AIR QUALITY MONITORING

- In this project we are going to make an **IoT Based Air Pollution Monitoring System** in which we will **monitor the Air Quality over a webserver using internet** and will trigger a alarm when the air quality goes down beyond a certain level, means when there are sufficient amount of harmful gases are present in the air like CO2, smoke, alcohol, benzene and NH3. It will show the air quality in PPM on the LCD and as well as on webpage so that we can monitor it very easily.

Required Components:

- MQ135 Gas sensor
- Arduino Uno
- Wi-Fi module ESP8266
- 16X2 LCD
- Breadboard
- 10K potentiometer
- 1K ohm resistors
- 220 ohm resistor
- Buzzer

Circuit Diagram and Explanation:

First of all we will connect the **ESP8266 with the Arduino**. ESP8266 runs on 3.3V and if you will give it 5V from the Arduino then it won't work properly and it may get damage. Connect the VCC and the CH_PD to the 3.3V pin of Arduino. The RX pin of ESP8266 works on 3.3V and it will not communicate with the Arduino when we will connect it directly to the Arduino. So, we will have to make a voltage divider for it which will convert the 5V into 3.3V. This can be done by connecting three resistors in series like we did in the circuit. Connect the TX pin of the ESP8266 to the pin 10 of the Arduino and the RX pin of the esp8266 to the pin 9 of Arduino through the resistors.

- Connect pin 1 (VEE) to the ground.
- Connect pin 2 (VDD or VCC) to the 5V.
- Connect pin 3 (V0) to the middle pin of the 10K potentiometer and connect the other two ends of the potentiometer to the VCC and the GND.

The potentiometer is used to control the screen contrast of the LCD. Potentiometer of values other than 10K will work too.

- Connect pin 4 (RS) to the pin 12 of the Arduino.
- Connect pin 5 (Read/Write) to the ground of Arduino. This pin is not often used so we will connect it to the ground.
- Connect pin 6 (E) to the pin 11 of the Arduino. The RS and E pin are the control pins which are used to send data and characters.
- The following four pins are data pins which are used to communicate with the Arduino.

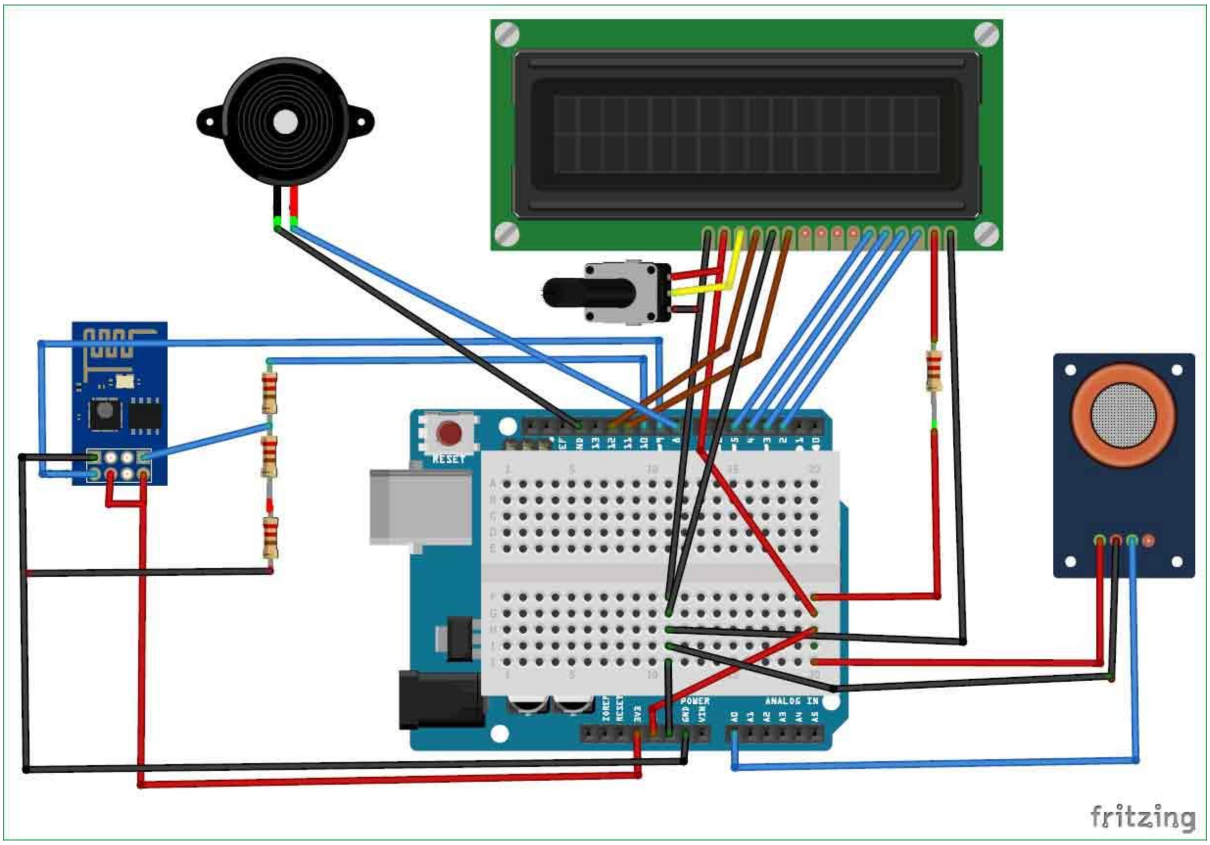
Connect pin 11 (D4) to pin 5 of Arduino.

