadboard*:

A breadboard is a solder less rectangular plastic board with a bunch of tiny holes in it. These holes let us easily insert electronic components to prototype an electronic circuit.



5. *Jumper Wires*

APPLICATIONS USED:

- ***ThingSpeak :***

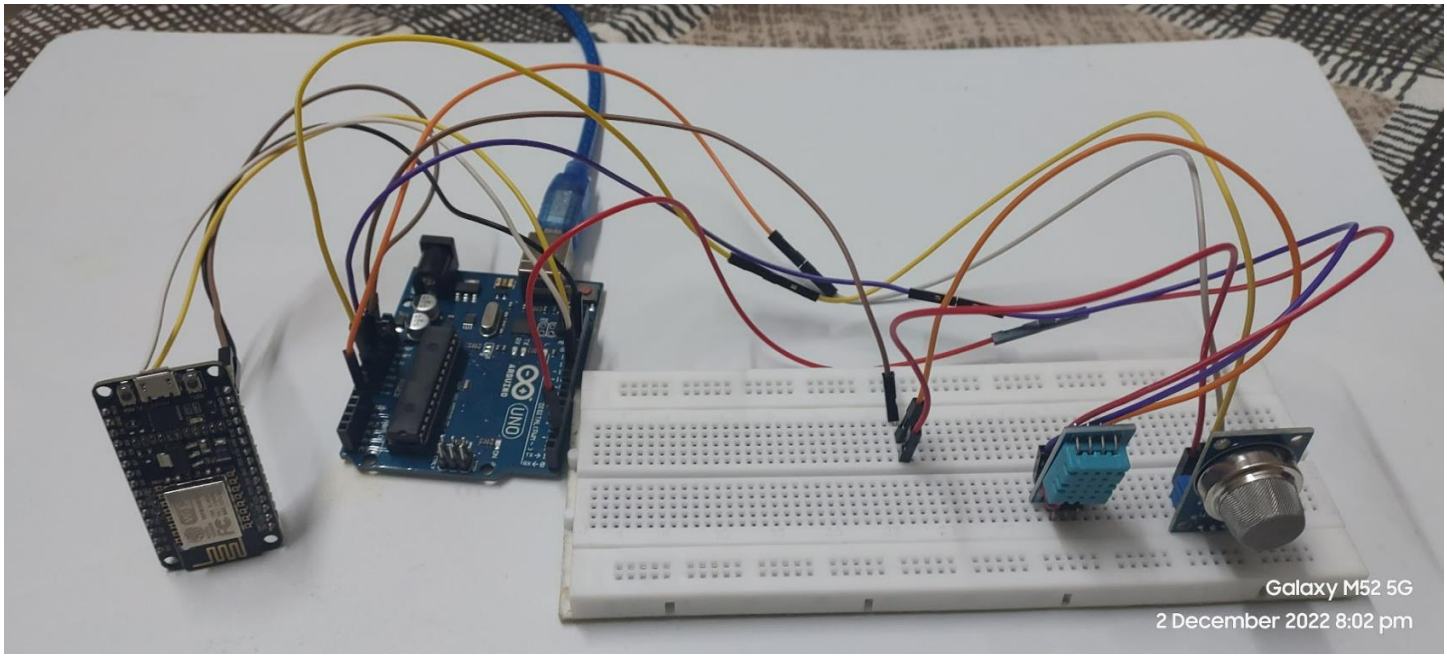
ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize, and analyze live data streams in the cloud. You can send data to ThingSpeak from your devices, create instant visualization of live data, and send alerts.

