# DIY IOT Hardware Project

Name - Pauras  Manoj Tarle
Roll No.- 200101077

## Main Objective:-

To create a smart fire alarm and air quality monitoring system using iot.

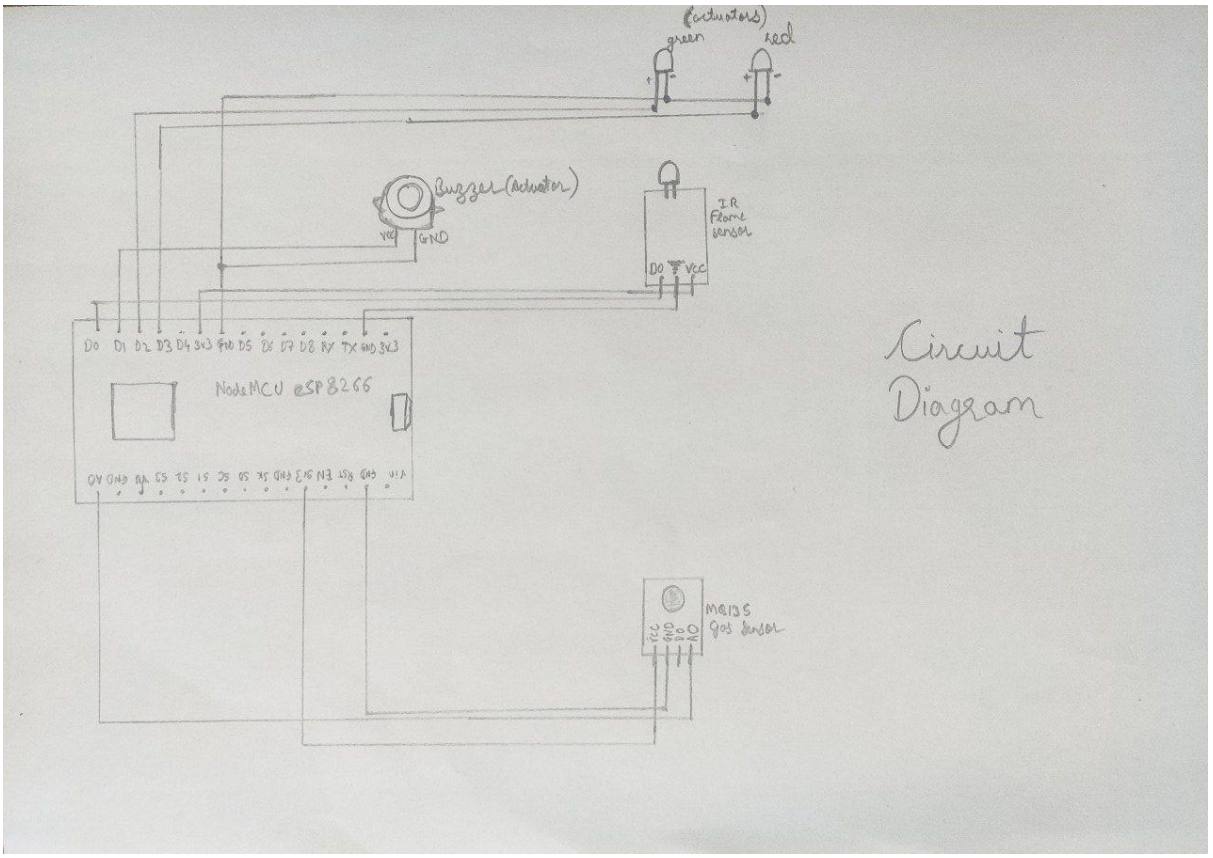
## Components Used-:

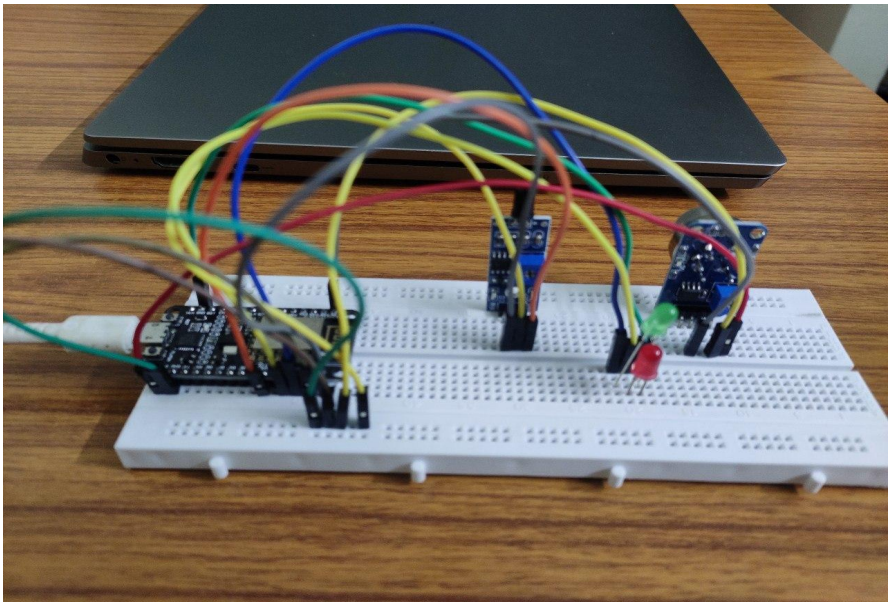
- NodeMCU esp8266
- Gas Sensor (MQ 135)
- IR flame sensor
- Breadboard
- Jumper Wires
- Buzzer
- LEDs
- Micro-usb Cable

## Implemented Attributes:-

**IR Fire Sensor -** This flame sensor or fire sensor module works on the concept that when a flame or fire is burning it emits IR signals. This IR signal is then received by the IR receiver on the fire sensor module to detect the flame or fire**.**
**MQ-135 Gas sensor** - It can detect gases like Ammonia (NH3), sulfur (S), Benzene (C6H6), CO2, and other harmful gases and smoke. This sensor also has a digital and analog output pin. When the level of these gases go beyond a threshold limit in the air the digital pin goes high. This threshold value can be set by using the on-board potentiometer. The analog output pin, outputs an analog voltage which can be used to approximate the level of these gases in the atmosphere.
**LEDs -** Two leds have been used as **actuators** to denote whether the Air Quality Index(AQI) is within safe range or not. They are turned on/turned off using command from cloud application in this project.
**Buzzer -** It has been used as an alarm for fire thus act as an **actuator**. It is connected to GPIO output pin.
**NodeMCU ESP8266** is an open-source firmware and development board specially targeted for IoT based applications. It includes firmware that runs on the **ESP8266 Wi-Fi SoC** and hardware which is based on the ESP-12 module, and like this, it can also be programmed using Arduino IDE and can act as both WiFi Hotspot or can connect to on.
**SMTP2Go Cloud Platform -** This platform is used to **send email** to the owner alerting him about the fire.
**ThingsSpeak Cloud Platform -** This platform has been used to **display the data** acquired by gas sensor in the form of graph. Also this has been used to send **commands** to the system to turn on/turn off the leds.

