IOT BASED AIR POLLUTION MONITORING SYSTEM

V. SHRI SARVESH (108118109)
ADITHYA SINEESH (108118005)
BALAJI K S (108118021)

INTRODUCTION

- Air Pollution is the presence of substances in the atmosphere that are harmful to the health of humans and other animals. There are different types of air pollutants such as gases (Ozone, Nitrogen Dioxide, Carbon Monoxide, etc.) and particulates (PM10 and PM2.5).
- Air pollution can cause health problems, like heart attacks, strokes, diabetes and high blood pressure, that have been identified as the pre-existing medical conditions that raise the chances of death from COVID-19 infection.
- Emerging research, including a study from Harvard T.H. Chan School of Public Health, finds that breathing more polluted air over many years may itself worsen the effects of COVID-19.
- Currently, the monitoring of air quality levels is achieved by placing sensors in various locations and the sensors alert if the pollution levels exceed the threshold level.
- However, there hardly exists a mobile solution wherein an individual can contribute to the pollution level checking from the convenience of their homes.
- Moreover, solutions for forecasting gas and particulate matter levels are location specific and require expensive training data (Meteorological data, wind speeds, etc.) which is unavailable in the reach of the common people.

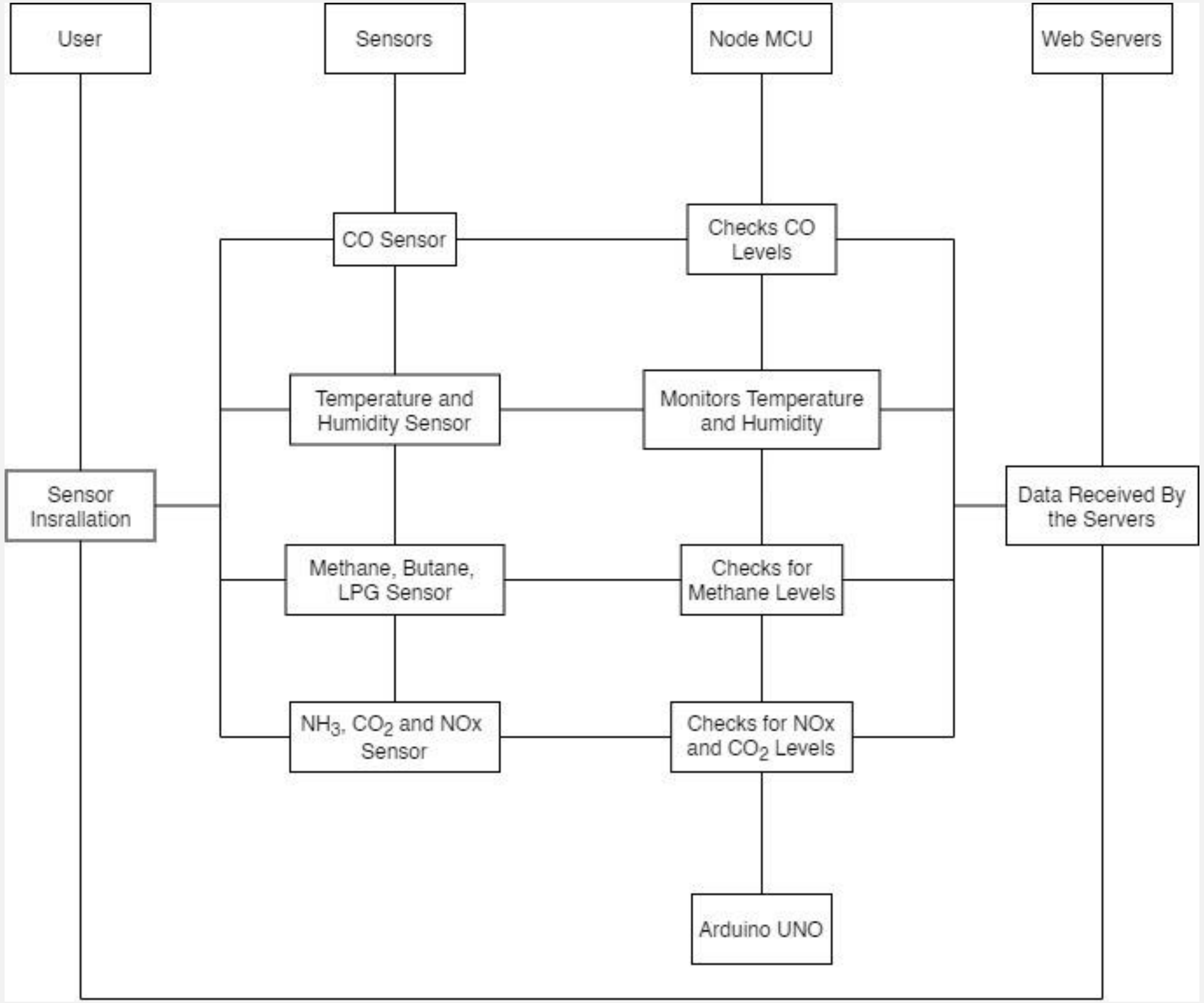
PROPOSED IDEA

- We aim to collect data about the concentration of NO_2, Ozone, PM2.5 and PM10 in different locations and host its visualization on a website.
- Furthermore, we also aim to predict their future concentrations using Machine Learning approaches.
- The future concentration levels that are being predicted is aimed to provide maximum accuracy based on the regions in which the system is placed.
- The proposed device can be placed anywhere in the user's home and the user can obtain real time data of the pollution levels in their immediate locality.
- The accuracy of the prediction model can be improved based on the data sensed from the user's surroundings.
- The data obtained can be also be used for further independent analysis.

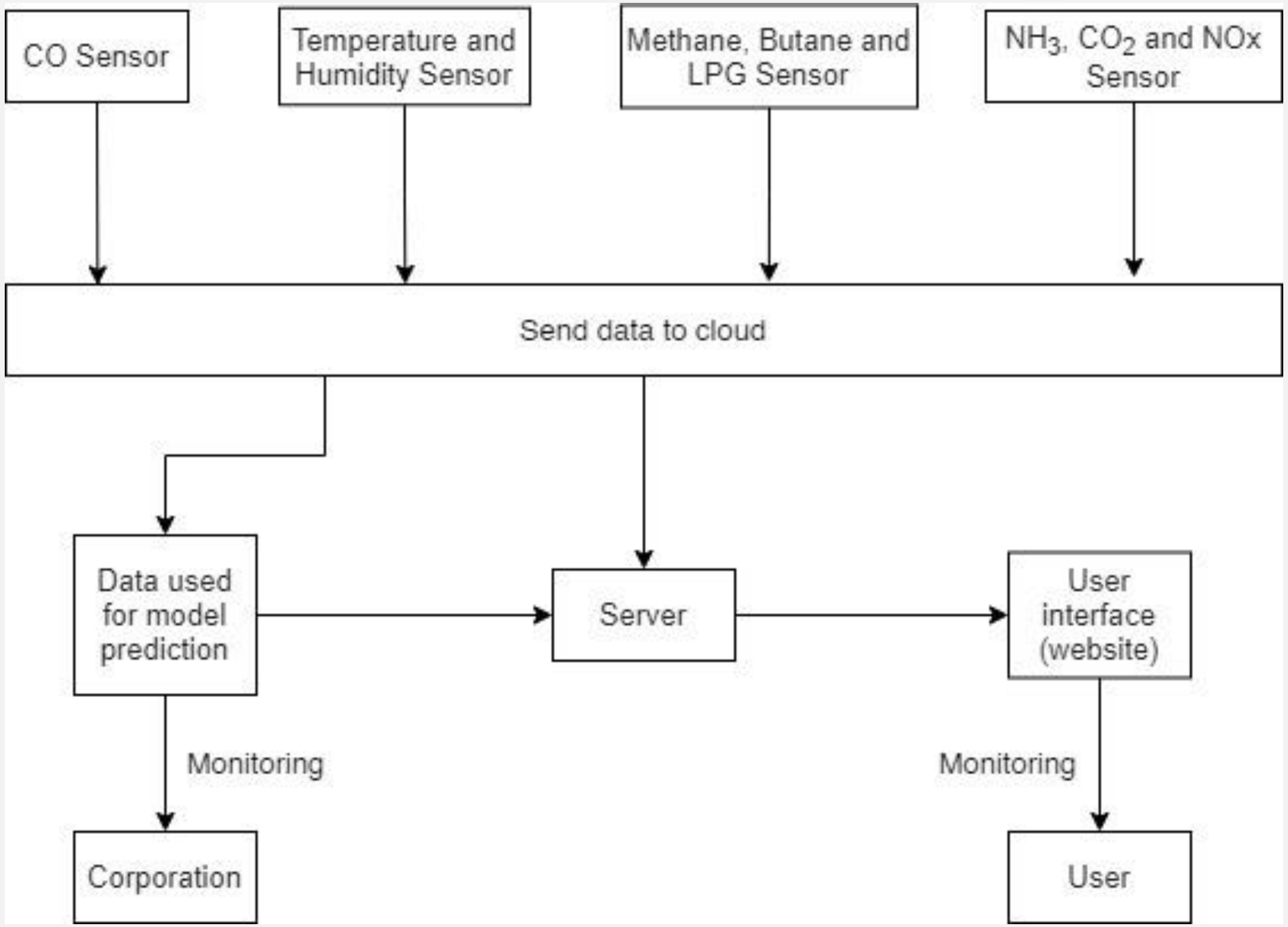
PROTOTYPE DESIGNED

- Due to the unavailability of the proposed sensors to be used due to the pandemic, the current prototype designed uses sensors that we were able to procure from local shops.
- The code is designed in such a way that the other sensors, once procured, can be easily added on later.
- The Prototype collects the data about the concentration of CO, Temperature, Humidity, Nitrogen based gases and Other Gases (CH4, LPG Gas).
- The data is visualized in real time, via a Website which is hosted in the Cloud via Google Firebase.
- Real Time data can also be viewed through a Mobile App called Blynk, which is Cloud Hosted.
- Furthermore, we have trained a model that can be used to predict future concentration levels.

