**IoT BASED AIR POLLUTION/QUALITY MONITORING WITH ESP8266**

**INTRODUCTION:**

Air pollution is the biggest problem of every nation. As per 2012 survey; nearly 7 million people lost their life as consequence of various ailments such as bronchitis, obstructive pneumonia and other breathing problems. According to a survey, due to air pollution 50,000 to 100,000 premature deaths per year occur in the U.S. alone. Whereas in EU number reaches to 300,000 and over 3,000,000 worldwide. By monitoring the Air pollution level, we can prevent ourself from getting into a place with more pollution. This project is an IoT Based Air Pollution Monitoring System which monitors the Air quality over a web server using Internet and will trigger an alarm when the air quality goes down beyond a certain threshold level, means when there are sufficient number of harmful gases present in the air like CO2, smoke, alcohol, benzene, NH3, LPG and NOx.

**AIR QUALITY MEASUREMENT:**

In this project, we measure the quality/pollution level of the surrounding air using a Gas sensor. It measures the concentrations of major air pollutant gases in the surrounding. It will be connected to a microcontroller then it will sense all gases, and it will give the Pollution level. The microcontroller will process the data which it gets from the gas sensor and sends the information to Webserver. Where the quality of air can be shown and you can monitor the pollution level from anywhere in the world using your computer or your smartphone. This project will focus on development of the ThingSpeak an IoT platform that to show the data on the sensor.

**WORKING:**

The proposed air pollution monitoring system measures the quality/pollution level of the air by using MQ135 gas sensor. The MQ135 sensor is like a variable resistor, whose resistance depends on the gases surrounding it. To sense the gases, the MQ135 sensor contains a gas sensing material that is made up of SnO2. An electrode and electrode line of materials Au and Pt is present in the MQ135 sensor. A heater coil that is made up of Ni-Cr alloy, is used to provide the necessary working conditions for the sensor to work. The MQ135 sensor can sense NH3, NOx, alcohol, Benzene, smoke, CO2, SO2 etc. So, it is a dynamic gas sensor for our Air pollution Monitoring system. When it will be connected to Node MCU then it will sense all gases, and it will give the Pollution level. The Node MCU is connected to ThingSpeak, an IoT platform. The measured data about the pollution level is sent to the ThingSpeak server where the values can be seen from anywhere in the world. This proposed project can be used for many applications.

**CODING USED:**

#include <ESP8266WiFi.h>

#include <SFE\_BMP180.h>

#include <Wire.h>

String apiWritekey = "6Y2G86I96GAZEJBW";

const char\* ssid = "123456789";

const char\* password = "0987654321" ;

int mq135 = A0;

SFE\_BMP180 pressure;

#define ALTITUDE 1655.0 // Altitude of SparkFun's HQ in Boulder, CO. in meters

const char\* server = "api.thingspeak.com";

float resolution=3.3/1023;

WiFiClient client;

int data = 0;

void setup() {

  Serial.begin(115200);

  WiFi.disconnect();

  delay(10);

  WiFi.begin(ssid, password);

  Serial.println();

  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.print("NodeMcu connected to wifi...");

  Serial.println(ssid);

  Serial.println();

  if (pressure.begin())

    Serial.println("BMP180 init success");

  else

  {

    // Oops, something went wrong, this is usually a connection problem,

    // see the comments at the top of this sketch for the proper connections.

    Serial.println("BMP180 init fail\n\n");

    while(1); // Pause forever.

  }

}

void loop() {

   char status;

  double T,P,p0,a;

  // Loop here getting pressure readings every 10 seconds.

  // If you want sea-level-compensated pressure, as used in weather reports,

  // you will need to know the altitude at which your measurements are taken.

  // We're using a constant called ALTITUDE in this sketch:

  Serial.println();

  Serial.print("provided altitude: ");

  Serial.print(ALTITUDE,0);

  Serial.print(" meters, ");

  Serial.print(ALTITUDE\*3.28084,0);

  Serial.println(" feet");

  // If you want to measure altitude, and not pressure, you will instead need

  // to provide a known baseline pressure. This is shown at the end of the sketch.