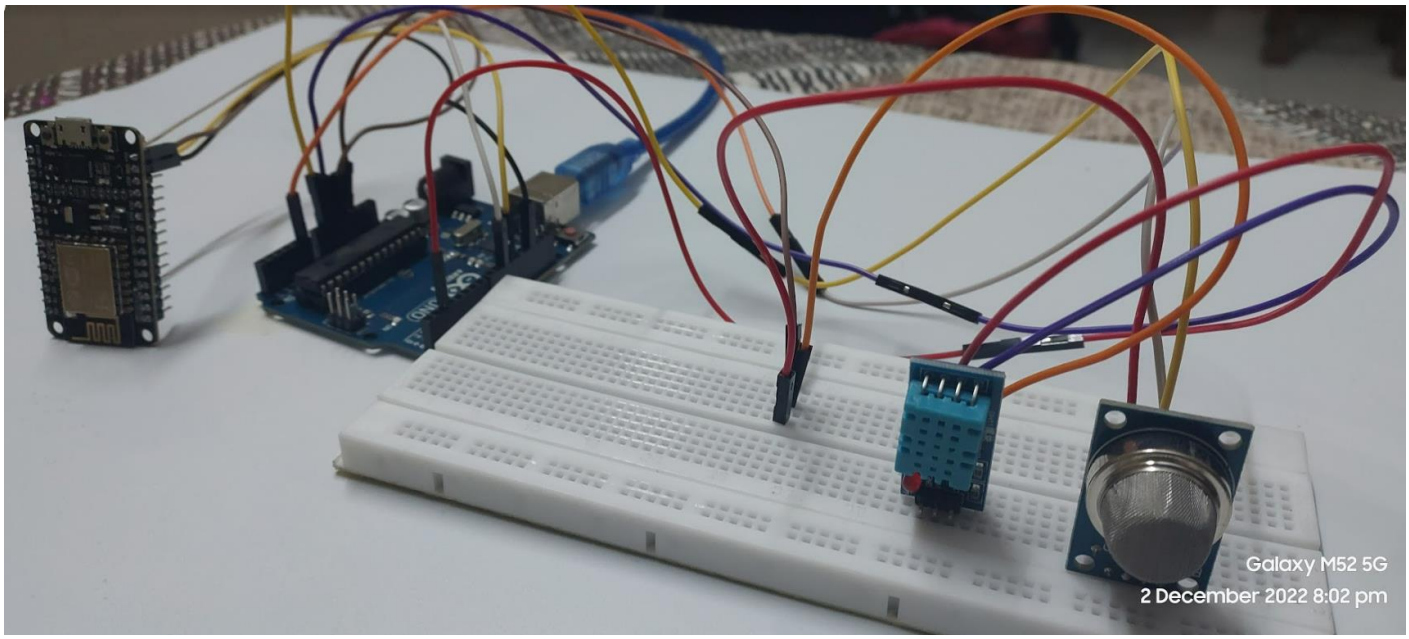
Our channel Link: https://thingspeak.com/channels/1955506