Connect pin 12 (D5) to pin 4 of Arduino.

Connect pin 13 (D6) to pin 3 of Arduino.

Connect pin 14 (D7) to pin 2 of Arduino.

- Connect pin 15 to the VCC through the 220 ohm resistor. The resistor will be used to set the back light brightness. Larger values will make the back light much more darker.
- Connect pin 16 to the Ground.



Working Explanation:

The MQ135 sensor can sense NH3, NOx, alcohol, Benzene, smoke, CO2 and some other gases, so it is perfect gas sensor for our **Air Quality Monitoring Project**. When we will connect it to Arduino then it will sense the gases, and we will get the Pollution level in PPM (parts per million). MQ135 gas sensor gives the output in form of voltage levels and we need to convert it into PPM. So for converting the output in PPM, here we have used a library for MQ135 sensor, it is explained in detail in “Code Explanation” section below.

Sensor was giving us value of 90 when there was no gas near it and the safe level of air quality is 350 PPM and it should not exceed 1000 PPM. When it exceeds the limit of 1000 PPM, then it starts cause Headaches, sleepiness and stagnant, stale, stuffy air and if exceeds beyond 2000 PPM then it can cause increased heart rate and many other diseases.

When the value will be less than 1000 PPM, then the LCD and webpage will display “Fresh Air”. Whenever the value will increase 1000 PPM, then the buzzer will start beeping and the LCD and webpage will display “Poor Air, Open Windows”. If it will increase 2000 then the buzzer will keep beeping and the LCD and webpage will display “Danger! Move to fresh Air”.

Code Explanation:

Before beginning the coding for this project, we need to first Calibrate the MQ135 Gas sensor. There are lots of calculations involved in converting the output of sensor into PPM value, we have done this calculation before in our previous Smoke Detector project. But here we are using the Library for MQ135, you can download and install this MQ135 library from here: https://github.com/GeorgK/MQ135.

Using this library you can directly get the PPM values, by just using the below two lines:

```
MQ135 gasSensor = MQ135(A0);

float air_quality = gasSensor.getPPM();
```

But before that we need to **calibrate the MQ135 sensor**, for calibrating the sensor upload the below given code and let it run for 12 to 24 hours and then get the *RZERO* value.

```
#include "MQ135.h"
void setup (){
Serial.begin (9600);
}
void loop() {
MQ135 gasSensor = MQ135(A0); // Attach sensor to pin A0
float rzero = gasSensor.getRZero();
Serial.println (rzero);
delay(1000);
}
```

After getting the *RZERO* value. **Put the RZERO value in the library file** you downloaded "MQ135.h": *#define RZERO 494.63*

Now we can begin the actual code for our Air quality monitoring project.