SEQUENCE DIAGRAM



BLOCK DIAGRAM



WORKING PRINCIPLE

Perception Layer:

- Here, the MQ135, MQ7, and MQ2 sensors are used to obtain raw values of the concentration of NH3, CO2, CO, NOx and Methane levels in the atmosphere. The Temperature and humidity sensor is also used to obtain the temperature and humidity in air that will aid us in checking the general air quality. All this information is sent to the processing layer

Processing Layer:

- In this layer, we utilize the Arduino UNO. The data collected from the perception layer is received by the Arduino UNO and processed. There are two main processes that are involved here:
  - **Data Computation:** Here, the raw data obtained from the Gas sensors is used to obtain the proper ppm value of the gas concentrations in the atmosphere. This is done by referring to the datasheet of the gas sensors and performing appropriate calibration. The DHT Library can be used to obtain data from the Temperature and Humidity sensor.
  - **Data Serialization:** Data serialization is the process of converting data objects present in complex data structures into a byte stream for storage, transfer and distribution purposes. To achieve this, serialized data will be relayed to the Node MCU from the Arduino end by using ArduinoJSON. JSON (JavaScript Object Notation) is a popular, lightweight format for storing and transporting data, and the ArduinoJSON supports serialization and deserialization. The serialized data from one end will be deserialized in the other end of the node.

WORKING PRINCIPLE

Transport Layer:

- Transport Layer provides transparent transfer of data between end users, providing reliable data transfer services to the upper layers. The transport protocol that's being utilized is UART, which stands for Universal Asynchronous Receiver/Transmitter. It's not a communication protocol like SPI and I2C, but a physical circuit in a microcontroller, or a stand-alone IC. A UART's main purpose is to transmit and receive serial data.

Application Layer

- Here, we define how applications interface with the lower layer protocols to send data over the network. The application data, in files, is encoded by the application layer protocol. It is encapsulated in the transport layer protocol which provides transaction-oriented communication over the network. Application layer protocol enables process-to-process connection using ports. Here we are using the HTTP Protocol (Hypertext Transfer Protocol).

DESIGNED ML MODEL

```
#Function to calculate so2 individual pollutant index(si)
def calculate_si(so2):
    si=0
    if (so2<=40):
     si= so2*(50/40)
    if (so2>40 and so2<=80):
     si= 50+(so2-40)*(50/40)
    if (so2>80 and so2<=380):
     si= 100+(so2-80)*(100/300)
    if (so2>380 and so2<=800):
     si= 200+(so2-380)*(100/800)
    if (so2>800 and so2<=1600):
     si= 300+(so2-800)*(100/800)
    if (so2>1600):
     si= 400+(so2-1600)*(100/800)
    return si
data['si']=data['so2'].apply(calculate_si)
df= data[['so2','si']]
```

```
def calculate_ni(no2):
    ni=0
    if(no2<=40):
     ni= no2*50/40
    elif(no2>40 and no2<=80):
     ni= 50+(no2-14)*(50/40)
    elif(no2>80 and no2<=180):
     ni= 100+(no2-80)*(100/100)
    elif(no2>180 and no2<=280):
     ni= 200+(no2-180)*(100/100)
    elif(no2>280 and no2<=400):
     ni= 300+(no2-280)*(100/120)
    else:
     ni= 400+(no2-400)*(100/120)
    return ni
data['ni']=data['no2'].apply(calculate_ni)
df= data[['no2','ni']]
```

```
def calculate_(rspm):
    rpi=0
    if(rpi<=30):
     rpi=rpi*50/30
    elif(rpi>30 and rpi<=60):
     rpi=50+(rpi-30)*50/30
    elif(rpi>60 and rpi<=90):
     rpi=100+(rpi-60)*100/30
    elif(rpi>90 and rpi<=120):
     rpi=200+(rpi-90)*100/30
    elif(rpi>120 and rpi<=250):
     rpi=300+(rpi-120)*(100/130)
    else:
     rpi=400+(rpi-250)*(100/130)
    return rpi
data['rpi']=data['rspm'].apply(calculate_si)
df= data[['rspm','rpi']]
```

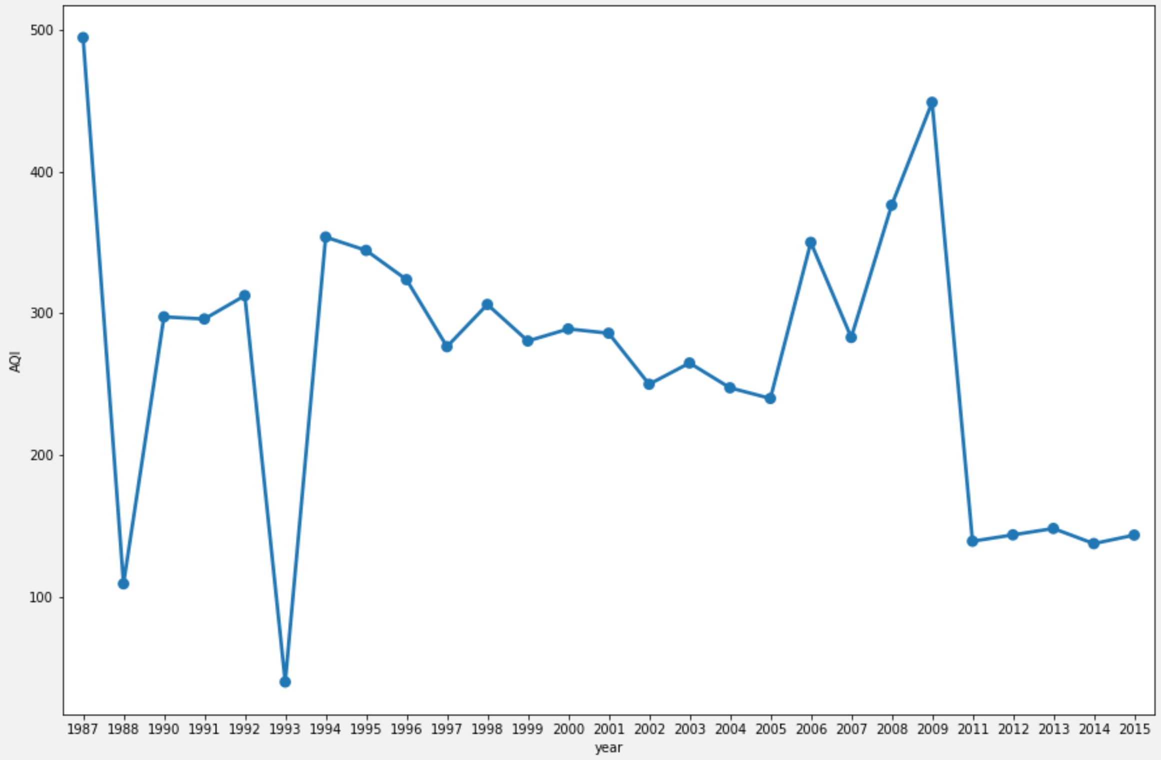
DESIGNED ML MODEL

```
def calculate_spi(spm):
    spi=0
    if(spm<=50):
     spi=spm
    if(spm<50 and spm<=100):
     spi=spm
    elif(spm>100 and spm<=250):
     spi= 100+(spm-100)*(100/150)
    elif(spm>250 and spm<=350):
     spi=200+(spm-250)
    elif(spm>350 and spm<=450):
     spi=300+(spm-350)*(100/80)
    else:
     spi=400+(spm-430)*(100/80)
    return spi
data['spi']=data['spm'].apply(calculate_spi)
df= data[['spm','spi']]
```