  // You must first get a temperature measurement to perform a pressure reading.

  // Start a temperature measurement:

  // If request is successful, the number of ms to wait is returned.

  // If request is unsuccessful, 0 is returned.

  status = pressure.startTemperature();

  if (status != 0)

  {

    // Wait for the measurement to complete:

    delay(status);

    // Retrieve the completed temperature measurement:

    // Note that the measurement is stored in the variable T.

    // Function returns 1 if successful, 0 if failure.

    status = pressure.getTemperature(T);

    if (status != 0)

    {

      // Print out the measurement:

      Serial.print("temperature: ");

      Serial.print(T,2);

      Serial.print(" deg C, ");

      Serial.print((9.0/5.0)\*T+32.0,2);

      Serial.println(" deg F");

      // Start a pressure measurement:

      // The parameter is the oversampling setting, from 0 to 3 (highest res, longest wait).

      // If request is successful, the number of ms to wait is returned.

      // If request is unsuccessful, 0 is returned.

      status = pressure.startPressure(3);

      if (status != 0)

      {

        // Wait for the measurement to complete:

        delay(status);

        // Retrieve the completed pressure measurement:

        // Note that the measurement is stored in the variable P.

        // Note also that the function requires the previous temperature measurement (T).

        // (If temperature is stable, you can do one temperature measurement for a number of pressure measurements.)

        // Function returns 1 if successful, 0 if failure.

        status = pressure.getPressure(P,T);

        if (status != 0)

        {

          // Print out the measurement:

          Serial.print("absolute pressure: ");

          Serial.print(P,2);

          Serial.print(" mb, ");

          Serial.print(P\*0.0295333727,2);

          Serial.println(" inHg");

          // The pressure sensor returns abolute pressure, which varies with altitude.

          // To remove the effects of altitude, use the sealevel function and your current altitude.

          // This number is commonly used in weather reports.

          // Parameters: P = absolute pressure in mb, ALTITUDE = current altitude in m.

          // Result: p0 = sea-level compensated pressure in mb

          p0 = pressure.sealevel(P,ALTITUDE); // we're at 1655 meters (Boulder, CO)

          Serial.print("relative (sea-level) pressure: ");

          Serial.print(p0,2);

          Serial.print(" mb, ");

          Serial.print(p0\*0.0295333727,2);

          Serial.println(" inHg");

          // On the other hand, if you want to determine your altitude from the pressure reading,

          // use the altitude function along with a baseline pressure (sea-level or other).

          // Parameters: P = absolute pressure in mb, p0 = baseline pressure in mb.

          // Result: a = altitude in m.

          a = pressure.altitude(P,p0);

          Serial.print("computed altitude: ");

          Serial.print(a,0);

          Serial.print(" meters, ");

          Serial.print(a\*3.28084,0);

          Serial.println(" feet");

        }

        else Serial.println("error retrieving pressure measurement\n");

      }

      else Serial.println("error starting pressure measurement\n");

    }

    else Serial.println("error retrieving temperature measurement\n");

  }

  else Serial.println("error starting temperature measurement\n");

  data = analogRead(mq135);

  float temp = ((analogRead(A0) \* resolution) \* 100)+23.89;

  if (client.connect(server,80)) {

    String tsData = apiWritekey;

    tsData +="&field1=";

    tsData += String(P);

    tsData +="&field2=";

    tsData += String(data);

    tsData += "\r\n\r\n";

    client.print("POST /update HTTP/1.1\n");

    client.print("Host: api.thingspeak.com\n");

    client.print("Connection: close\n");

    client.print("X-THINGSPEAKAPIKEY: "+apiWritekey+"\n");

    client.print("Content-Type: application/x-www-form-urlencoded\n");

    client.print("Content-Length: ");

    client.print(tsData.length());

    client.print("\n\n");

    client.print(tsData);

    Serial.print("Pressure: ");

    Serial.print(P);

    Serial.print("Gas: ");

    Serial.print(data);