In the code, first of all we have defined the libraries and the variables for the Gas sensor and the LCD. By using the Software Serial Library, we can make any digital pin as TX and RX pin. In this code, we have made Pin 9 as the RX pin and the pin 10 as the TX pin for the ESP8266. Then we have included the library for the LCD and have defined the pins for the same. We have also defined two more variables: one for the sensor analog pin and other for storing *air_quality* value.

```
#include <SoftwareSerial.h>
#define DEBUG true
SoftwareSerial esp8266(9,10);
#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11, 5, 4, 3, 2);
const int sensorPin= 0;
```

```
int air_quality;
```

Then we will declare the pin 8 as the output pin where we have connected the buzzer. l*cd.begin(16,2)* command will start the LCD to receive data and then we will set the cursor to first line and will print the *‘circuitdigest’*. Then we will set the cursor on the second line and will print *‘Sensor Warming’*.

```
pinMode(8, OUTPUT);
lcd.begin(16,2);
lcd.setCursor (0,0);
lcd.print ("circuitdigest ");
lcd.setCursor (0,1);
lcd.print ("Sensor Warming ");
delay(1000);
```

Then we will set the baud rate for the serial communication. Different ESP’s have different baud rates so write it according to your ESP’s baud rate. Then we will send the commands to set the ESP to communicate with the Arduino and show the IP address on the serial monitor.

```
Serial.begin(115200);
esp8266.begin(115200);
  sendData("AT+RST\r\n",2000,DEBUG);
  sendData("AT+CWMODE=2\r\n",1000,DEBUG);
  sendData("AT+CIFSR\r\n",1000,DEBUG);
  sendData("AT+CIPMUair_quality=1\r\n",1000,DEBUG);
  sendData("AT+CIPSERVER=1,80\r\n",1000,DEBUG);
pinMode(sensorPin, INPUT);
lcd.clear();
```

For printing the output on the webpage in web browser, we will have to use **HTML programming**. So, we have created a string named *webpage* and stored the output in it. We are subtracting 48 from the output because the *read()* function returns the ASCII decimal value and the first decimal number which is 0 starts at 48.

```
if(esp8266.available())
  {
    if(esp8266.find("+IPD,"))
    {
     delay(1000);
     int connectionId = esp8266.read()-48;
     String webpage = "<h1>IOT Air Pollution Monitoring System</h1>";
       webpage += "<p><h2>";
       webpage+= " Air Quality is ";
       webpage+= air_quality;
       webpage+=" PPM";
       webpage += "<p>";
```

The following code will call a function named *sendData* and will send the data & message strings to the webpage to show.

```
     sendData(cipSend,1000,DEBUG);
     sendData(webpage,1000,DEBUG);

     cipSend = "AT+CIPSEND=";
     cipSend += connectionId;
     cipSend += ",";
     cipSend +=webpage.length();
     cipSend +="\r\n";
```

The following code will print the data on the LCD. We have applied various conditions for checking air quality, and LCD will print the messages according to conditions and buzzer will also beep if the pollution goes beyond 1000 PPM.