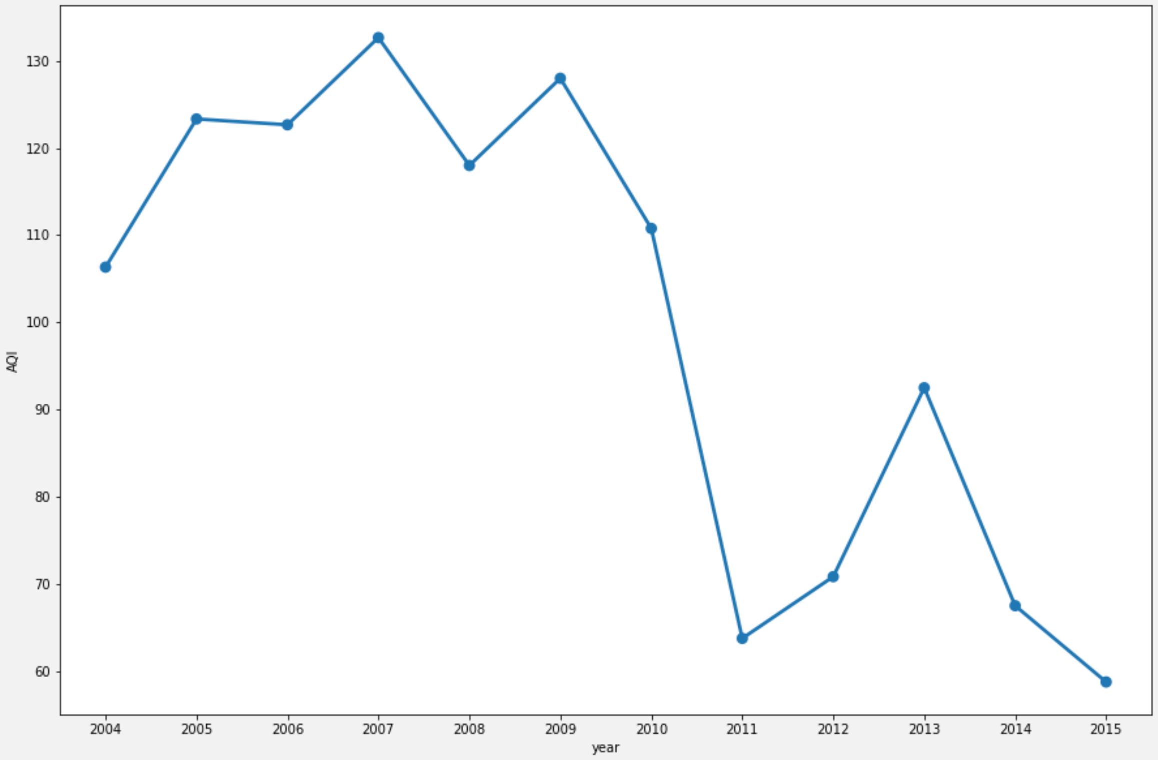
```
#function to calculate the air quality index (AQI) of every data value
#its is calculated as per indian govt standards
def calculate_aqi(si,ni,spi,rpi):
    aqi=0
    if(si>ni and si>spi and si>rpi):
     aqi=si
    if(spi>si and spi>ni and spi>rpi):
     aqi=spi
    if(ni>si and ni>spi and ni>rpi):
     aqi=ni
    if(rpi>si and rpi>ni and rpi>spi):
     aqi=rpi
    return aqi
data['AQI']=data.apply(lambda x:calculate_aqi(x['si'],x['ni'],x['spi'],x['rpi']),axis=1)
df= data[['sampling_date','state','si','ni','rpi','spi','AQI']]
df.head()
```

DESIGNED ML MODEL

AQI vs. Year Plot:



Delhi



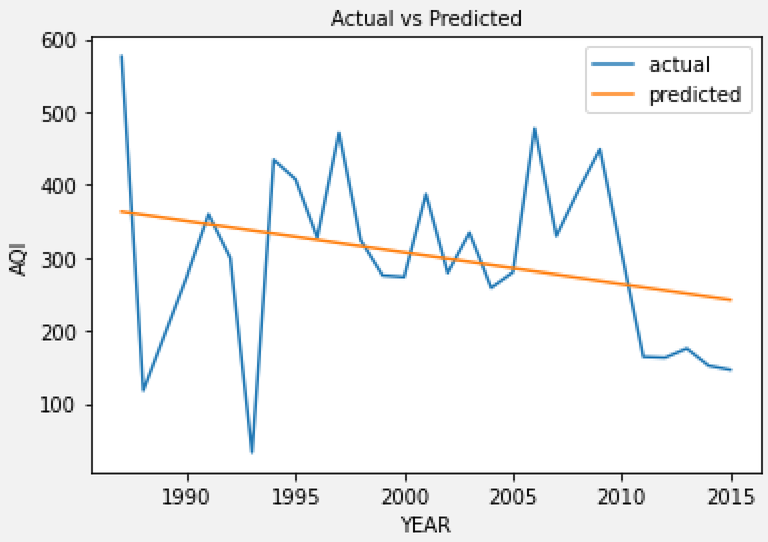
Chennai

DESIGNED ML MODEL

```
# Applying GRADIENT DESCENT

alpha = 0.01 #Step size
iterations = 3000 #No. of iterations
m = y.size #No. of data points
np.random.seed(4) #Setting the seed
theta = np.random.rand(2) #Picking random values to start with

def gradient_descent(x, y, theta, iterations, alpha):
    past_costs = []
    past_thetas = [theta]
    for i in range(iterations):
        prediction = np.dot(x, theta)
        error = prediction - y
        cost = 1/(2*m) * np.dot(error.T, error)
        past_costs.append(cost)
        theta = theta - (alpha * (1/m) * np.dot(x.T, error))
        past_thetas.append(theta)

    return past_thetas, past_costs
past_thetas, past_costs = gradient_descent(x, y, theta, iterations, alpha)
theta = past_thetas[-1]

#Printing the results...
print("Gradient Descent: {:.2f}, {:.2f}".format(theta[0], theta[1]))
```

DESIGNED ML MODEL

<u>Delhi:</u>

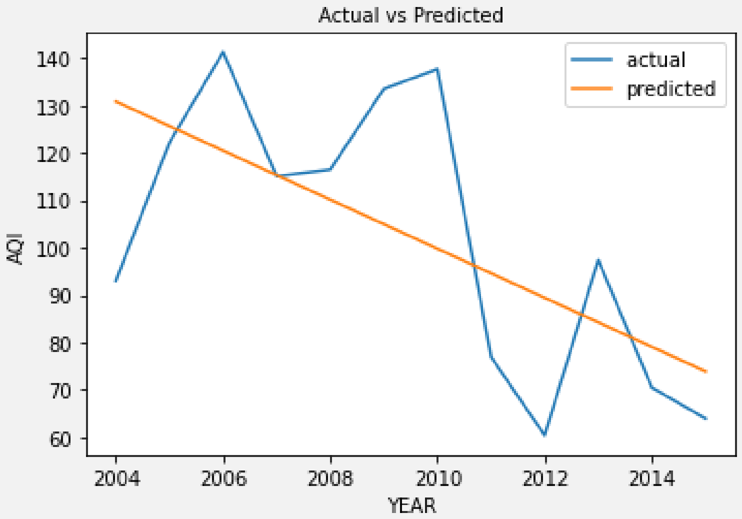


	year	AQI	Actual	Predicted
28	2015	101.258882	101.258882	197.637906
27	2014	102.785280	102.785280	197.818770
26	2013	106.729246	106.729246	197.999634
25	2012	99.696254	99.696254	198.180498
24	2011	117.824783	117.824783	198.361362
23	2010	188.283360	188.283360	198.542226
22	2009	221.368166	221.368166	198.723089
21	2008	214.378174	214.378174	198.903953
20	2007	211.160807	211.160807	199.084817
19	2006	219.623267	219.623267	199.265681
18	2005	207.546049	207.546049	199.446545
17	2004	164.661496	164.661496	199.627409
16	2003	163.875510	163.875510	199.808272
15	2002	149.706871	149.706871	199.989136
14	2001	205.138247	205.138247	200.170000
13	2000	195.772377	195.772377	200.350864
12	1999	225.439218	225.439218	200.531728

	year	AQI	Actual	Predicted
11	1998	234.740657	234.740657	200.712591
10	1997	220.903571	220.903571	200.893455
9	1996	216.850189	216.850189	201.074319
8	1995	227.102233	227.102233	201.255183
7	1994	205.636343	205.636343	201.436047
6	1993	65.754613	65.754613	201.616911
5	1992	177.485106	177.485106	201.797774
4	1991	238.060052	238.060052	201.978638
3	1990	239.071032	239.071032	202.159502
2	1989	236.513310	236.513310	202.340366
1	1988	211.076502	211.076502	202.521230
0	1987	242.438652	242.438652	202.702094