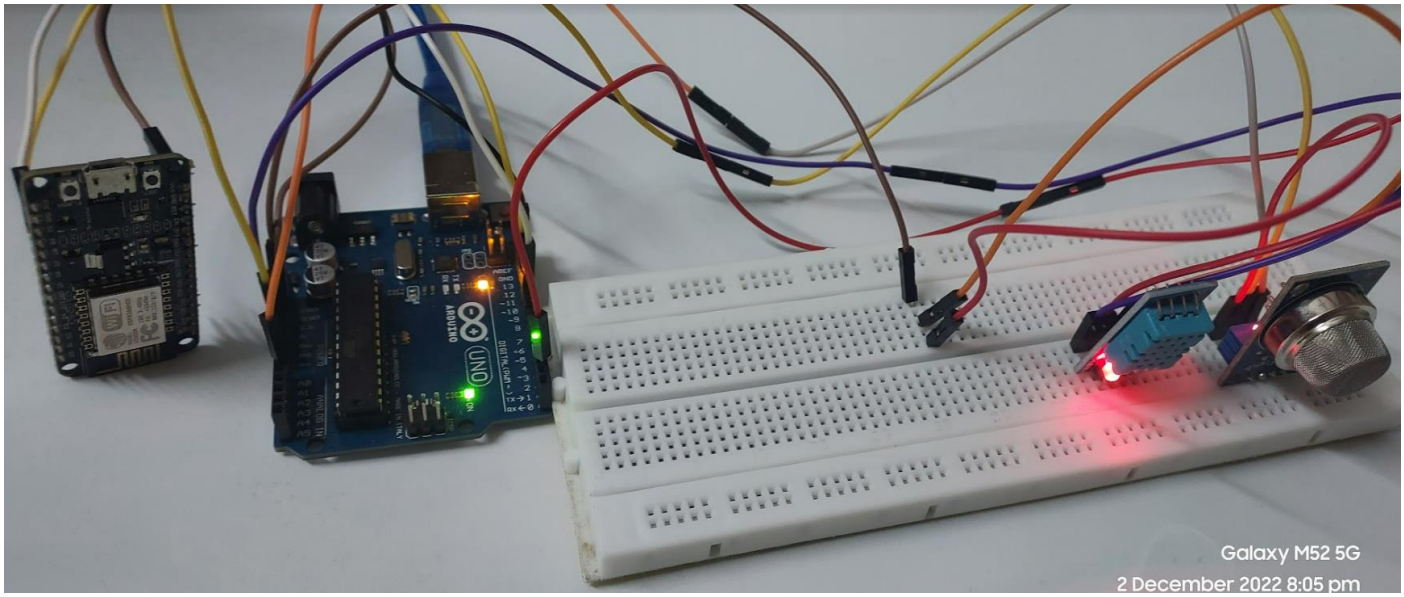
- ***Arduino IDE :***

  The Arduino Integrated Development Environment – or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

FULL CIRCUIT IMAGES:







CONNECTION DESCRIPTION:

1) **NodeMCU Esp8266**

   TX - 10

   RX - 11

   Vin - 5v

   Gnd - gnd

2) **DHT11**

   Vcc - 3.3v

   Data - 2

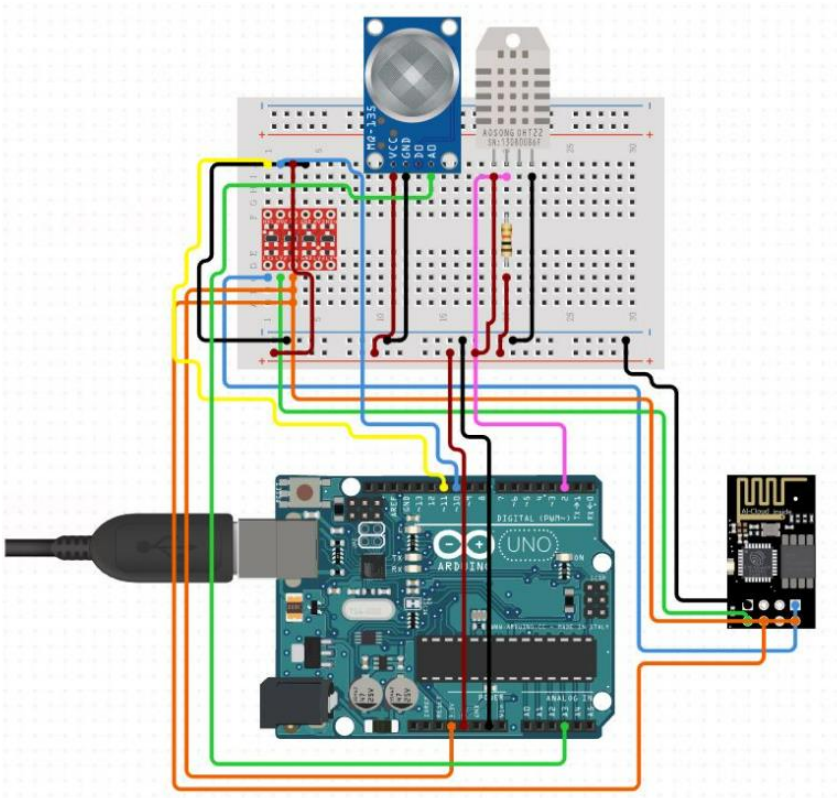
   Gnd - gnd

3) **MQ135**

   A0 - A0

   Vcc - 3.3v

   Gnd – gnd

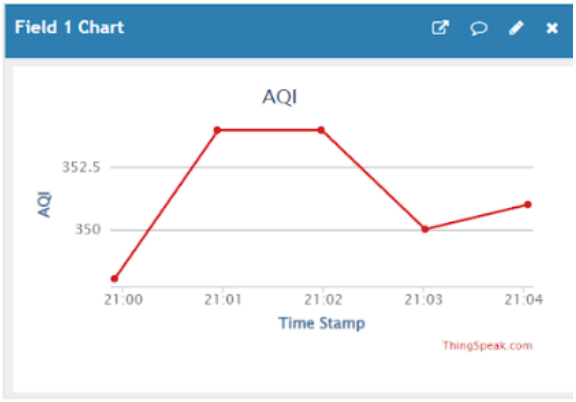


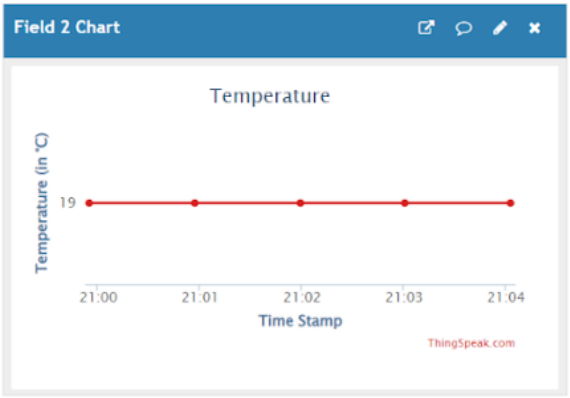
Working Flow:

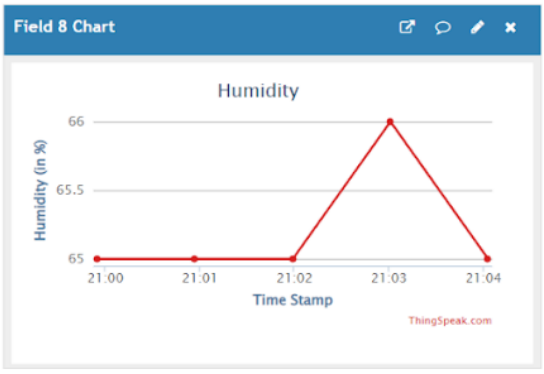
The device developed in this project can be installed near any Wi-Fi hotspot in a populated urban area. As the device is powered, the Arduino board loads the required libraries, flashes some initial messages on the serial monitor and starts sensing data from the MQ-135 sensor and DHT11 sensor. The sensitivity curve of the gas sensor for different combustible gases is already mentioned above. The sensor can be calibrated so that its analog output voltage is

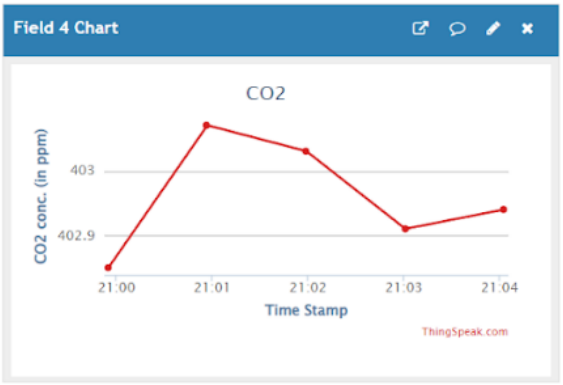
proportional to the concentration of polluting gases in PPM. The analog voltage sensed at the pin of the Arduino is converted to a digital value by using the in-built ADC channel of the Arduino. The Arduino board has 10-bit ADC channels, so the digitized value ranges from 0 to 1023. The digitized value can be assumed proportional to the concentration of gases in PPM. The read value of the Total AQI is first displayed on serial monitor and passed to the ESP8266 module wrapped in proper string through virtual serial function. The Wi-Fi module is configured to connect with the ThingSpeak IOT platform. ThingSpeak is an IOT analytics platform service that allows to aggregate, visualize and analyze live data streams in the cloud. ThingSpeak provides instant visualizations of data posted by the IOT devices to ThingSpeak server.