## Configuration Dlagram:-

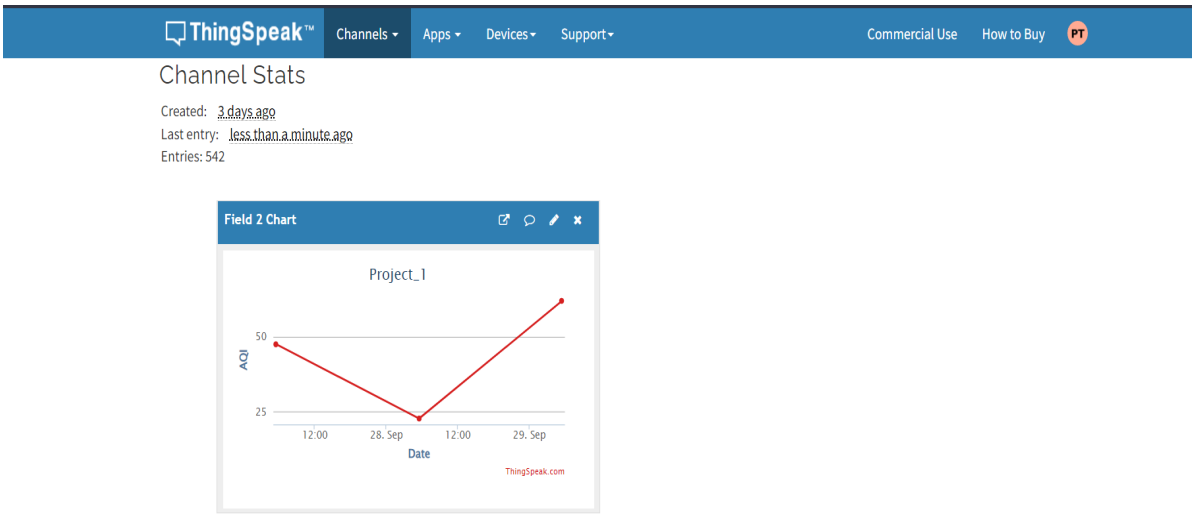


## Sample Outputs -:

```
.............
WiFi Connected.
Server started
Use this URL to connect: http://IP address: 192.168.78.136
/
loop running
61.78
200
checking queue...
  Nothing new.
```

The above picture of serial monitor shows that the wifi module has connected to the hotspot and also shows the ip address of the wifi shield. 61.78 is the AQI of the surrounding and 200 is the http response from thinkspeak cloud paltform which denote that the data from the gas sensor has successfully been uploaded to the cloud. Further checking queue denote that is checking the queue if there are any commands from the thinkspeak cloud platform.



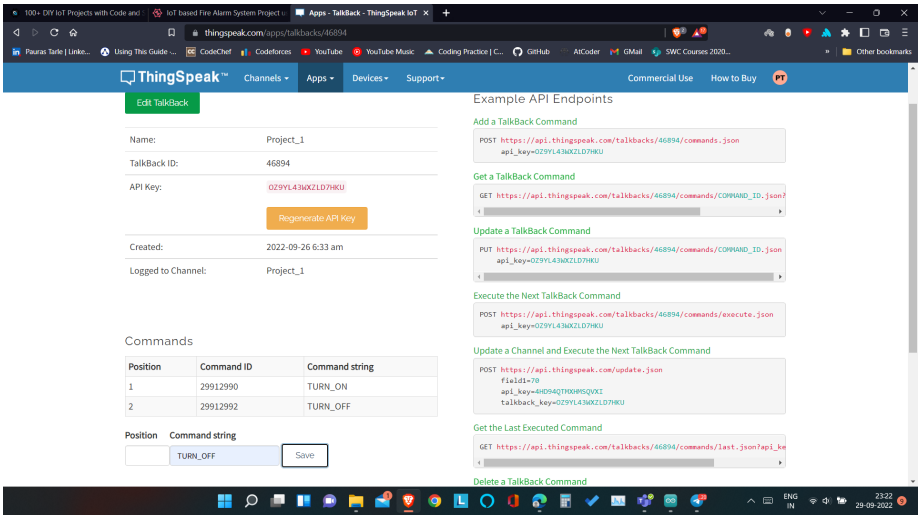

The above picture shows graph of the AQI of surroudings which was aquired from the gas sensor.

```
loop running
63.15
200
checking queue...
  Latest command from queue: 7
TURN_ON
0
```

The above picture shows that there was a command TURN_ON given from the cloud platform

```
loop running
60.80
200
checking queue...
  Latest command from queue: 8
TURN_OFF
0
```

The above picture shows that there was a command TURN_OFF given from the cloud platform




The above picture shows the platform to upload the commands.