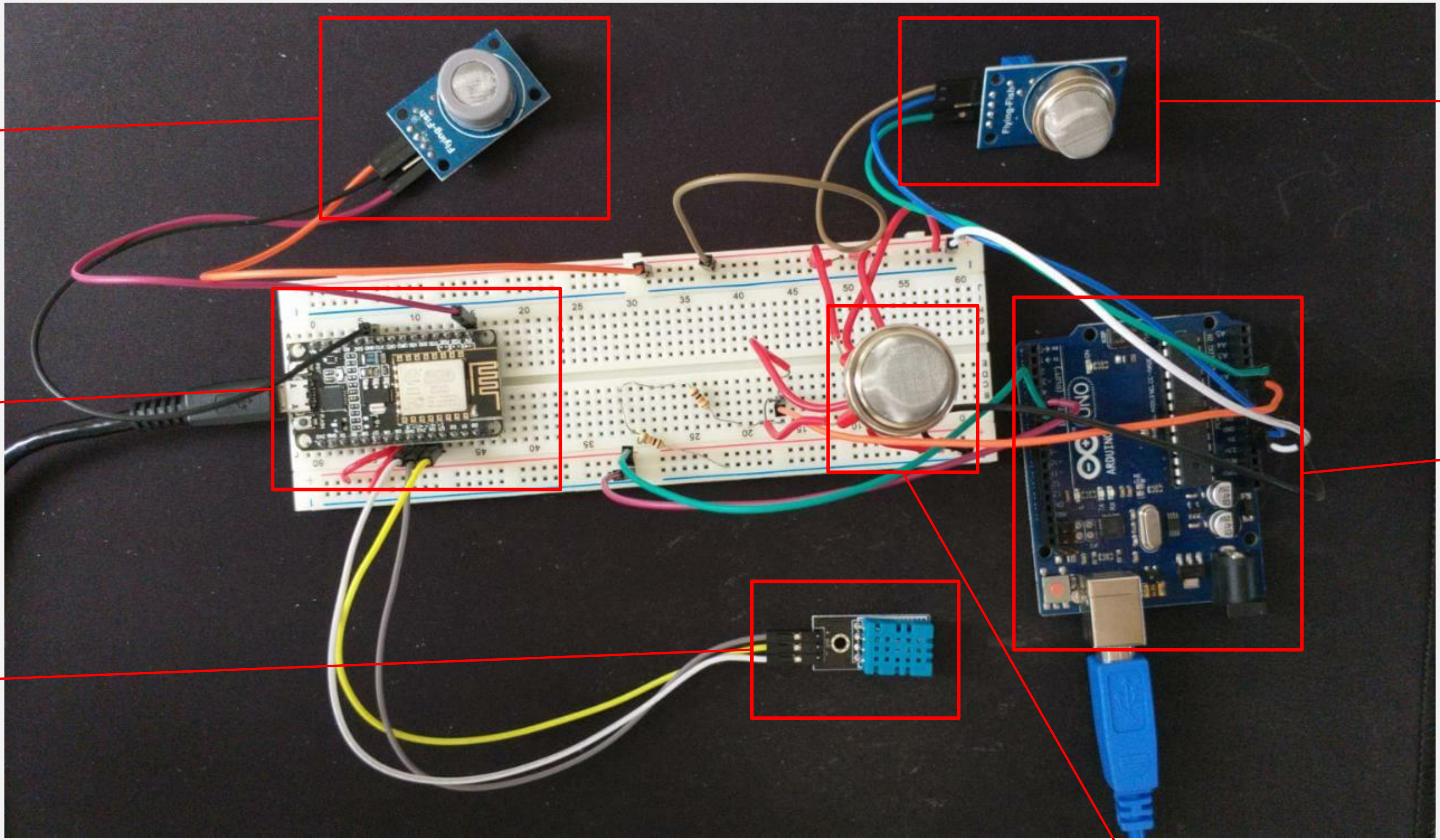
DESIGNED ML MODEL

Chennai:



	year	AQI	Actual	Predicted
11	2015	63.959596	63.959596	73.895559
10	2014	70.470657	70.470657	79.070912
9	2013	97.416667	97.416667	84.246265
8	2012	60.412286	60.412286	89.421618
7	2011	76.930233	76.930233	94.596971
6	2010	137.656463	137.656463	99.772324
5	2009	133.511905	133.511905	104.947676
4	2008	116.470238	116.470238	110.123029
3	2007	115.049242	115.049242	115.298382
2	2006	141.247549	141.247549	120.473735
1	2005	122.186508	122.186508	125.649088
0	2004	93.039907	93.039907	130.824441

PROTOTYPE CIRCUIT



MQ-7 Sensor (CO Sensor)

NodeMCU (ESP8266 Wifi Module)

DHT11 Sensor (Temperature and Humidity Sensor)

MQ-2 Sensor (CH4 Sensor)

Arduino UNO

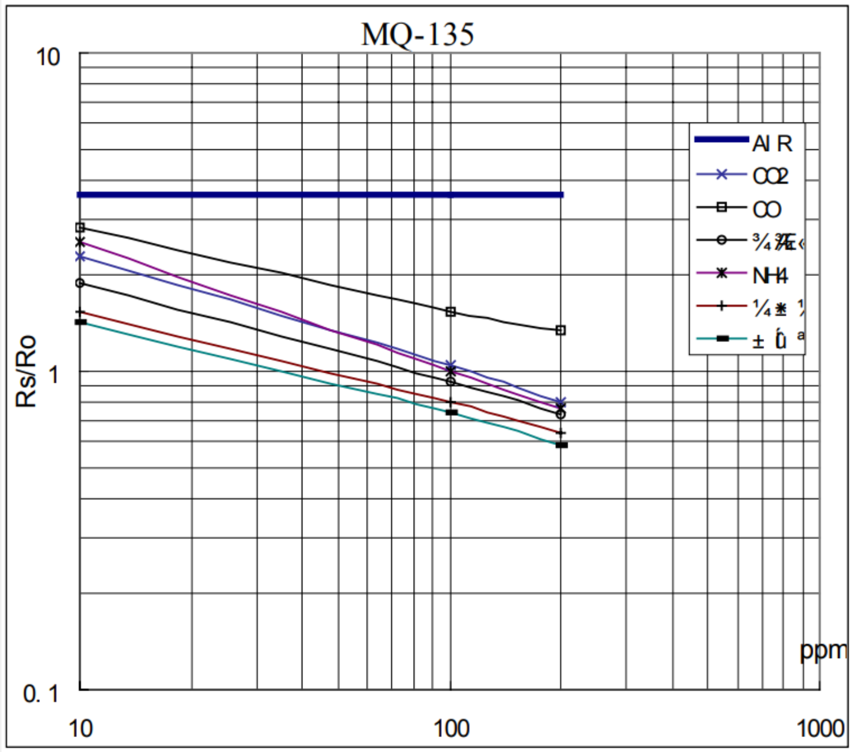
MQ-135 Sensor (NH3 and NOx Sensor)

IMPLEMENTATION

- For recording the air quality level, we have used:
  - **Gas sensors:** To determine CO2, NOx, NH3, CO, Methane and Smoke level, we have used gas sensors like the MQ135, MQ7, and the MQ2.
  - **Temperature and Humidity Sensor:** Temperature and Humidity also plays a crucial role in determining the overall air quality, so we use the DHT11 sensor for the same.
- The data we obtain from the sensors is sent at regular intervals to the Thingspeak IoT Cloud as well as the Blynk IoT Cloud using the Node MCU.
- The Thingspeak IoT Cloud reads sensor value and performs a graph visualization over time.
- The website, which is hosted in the Google Firebase Cloud, communicates with the Thingspeak IoT cloud and obtains the real time visualization of the sensor data, which can now be viewed through the website.
- At the same time, The Blynk App obtains sensor data from the Blynk IoT cloud and displays it real time using widgets.
- The values that are being obtained from the sensors can be exported as a .csv file from the Thingspeak IoT Dashboard. This file is sent to our prediction model to check and improve upon the accuracy of the model.
- The visualization of the predicted values vs. the original values is performed in Google Collaboratory.
- Due to Geographical and Time Constraints, the model is initially trained using existing datasets of India's Air Pollution from 2004 to 2020. The training of the model is done in Google Collaboratory.

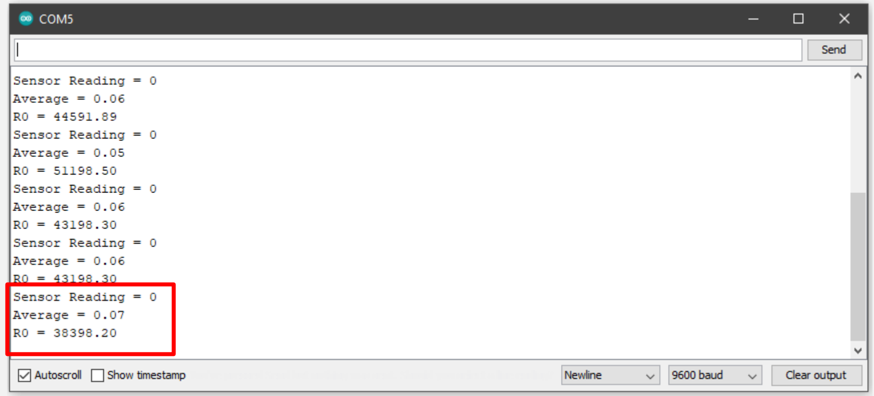
MQ SENSOR CALIBRATION

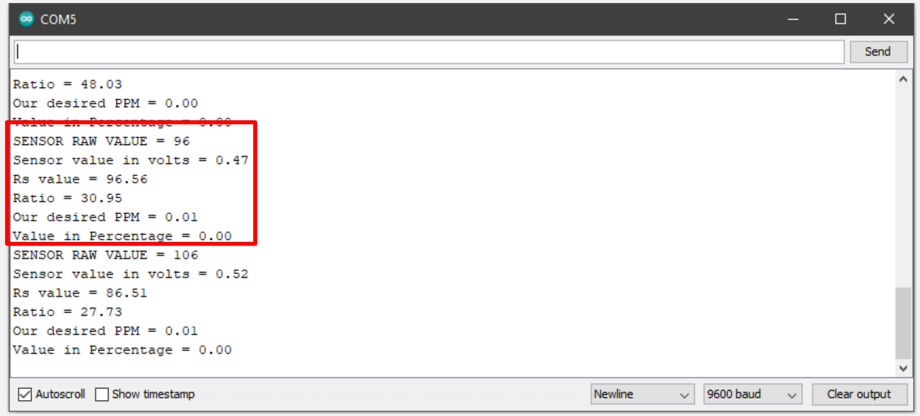
```
void setup() {
  Serial.begin(9600); //Baud rate
  pinMode(A0,INPUT);
}

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
  Serial.print("Average = ");
  Serial.println(sensorValue);
  sensor_volt = sensorValue*(5.0/1023.0); //Convert average to voltage
  RS_air = ((5.0*10.0)/sensor_volt)-10.0; //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

}
```