    Serial.println("uploaded to Thingspeak server....");

  }

  client.stop();

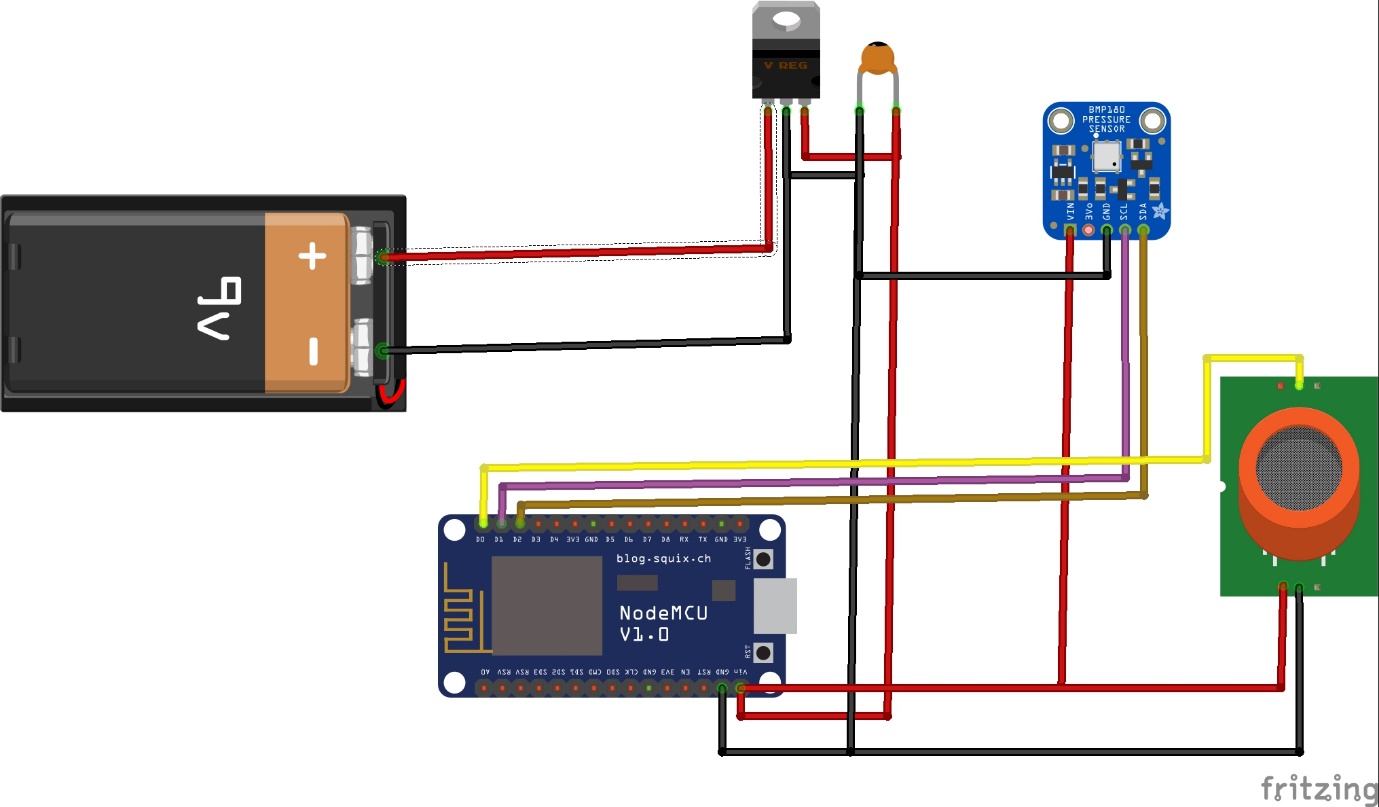
  Serial.println("Waiting to upload next reading...");