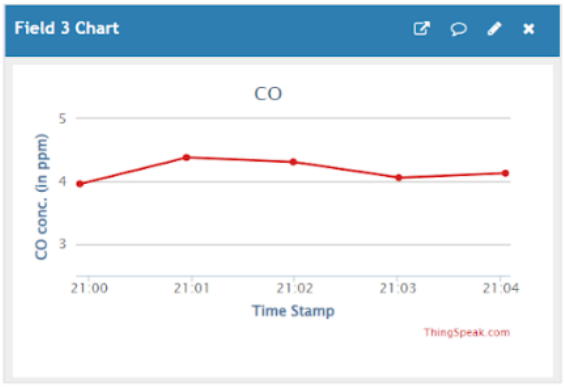
Plots of the Data sensed by the MQ135 and DHT11 sensor as analysed by the ThingSpeak server

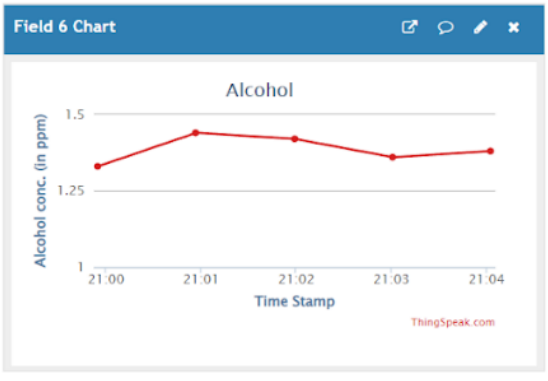


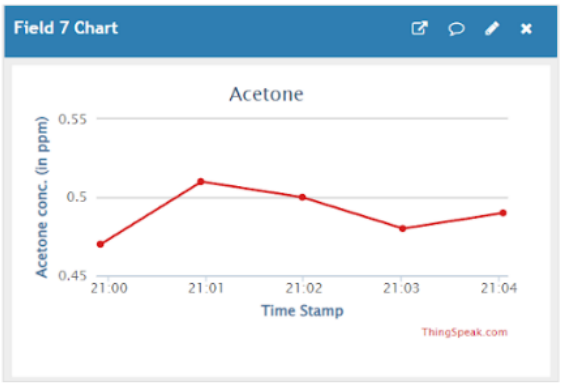


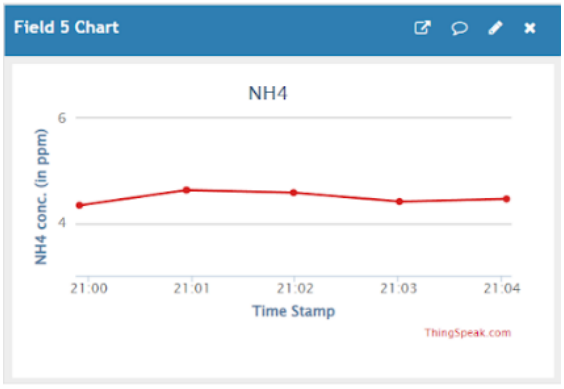




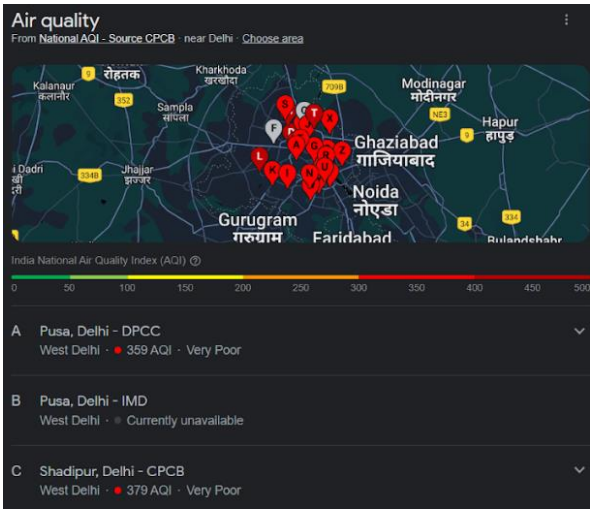








(Data used for the plots is of South-West Delhi as of 1st December 2022 at 9 pm)