MQ SENSOR CALIBRATION

```
int gas_sensor = A0; //Sensor pin
float m = -0.3376; //Slope
float b = 0.7165; //Y-Intercept
float R0 = 3.12; //Sensor Resistance in fresh air from previous code

void setup() {
  Serial.begin(9600); //Baud rate

  pinMode(gas_sensor, INPUT); //Set gas sensor as input
}

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 = analogRead(gas_sensor); //Read analog values of sensor
  Serial.print("SENSOR RAW VALUE = ");
  Serial.println(sensorValue);
  sensor_volt = sensorValue*(5.0/1023.0); //Convert analog values to voltage
  Serial.print("Sensor value in volts = ");
  Serial.println(sensor_volt);
  RS_gas = ((5.0*10.0)/sensor_volt)-10.0; //Get value of RS in a gas
  Serial.print("Rs value = ");
  Serial.println(RS_gas);
  ratio = RS_gas/R0;  // Get ratio RS_gas/RS_air

  Serial.print("Ratio = ");
  Serial.println(ratio);
  float ppm_log = (log10(ratio)-b)/m; //Get ppm value in linear scale according to the the ratio value
  float ppm = pow(10, ppm_log); //Convert ppm value to log scale
  Serial.print("Our desired PPM = ");
  Serial.println(ppm);
  double percentage = ppm/10000; //Convert to percentage
  Serial.print("Value in Percentage = "); //Load screen buffer with percentage value
  Serial.println(percentage);
  delay(1000);
}
```

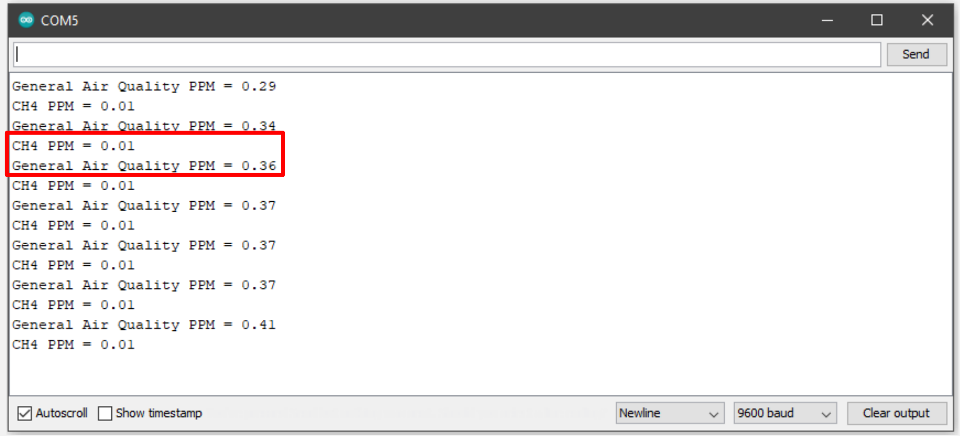




CODE

```
#include <SoftwareSerial.h>
#include <ArduinoJson.h>
SoftwareSerial s(5,6); // setting 5-Rx pin and 6-Tx pin

int gas_sensor = A0; //Sensor pin
float m = -0.3376; //Slope
float b = 0.7165; //Y-Intercept
float R0 = 2.82; //Sensor Resistance in fresh air from previous code

int CH4_sensor = A1; //Sensor pin
float m1 = -0.3720; //Slope
float b1 = 1.3492; //Y-Intercept
float R01 = 1.45; //Sensor Resistance

void setup() {


  Serial.begin(9600);      // PC to Arduino Serial Monitor
  pinMode(gas_sensor, INPUT);
  pinMode(CH4_sensor,INPUT);
  pinMode(A2,INPUT); //For DHT Sensor
  s.begin(57600);
 }

 StaticJsonBuffer<500> jsonBuffer;
 JsonObject& root = jsonBuffer.createObject();

void loop() {
  float sensor_volt; //Define variable for sensor voltage
  float RS_gas; //Define variable for sensor resistance
  float ratio; //Define variable for ratio
  float 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*10.0)/sensor_volt)-10.0; //Get value of RS in a gas
  ratio = RS_gas/R0;  // Get ratio RS_gas/RS_air
  double ppm_log = (log10(ratio)-b)/m; //Get ppm value in linear scale according to the the ratio value
  double ppm = pow(10, ppm_log); //Convert ppm value to log scale
  Serial.print("General Air Quality PPM = ");
  Serial.println(ppm);
```

```
  float sensor_volt1; //Define variable for sensor voltage
  float RS_gas1; //Define variable for sensor resistance
  float ratio1; //Define variable for ratio
  float sensorValue1 = analogRead(CH4_sensor); //Read analog values of sensor
  sensor_volt1 = sensorValue1*(5.0/1023.0); //Convert analog values to voltage
  RS_gas1 = ((5.0*10.0)/sensor_volt1)-10.0; //Get value of RS in a gas
  ratio1 = RS_gas1/R01;  // Get ratio RS_gas/RS_air
  double ppm_log1 = (log10(ratio1)-b1)/m1; //Get ppm value in linear scale according to the the ratio value
  double ppm1 = pow(10, ppm_log1); //Convert ppm value to log scale
  Serial.print("CH4 PPM = ");
  Serial.println(ppm1);

  root["ppm1"] = ppm;
  root["ppm2"] = ppm1;

  if(s.available()>0)
  {
    root.printTo(s);
  }

  delay(4000);                              // wait for 4 sec
 }
```