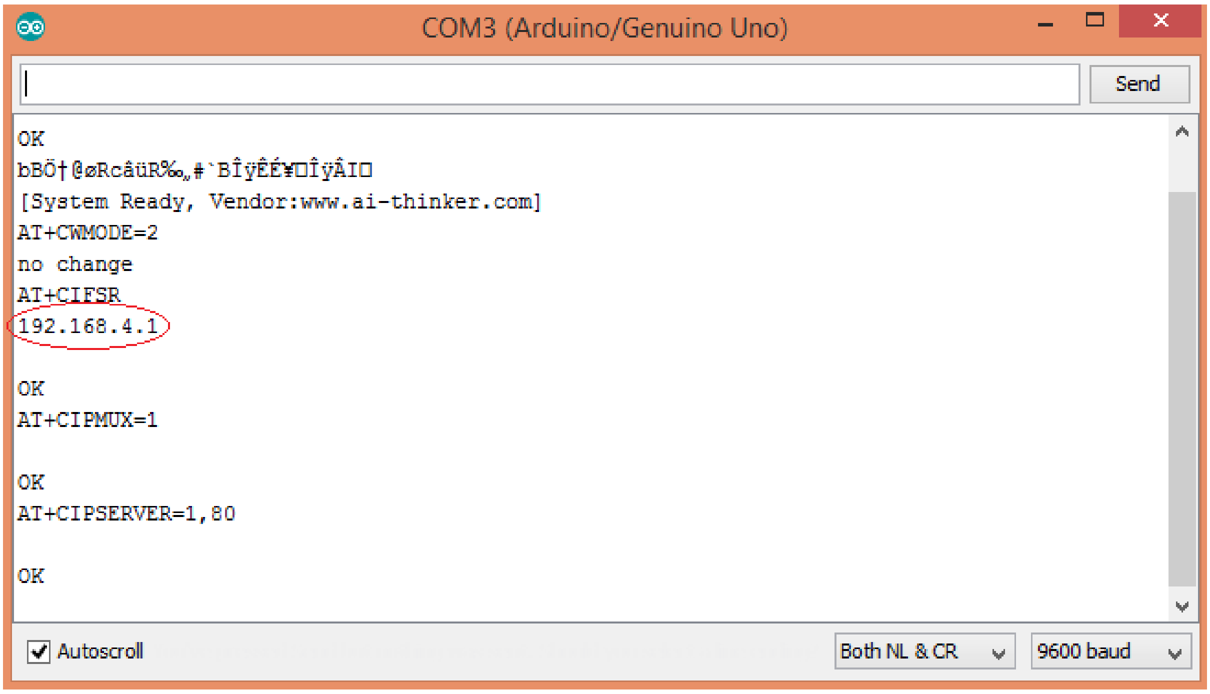
```
lcd.setCursor (0, 0);
lcd.print ("Air Quality is ");
lcd.print (air_quality);
lcd.print (" PPM ");
lcd.setCursor (0,1);
if (air_quality<=1000)
{
lcd.print("Fresh Air");
digitalWrite(8, LOW);
```

Finally the below function will send and show the data on the webpage. The data we stored in string named *'webpage'* will be saved in string named *'command'*. The ESP will then read the character one by one from the *'command'* and will print it on the webpage.

```
String sendData(String command, const int timeout, boolean debug)
{
    String response = "";
    esp8266.print(command); // send the read character to the esp8266
    long int time = millis();
    while( (time+timeout) > millis())
    {
      while(esp8266.available())
      {
        // The esp has data so display its output to the serial window
        char c = esp8266.read(); // read the next character.
        response+=c;
      }
    }
    if(debug)
    {
      Serial.print(response);
    }
    return response;
}
```

Testing and Output of the Project:

Before uploading the code, make sure that you are connected to the Wi-Fi of your ESP8266 device. After uploading, open the serial monitor and it will show the IP address like shown below.