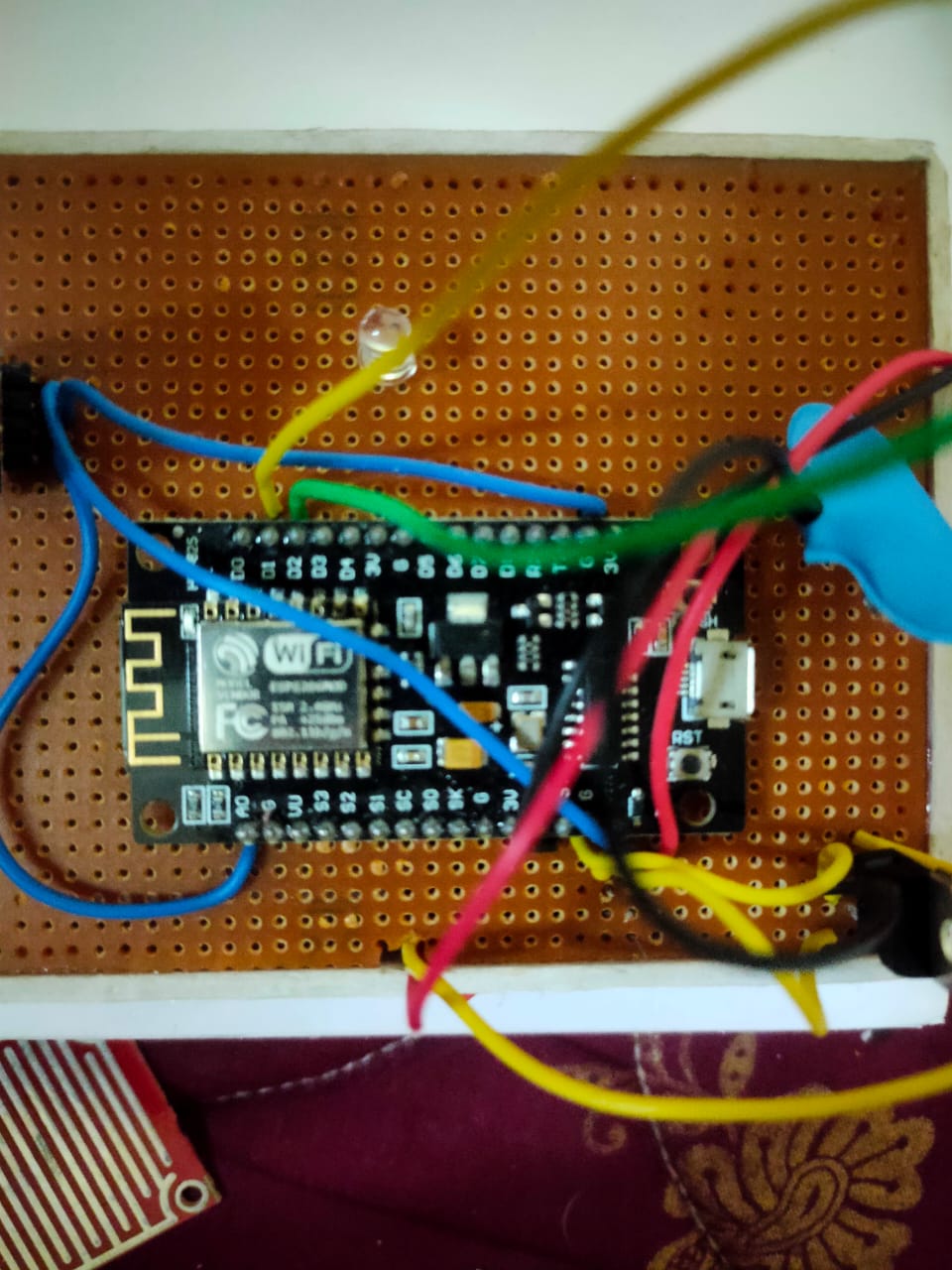
  Serial.println();

  delay(15000);

}

**CIRCUIT DIAGRAM:**



**PROJECT IMAGES:**

****

**RESULT:**

The proposed project can be used in various applications where monitoring air quality is necessary. One of its application can be IoT based pollution monitoring system in the exhaust of the vehicles. Using which the pollution level emitted from the vehicle.