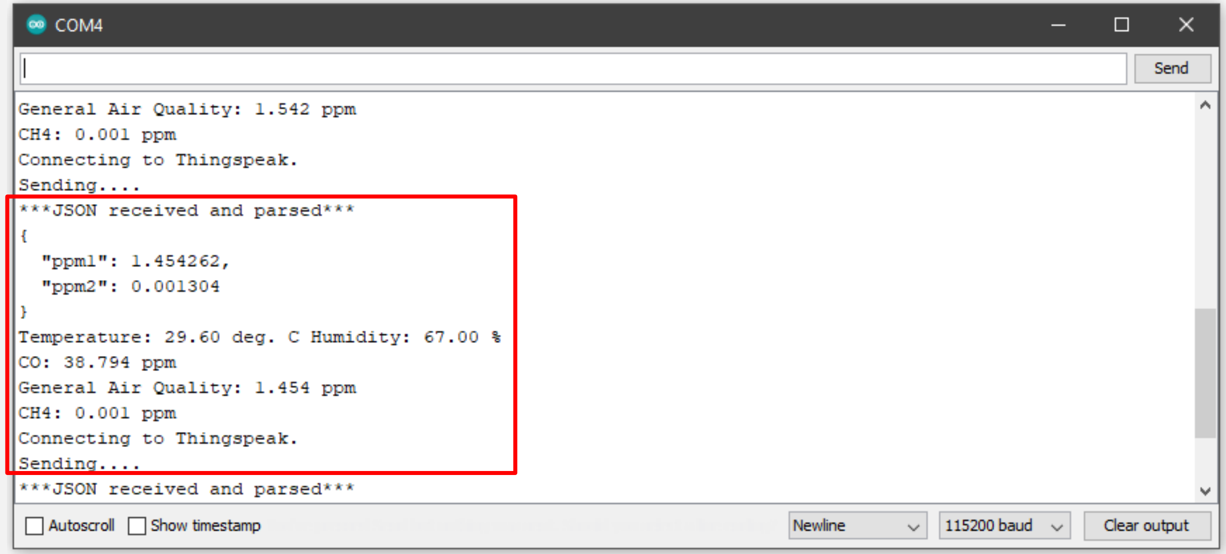
CODE

```
#include <DHT.h>
#include <ESP8266WiFi.h>
#include <SoftwareSerial.h>
#include <ArduinoJson.h>

SoftwareSerial s(D6,D5); // setting D6-Rx ping and D5-Tx pin

#define DHTPIN 2           //DHT11 is connected to GPIO Pin 2
int sensorValue;           //the AOUT pin of the CO sensor goes into analog pin A0 of the arduino
float m = -0.6527; //Slope
float b = 1.30; //Y-Intercept
float R0 = 21.91; //Sensor Resistance in fresh air from previous code

String apiKey = "VK8KPOZH48MIM81E";     //  Enter your Write API key from ThingSpeak
const char* ssid =  "[redacted]";     // Enter your WiFi Network's SSID
const char* pass =  "[redacted]"; // Enter your WiFi Network's Password
const char* server = "api.thingspeak.com";

float humi;
float temp;

DHT dht(DHTPIN, DHT11);
WiFiClient client;

void setup()
{
       Serial.begin(115200);
       s.begin(57600);
       delay(10);
       dht.begin();

       pinMode(16,INPUT);
       Serial.println("Connecting to ");
       Serial.println(ssid);


       WiFi.begin(ssid, pass);

      while (WiFi.status() != WL_CONNECTED)
     {
            delay(100);
            Serial.print("*");
     }
```

```
      Serial.println("");
      Serial.println("***WiFi connected***");
      Serial.println(" ");

}

void loop()
{
      s.write("s");
      StaticJsonBuffer<1000> jsonBuffer;
      JsonObject& root = jsonBuffer.parseObject(s);

      if (root == JsonObject::invalid())
      {
        return;
      }

      Serial.println("***JSON received and parsed***");
      root.prettyPrintTo(Serial);
      float ppm1 = root["ppm1"];
      float ppm2 = root["ppm2"];
      Serial.println(" ");

      float sensor_volt; //Define variable for sensor voltage
      float RS_gas; //Define variable for sensor resistance
      float ratio; //Define variable for ratio
      int sensorValue = analogRead(0); //Read analog values of sensor

      humi = dht.readHumidity();
      temp = dht.readTemperature();

      sensor_volt = sensorValue*(5.0/1023.0); //Convert analog values to voltage
      RS_gas = ((5.0*10.0)/sensor_volt)-10.0; //Get value of RS in a gas
      ratio = RS_gas/R0;  // Get ratio RS_gas/RS_air

      double ppm_log = (log10(ratio)-b)/m; //Get ppm value in linear scale according to the the ratio value
      double ppm = pow(10, ppm_log); //Convert ppm value to log scale
      delay(2000);

      if (client.connect(server,80))   //   "184.106.153.149" or api.thingspeak.com
      {
       String sendData = apiKey+"&field1="+String(temp)+"&field2="+String(humi)+"&field3="+String(ppm)+"&field4="+String(ppm1)+"&field5="+String(ppm2)+"\r\n\r\n";
```

CODE

```
        //Serial.println(sendData);

        client.print("POST /update HTTP/1.1\n");
        client.print("Host: api.thingspeak.com\n");
        client.print("Connection: close\n");
        client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
        client.print("Content-Type: application/x-www-form-urlencoded\n");
        client.print("Content-Length: ");
        client.print(sendData.length());
        client.print("\n\n");
        client.print(sendData);

        Serial.print("Temperature: ");
        Serial.print(temp);
        Serial.print(" deg. C Humidity: ");
        Serial.print(humi);
        Serial.println(" %");
        Serial.print("CO: ");
        Serial.print(ppm, 3);  // prints the value read
        Serial.println(" ppm");
        Serial.print("General Air Quality: ");
        Serial.print(ppm1, 3);  // prints the value read
        Serial.println(" ppm");
        Serial.print("CH4: ");
        Serial.print(ppm2, 3);  // prints the value read
        Serial.println(" ppm");
        Serial.println("Connecting to Thingspeak.");
        }

       client.stop();

       Serial.println("Sending....");

 delay(10000);
}
```



CODE SNIPPETS FOR THE APP

```
#define BLYNK_PRINT Serial


#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <DHT.h>

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "g3lkjclkZ4bW4jtN-jRR-4xSYGggwHN8";

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "[redacted]";
char pass[] = "[redacted]";
```

```
BlynkTimer timer;
```

```
  // You can send any value at any time.
  // Please don't send more that 10 values per second.
  Blynk.virtualWrite(V5, t);
  Blynk.virtualWrite(V6, h);
  Blynk.virtualWrite(V7, sensorValue);
}

void setup()
{
  // Debug console
  Serial.begin(9600);

  Blynk.begin(auth, ssid, pass);
  // You can also specify server:
  //Blynk.begin(auth, ssid, pass, "blynk-cloud.com", 8442);
  //Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 8442);

  pinMode(16,OUTPUT);
  dht.begin();

  // Setup a function to be called every second
  timer.setInterval(1000L, sendSensor);
}

void loop()
{
  Blynk.run();
  timer.run();
}
```

CODE SNIPPETS OF THE WEBSITE

```html
<html lang="en" dir="ltr">
  <head>
  <meta charset="utf-8">
  <title>IoT PROJECT</title>
  <link rel="stylesheet" type = "text/css" href="style1.css">
  <script src="https://cdn.jsdelivr.net/npm/chart.js@3.2.0/dist/chart.min.js"> </script>
  </head>
  <body>
  <h1>AIR POLLUTION MONITORING SYSTEM</h1>
  <div class="topnav">
  <a class="active" href="index.html">Home</a>
  <a class href="Air_pollution1.html">Temperature</a>
  <a class href="Air_pollution2.html">Humidity</a>
  <a class href="Air_pollution3.html">CO</a>
  <a class href="Air_pollution4.html">Nitrogen Based</a>
  <a class href="Air_pollution5.html">Other Gases</a>
  </div>
     <h2>AIR POLLUTTION</h2>
     <div class="home1">
       <h3>Introduction</h3>
       <p>Air pollution is a mix of hazardous substances from both human-made and natural sources. Air pollution is a familiar environmental
          health hazard. We know what we're looking at when brown haze settles over a city, exhaust billows across a busy highway, or a plume rises from a smokestack.
          Some air pollution is not seen, but its pungent smell alerts you.</p>
       <p>From smog hanging over cities to smoke inside the home, air pollution poses a major threat to health and climate.
          The combined effects of ambient (outdoor) and household air pollution cause about seven million premature deaths every year, largely as a result of increased
          mortality from stroke, heart disease, chronic obstructive pulmonary disease, lung cancer and acute respiratory infections.</p>
       <button class="b1" type="button" onclick="document.location='https://climatekids.nasa.gov/air-pollution/'">Causes of Air pollution</button>
     </div>
     <div class="home2">
       <h3>Health and Environmental Effects</h3>
       <p>Even healthy people can experience health impacts from polluted air including respiratory irritation or breathing difficulties during exercise or
          outdoor activities. Your actual risk of adverse effects depends on your current health status, the pollutant type and concentration, and the length of your
          exposure to the polluted air. It can also cause other health problems including:</p>
       <ul>
         <li>Aggravated respiratory disease such as emphysema, bronchitis and asthma</li>
         <li>Lung damage, even after symptoms such as coughing or a sore throat disappear</li>
         <li>Wheezing, chest pain, dry throat, headache or nausea</li>
         <li>Reduced resistance to infections</li>
         <li>Increased fatigue and weakened athletic performance</li>
       </ul>
       <button class="b1" type="button" onclick="document.location='https://www.tecamgroup.com/effects-air-pollution-environment/'">Effects of Air pollution</button>
     </div>
  </body>
</html>
```

CODE SNIPPETS OF THE WEBSITE

```html
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
  <meta charset="utf-8">
  <title>IoT PROJECT</title>
  <link rel="stylesheet" type = "text/css" href="style1.css">
  <script src="https://cdn.jsdelivr.net/npm/chart.js@3.2.0/dist/chart.min.js"> </script>
  </head>
  <body>
  <h1>AIR POLLTUION MONITORING SYSTEM</h1>
  <div class="topnav">
  <a class="active" href="index.html">Home</a>
  <a class href="Air_pollution1.html">Temperature</a>
  <a class href="Air_pollution2.html">Humidity</a>
  <a class href="Air_pollution3.html">CO</a>
  <a class href="Air_pollution4.html">Nitrogen Based</a>
  <a class href="Air_pollution5.html">Other Gases</a>
  </div>
  <div class="home1">
    <h3>Temperature Measurement</h3>
    <iframe width="450" height="260" style="border: 2px solid #000000;"
     src="https://thingspeak.com/channels/1371486/charts/1?bgcolor=%23ffffff&color=%23d62020&dynamic=true&results=10&type=line"></iframe>
  </div>
  <div class="home2">
    <h3>Temperature</h3>
    <p>Temperature is a physical quantity that expresses hot and cold. It is the manifestation of thermal energy, present in all matter, which is the source
       of the occurrence of heat, a flow of energy, when a body is in contact with another that is colder or hotter. </p>
    <p>In India, temperatures typically range from -2 °C to 40 °C, but can reach 47 °C in summer and -4 °C in winter.</p>
    <button class="b1" type="button" onclick="document.location='https://www.mapsofindia.com/maps/india/annualtemperature.htm'">Temperature in India</button>
  </div>

  </body>
</html>
```