## Codes and working Explanation :-

### Setup and initialization-:

```
#include <ESP8266WiFi.h>
#include "ThingSpeak.h"
//----------- Channel details ----------------//
unsigned long Channel_ID = 1872709; // Your Channel ID
const char * myWriteAPIKey = "4HD94QTMXHMSQVXI"; //Your write API key
unsigned long myTalkBackID = 46894;
const char * myTalkBackKey = "OZ9YL43WXZLD7HKU";
//-------------------------------------------//

const char* ssid = "Redmi";
const char* password = "pauras";
char server1[] = "mail.smtp2go.com";
const int Field_Number_1 = 1;
const int Field_Number_2 = 2;
const int flame = D0;
const int buzz = D1;
const int green = D2;
const int red = D3;
int MQ135= A0;
WiFiClient Client;                //define wifi client as client
WiFiServer server(80);

void setup() {

  pinMode(flame,INPUT);
```

```
  pinMode(buzz,OUTPUT);
  pinMode(green,OUTPUT);
  pinMode(red,OUTPUT);
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  ThingSpeak.begin(Client);
  Serial.println("");
  Serial.print("Connecting To: ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi Connected.");
  // Start the server
  server.begin();
  Serial.println("Server started");

  // Print the IP address on serial monitor
  Serial.print("Use this URL to connect: ");
  Serial.print("http://");    //URL IP to be typed in mobile/desktop
browser
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  Serial.println("/");

}
```

In setup, we initialise all the pins and connect the esp8266 module to the wifi and initialise thinspeak library.

**Loop:-**

```
void loop() {
  Serial.println("loop running");

  float AQI = analogRead(A0);
  if (isnan(AQI))
  {
      Serial.println("Failed to read from MQ135 sensor!");
  }
  else
  {
    AQI = AQI/1023*100;
    Serial.println(AQI);
    int x1 = ThingSpeak.writeField(Channel_ID, Field_Number_2, AQI,
myWriteAPIKey);
    Serial.println(x1);
    // if(x1!=200){
    //     Serial.println("Error uploading AQI");
    // }
    String tbURI = String("/talkbacks/") + String(myTalkBackID) +
String("/commands/execute");
    String postMessage =  String("api_key=") + String(myTalkBackKey);
    String newCommand = String();
    String on = "TURN_ON";
    String off = "TURN_OFF";
    int x = httpPOST(tbURI, postMessage, newCommand);
    Client.stop();
    if(x == 200){
      Serial.println("checking queue...");
      // Check for a command returned from TalkBack
      if(newCommand[0] != '0'){

        Serial.print("  Latest command from queue: ");
        //Serial.println(newCommand);
        String nwcmd = String(newCommand[9]);
        //Serial.println(nwcmd);
        Serial.println(newCommand);
        if(nwcmd == "N"){
          //command is TURN_ON
          Serial.println("Checking AQI");
          if(AQI<=300){
```

```
            digitalWrite(green, HIGH);
          }else{
            digitalWrite(red, HIGH);
          }
        }

        if(nwcmd == "F"){
            //command is TURN_OFF
          Serial.println("Hi");
          digitalWrite(green, LOW);
          digitalWrite(red, LOW);
        }
      }
      else{
        Serial.println("  Nothing new.");
      }

    }
    else{
      Serial.println("Problem checking queue. HTTP error code " +
String(x));
    }
  }

  int t = digitalRead(flame);
   if (t==0) {
    Serial.println("FIRE!!");
    digitalWrite(buzz,HIGH);
    sendEmail();
    Serial.print("Mail sent to:");
    Serial.println(" The recipient");
    Serial.println("");
  }
  digitalWrite(buzz,LOW);
  delay(200);

}
```

In loop, we are first storing the data from the gas sensor through the analog pin to a variable and then uploading it to thinkspeak cloud platform. Then we are checking whether there are any commands given from the cloud platform to turn on/ turn off the led. If the command is to turn on then we are checking the range in which the current AQI lies and turn one the led's pin to high else if the command is to turn off the led then we are changing both the pins to low.

Then we are checking if the flame sensor is giving output high or low. If it is low then nothing is to be done ,else we trigger the buzzer by turning pin D1 to high and then run the send_email() function.