CODE:

```
#include <SoftwareSerial.h> // library for wifi module
#include <MQUnifiedsensor.h> // library for MQ135 sensor
#include <dht.h> // library for DHT11 sensor

SoftwareSerial esp(10, 11); // TX=10, RX=11
unsigned long starttime;

String myAPIkey = "2RBIX9P3YKF4M5UI"; // API Key to write the data on the
channel

//these will be used to measure different values from the sensors
float aqi = 0;
float CO = 0;
float Alcohol = 0;
float CO2 = 0;
float NH4 = 0;
float Acetone = 0;
float temp = 0;
float humidity = 0;

//definitions for MQ135
MQUnifiedsensor MQ135("Arduino UNO", 5, 10, A0, "MQ-135"); // MQ135(arduino,
Voltage_Resolution, ADC_Bit_Resolution, pin, type);

//definition for DHT sensor
dht DHT;

// setup would run once
void setup()
{
  //start serial communications
  Serial.begin(9600); //default port for serial monitor
  esp.begin(115200); //default port for ESP8266

  //message for initializing
  Serial.println("Initializing...");

  //enable software serial
  esp.println("AT");
  delay(2000);

  //if the wifi odule is found
  if(esp.find("OK")){
      Serial.println("Communication with ESP8266 module: OK");
  }
  //in case we don't find it
  else {
```

```
    Serial.println("ESP8266 module ERROR");
  }

  //trying to connect to wifi
  Serial.println("Connecting wi-fi...");
  String cmd ="AT+CWMODE=1";
  esp.println(cmd);
  delay(2000);
  esp.flush();
  esp.println("AT+CWJAP=\"nopee\",\"welp22welp\"\r\n"); //contains SSID=nopee,
password=welp22welp
  delay(5000);

  //if connection succeeds
  if(esp.find("OK")){
    Serial.println("Connection succeeded!");
  }
  //in case it fails
  else{
    Serial.println("Connection failed!");
  }
  Serial.println();

  //Setting up MQ135

  //Set math model to calculate the PPM concentration and the value of
constants
  MQ135.setRegressionMethod(1); //PPM =  a*ratio^b

  MQ135.init(); //configure the pin of arduino

  // In this routine the sensor will measure the resistance of the sensor
supposedly before being pre-heated
  // and on clean air (Calibration conditions), setting up R0 value.
  Serial.print("Calibrating please wait.");
  float calcR0 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ135.update(); // Update data, the arduino will be read the voltage on
the analog pin
    calcR0 += MQ135.calibrate(3.6);
    Serial.print(".");
  }
  MQ135.setR0(calcR0/10);
  Serial.println("  done!.");

  //keeping errors in check
```

```
  if(isinf(calcR0)) {Serial.println("Warning: Conection issue founded, R0 is
infite (Open circuit detected) please check your wiring and supply");
while(1);}
  if(calcR0 == 0){Serial.println("Warning: Conection issue founded, R0 is zero
(Analog pin with short circuit to ground) please check your wiring and
supply"); while(1);}
  //MQ Setup and calibration finished

  //initializing sampling
  Serial.print("Sampling...");
}



// the loop

void loop()

{
  //reading the values of sensors
  aqi = analogRead(A0);
  DHT.read11(2);

  //printing the AQI on monitor
  Serial.print("Airquality = ");
  Serial.println(aqi);

  /*
    Exponential regression:
  GAS      | a      | b
  CO       | 605.18 | -3.937
  Alcohol  | 77.255 | -3.18
  CO2      | 110.47 | -2.862
  Toluen   | 44.947 | -3.445
  NH4      | 102.2  | -2.473
  Aceton   | 34.668 | -3.369
  */

  // MQ Reading and updating section
  MQ135.update(); // Update data, the arduino will be read the voltage on the
analog pin
  //setting to measure CO conc. in PPM
  MQ135.setA(605.18); MQ135.setB(-3.937);
  CO = MQ135.readSensor();

  //setting to measure Alcohol conc. in PPM
  MQ135.setA(77.255); MQ135.setB(-3.18);
  Alcohol = MQ135.readSensor();
```

```
  //setting to measure CO2 conc. in PPM
  MQ135.setA(110.47); MQ135.setB(-2.862);