Graph Visualization obtained from Thingspeak IoT Cloud

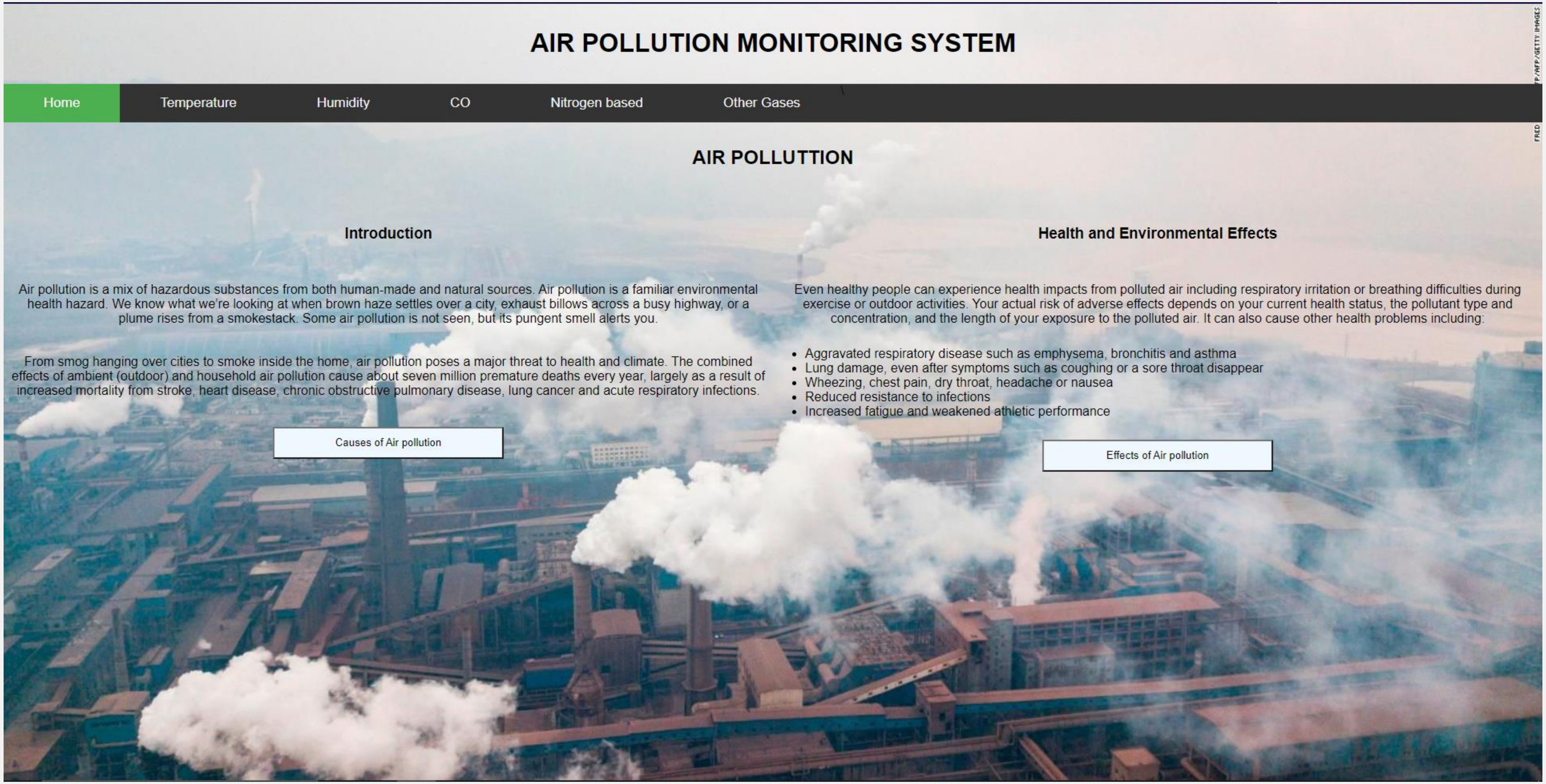
CODE SNIPPETS OF THE WEBSITE

```css
body{
   background-image: url("https://wallpapercave.com/wp/wp6100414.jpg");
   background-size: cover;
   background-repeat: no-repeat;
}
h1{
   padding-top: 10px;
   padding-bottom: 10px;
   text-align: center;
}
h2{
   padding-top: 10px;
   padding-bottom: 10px;
   text-align: center;
}
h3{
   padding-top: 10px;
   padding-bottom: 10px;
   text-align: center;
   margin: 30px;
}
p{
   padding: 10px;
}
body {
   margin: 0;
   font-family: Arial, Helvetica, sans-serif;
}
.home1{
   width: 50%;
   float: left;
}
.home2{
   width: 50%;
   float: right;
}
.topnav {
   overflow: hidden;
   background-color: #333;
}

```

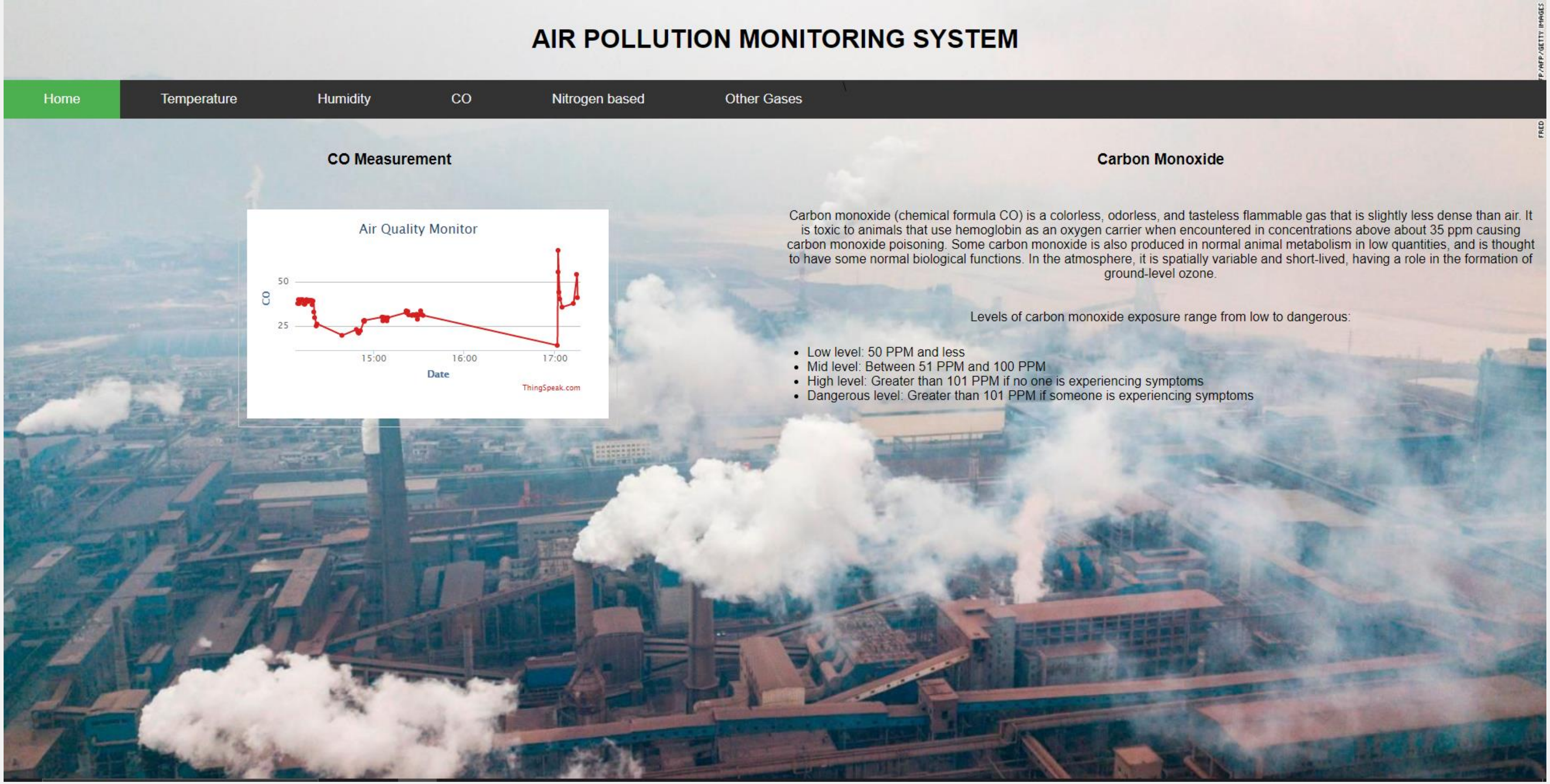
```css
.topnav a {
   float: left;
   color: #f2f2f2;
   text-align: center;
   padding: 14px 50px;
   text-decoration: none;
   font-size: 17px;
}

.topnav a:hover {
   background-color: #ddd;
   color: black;
}

.topnav a.active {
   background-color: #4CAF50;
   color: white;
}
.topnav a.login {
   float: right;
   background-color: #4CAF50;
   color: white;
}
.b1{
   background-color: rgb(240,248,255);
   color: black;
   text-align: center;
   alignment: centre;
   width: 80%;
   margin: 10px;
   height: 45px;
   border: 1px, solid;
}
iframe{
   padding: 10px;
   margin-left: 10%;
   align-items: center;
}
img{
   height: 400px;
   width: 400px;
   margin-left: 10%;
}