**Send email function and rest of the code:-**

```
byte sendEmail()
{
  if (Client.connect(server1, 2525) == 1)        // connect to smtp
server with port address 2525
  {
    Serial.println(F("connected to server"));
  }
  else
  {
    Serial.println(F("connection failed"));
    return 0;
  }
  if (!emailResp())          // if connection failed return now
    return 0;
  //
  Serial.println(F("Sending EHLO"));
  Client.println("EHLO paurastarle22@gmail.com");
  if (!emailResp())
    return 0;

  Serial.println(F("Sending auth login"));
  Client.println("AUTH LOGIN");
  if (!emailResp())
    return 0;

  Serial.println(F("Sending User"));
  Client.println("eWFzaDMwMTg="); //base64, ASCII encoded SMTP Username
  if (!emailResp())
    return 0;

  Serial.println(F("Sending Password"));
  Client.println("eWFzaGFnYXJ3YWxAMTIz");   //base64, ASCII encoded
SMTP Password
  if (!emailResp())
    return 0;

  Serial.println(F("Sending From"));
```

```
    Client.println(F("MAIL From: yashagarwal3018@gmail.com"));
  if (!emailResp())
    return 0;
  // change to recipient address
  Serial.println(F("Sending To"));
  Client.println(F("RCPT To: t.pauras@iitg.ac.in"));

  if (!emailResp())
    return 0;

  Serial.println(F("Sending DATA"));
  Client.println(F("DATA"));
  if (!emailResp())
    return 0;
  Serial.println(F("Sending email"));

  Client.println(F("To:  t.pauras@iitg.ac.in "));

  Client.println(F("From: yashagarwal3018@gmail.com "));
  Client.println(F("Subject: Fire Alarm\r\n"));
  Client.println(F("Attention: Fire Detected.\n"));
  Client.println(F("."));
  if (!emailResp())
    return 0;
  Serial.println(F("Sending QUIT"));
  Client.println(F("QUIT"));
  if (!emailResp())
    return 0;
  Client.stop();
  Serial.println(F("disconnected"));
  return 1;
}

int httpPOST(String uri, String postMessage, String &response){

  bool connectSuccess = false;
  connectSuccess = Client.connect("api.thingspeak.com",80);

  if(!connectSuccess){
      return -301;
  }

  postMessage += "&headers=false";
```

```
  String Headers =  String("POST ") + uri + String(" HTTP/1.1\r\n") +
                    String("Host: api.thingspeak.com\r\n") +
                    String("Content-Type:
application/x-www-form-urlencoded\r\n") +
                    String("Connection: close\r\n") +
                    String("Content-Length: ") +
String(postMessage.length()) +
                    String("\r\n\r\n");

  Client.print(Headers);
  Client.print(postMessage);

  long startWaitForResponseAt = millis();
  while(Client.available() == 0 && millis() - startWaitForResponseAt <
5000){
      delay(100);
  }

  if(Client.available() == 0){
    return -304; // Didn't get server response in time
  }

  if(!Client.find(const_cast<char *>("HTTP/1.1"))){
      return -303; // Couldn't parse response (didn't find HTTP/1.1)
  }

  int status = Client.parseInt();
  if(status != 200){
    return status;
  }

  if(!Client.find(const_cast<char *>("\n\r\n"))){
    return -303;
  }

  String tempString = String(Client.readString());
  response = tempString;

  return status;

}
```

```
byte emailResp()
{
  byte responseCode;
  byte readByte;
  int loopCount = 0;

  while (!Client.available())
  {
    delay(1);
    loopCount++;
    // Wait for 20 seconds and if nothing is received, stop.
    if (loopCount > 20000)
    {
      Client.stop();
      Serial.println(F("\r\nTimeout"));
      return 0;
    }
  }

  responseCode = Client.peek();
  while (Client.available())
  {
    readByte = Client.read();
    Serial.write(readByte);
  }

  if (responseCode >= '4')
  {
    //  efail();
    return 0;
  }
  return 1;
}
```

These functions send email through the SMTP2go cloud platform on behalf of a verified sender to the desired receiver.