  CO2 = MQ135.readSensor();

  //setting to measure NH4 conc. in PPM
  MQ135.setA(102.2 ); MQ135.setB(-2.473);
  NH4 = MQ135.readSensor();

  //setting to measure Acetone conc. in PPM
  MQ135.setA(34.668); MQ135.setB(-3.369);
  Acetone = MQ135.readSensor();

  //reading temperature
  temp = DHT.temperature;

  //reading humidity
  humidity = DHT.humidity;

  //calling funtion to write to Thingspeak using ESP8266 module
  writeThingSpeak();

  //creating 60 seconds delay
  Serial.print("Wait 60 seconds");
  delay(60000);


  //initializing new cycle
  Serial.println();
  Serial.print("Sampling...");
  starttime = millis();
}

//function to write to thingspeak
void writeThingSpeak(void)
{
  //calling function to connect to thingspeak
  startThingSpeakCmd();

  // preparing the string GET
  String getStr = "GET /update?api_key=";

  getStr += myAPIkey;
  getStr +="&field1=";
  getStr += String(aqi);
  getStr +="&field2=";
  getStr += String(temp);
  getStr +="&field3=";
  getStr += String(CO);
  getStr +="&field4=";
```

```
  // We have added 400 PPM because when the library is calibrated it assumes the current state of the
  // air as 0 PPM, and it is considered today that the CO2 present in the atmosphere is around 400 PPM.
  getStr += String(CO2+400);
  getStr +="&field5=";
  getStr += String(NH4);
  getStr +="&field6=";
  getStr += String(Alcohol);
  getStr +="&field7=";
  getStr += String(Acetone);
  getStr +="&field8=";
  getStr += String(humidity);
  getStr += "\r\n\r\n";

  //calling function to update the value
  GetThingspeakcmd(getStr);

}

//function to connect to thingspeak
void startThingSpeakCmd(void)
{
  esp.flush();
  String cmd = "AT+CIPSTART=\"TCP\",\"";
  cmd += "184.106.153.149"; // api.thingspeak.com IP address
  cmd += "\",80";
  esp.println(cmd);

  if(esp.find("Error"))
  {
    return;
  }
}

//function to update the value
String GetThingspeakcmd(String getStr)
{
  String cmd = "AT+CIPSEND=";
  cmd += String(getStr.length());
  esp.println(cmd);

  if(esp.find(">"))
  {
    esp.print(getStr);
    Serial.println(getStr);
    delay(500);
    String messageBody = "";
```

```
    while (esp.available())
    {
      String line = esp.readStringUntil('\n');
      if (line.length() == 1)
      {
        messageBody = esp.readStringUntil('\n');
      }
    }
    return messageBody;
  }
  else
  {
    esp.println("AT+CIPCLOSE");
    Serial.println("AT+CIPCLOSE");
  }
}
```

Individual Contributions:

We worked collectively thus each of us have complete knowledge about working of each component and hence the whole project but here we have tried to mention the fields where the contribution by one was a little more than the other.

Ishani Sharma

- Decided the Project Idea
- Collected the components
- Looked for the documentation of DHT11 and WiFi-Module
- Prepared the Circuit Diagram
- Verified the Code and rectified the errors in that
- Connected DHT11 and WiFi module with Arduino
- Made sure that the WiFi module was working properly
- Made sure that the data was displayed in the mobile app correctly
- Helped in making respective parts of documentation and PPT
- Conducted initial data collection and compilation before live demo
- Recorded the Video Demo

Pragya Harsh

- Decided the Project Idea
- Looked for the documentation of MQ135
- Verified the Circuit Diagram
- Wrote the basic structure of code
- Connected MQ135 with Arduino
- Made sure that the MQ135 was taking readings correctly
- Made sure that DHT11 was taking readings correctly
- Made sure that the data was sent to ThingSpeak website and plotted correctly
- Helped in making respective documentation and PPT
- Final data collection and compilation
- Presented the Video Demo