```

Website Styling (CSS)

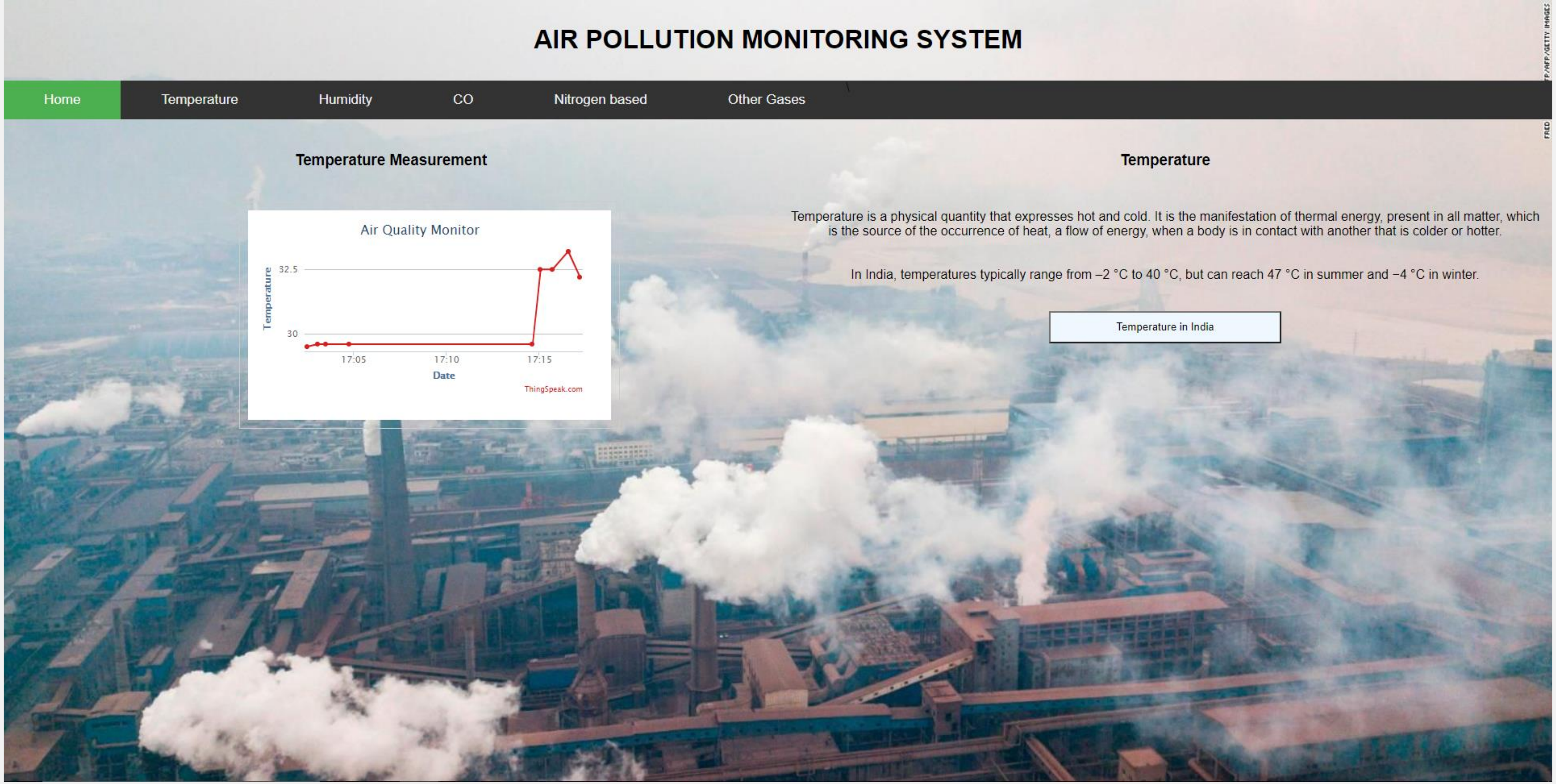
VISUALIZATION IN WEBSITE



VISUALIZATION IN WEBSITE



VISUALIZATION IN WEBSITE



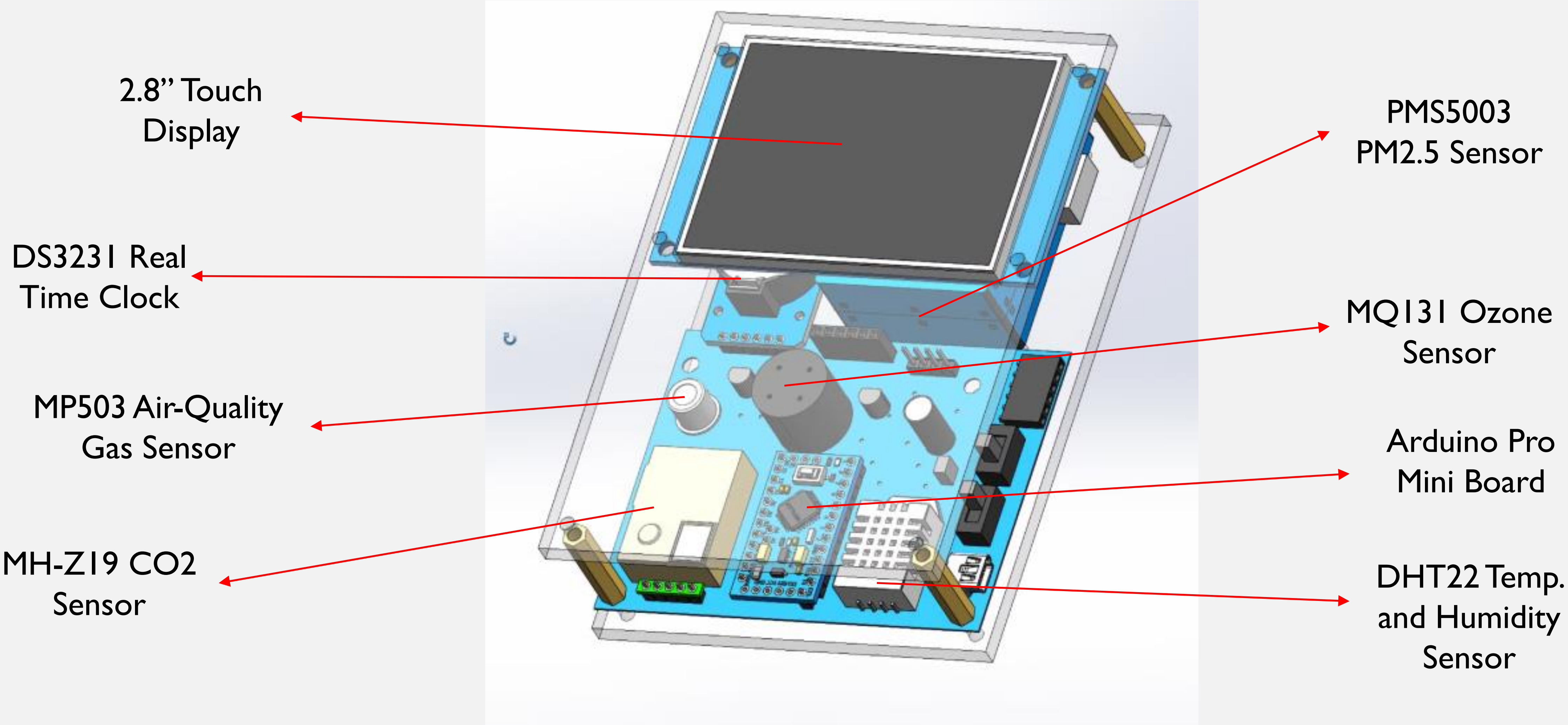
REAL TIME APPLICATIONS

- The readings collected by the sensors will be displayed on an interactive website.
- This will help the common people monitor the levels of air pollution in their neighborhood.
- The government can also use the data to implement dynamic pollution control measures (Eg: Like Delhi’s odd-even rule but over a localized area and on particular days rather than over weeks)

FUTURE SCOPE

- The sensors can be miniaturized and integrated into smartphones.
- As a result millions of datapoints can be generated for thousands of locations around the world. This can help to create better predictive models to estimate future patterns of air pollution across the world.
- With this information, governments can take more concrete steps to reduce air pollution like constructing Carbon Capture Systems (CCS) and rezoning industries in highly polluted regions.

FUTURE SCOPE



TEAM MEMBERS AND ROLES

- V. SHRI SARVESH (ECE-108118109)

  Interfacing and Calibrating sensors, and sending sensor data to the IoT Cloud

- ADITHYA SINEESH (ECE-108118005)

  Data Analytics, Development of the Machine Learning model, Training and Prediction

- BALAJI K S (ECE-108118021)

  Cloud Web Hosting, Setting up Communication with the Website and the IoT Cloud.

THANK YOU!