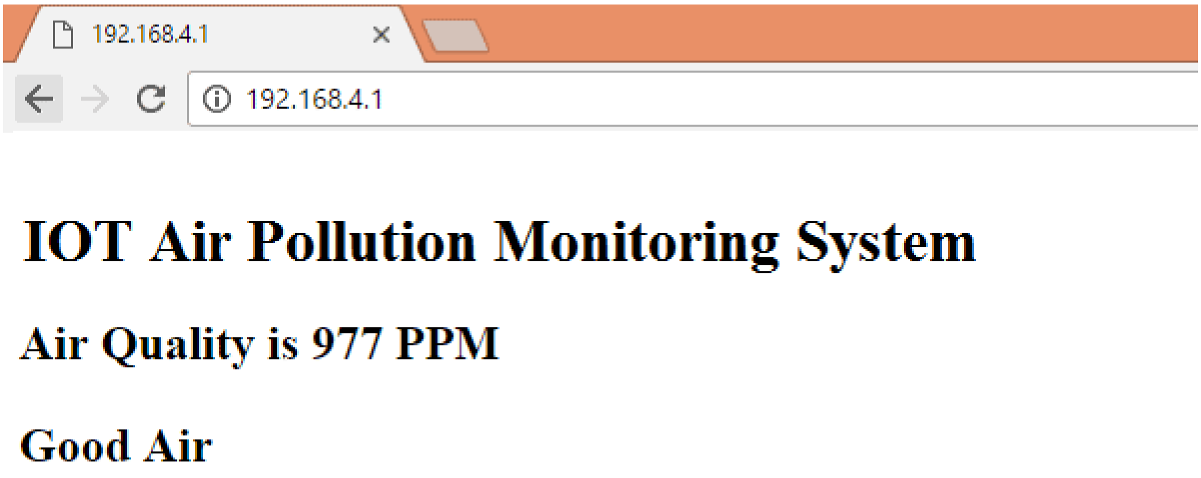
```
OK
bBÖ†@øRcâüR‰„#`BÎÿÊÉ¥□ÎÿÂI□
[System Ready, Vendor:www.ai-thinker.com]
AT+CWMODE=2
no change
AT+CIFSR
192.168.4.1

OK
AT+CIPMUX=1

OK
AT+CIPSERVER=1,80

OK
```

Type this IP address in your browser, it will show you the output as shown below. You will have to refresh the page again if you want to see the current Air Quality Value in PPM.



IOT Air Pollution Monitoring System

Air Quality is 977 PPM

Good Air

We have setup a local server to demonstrate its working, you can check the **Video** below. But to monitor the air quality from anywhere in the world, you need to **forward the port 80 (used for HTTP or internet) to your local or private IP address** (192.168*) of you device. After port forwarding all the incoming connections will be forwarded to this local address and you can open above shown webpage by just entering the public IP address of your internet from anywhere. You can forward the port by logging into your router (192.168.1.1) and find the option to setup the port forwarding.

Code

```
#include "MQ135.h"

#include <SoftwareSerial.h>

#define DEBUG true

SoftwareSerial esp8266(9,10); // This makes pin 9 of Arduino as RX pin and pin 10
of Arduino as the TX pin

const int sensorPin= 0;

int air_quality;

#include <LiquidCrystal.h>

LiquidCrystal lcd(12,11, 5, 4, 3, 2);
```

```
void setup() {

pinMode(8, OUTPUT);

lcd.begin(16,2);

lcd.setCursor (0,0);

lcd.print ("circuitdigest ");

lcd.setCursor (0,1);

lcd.print ("Sensor Warming ");

delay(1000);

Serial.begin(115200);

esp8266.begin(115200); // your esp's baud rate might be different

  sendData("AT+RST\r\n",2000,DEBUG); // reset module

  sendData("AT+CWMODE=2\r\n",1000,DEBUG); // configure as access point

  sendData("AT+CIFSR\r\n",1000,DEBUG); // get ip address

  sendData("AT+CIPMUair_quality=1\r\n",1000,DEBUG); // configure for multiple connections

  sendData("AT+CIPSERVER=1,80\r\n",1000,DEBUG); // turn on server on port 80

pinMode(sensorPin, INPUT);        //Gas sensor will be an input to the arduino

lcd.clear();

}

void loop() {

MQ135 gasSensor = MQ135(A0);

float air_quality = gasSensor.getPPM();

if(esp8266.available()) // check if the esp is sending a message

  {

    if(esp8266.find("+IPD,"))

    {

     delay(1000);

     int connectionId = esp8266.read()-48; /* We are subtracting 48 from the output because the read() function returns the ASCII decimal value and the first decimal number which is 0 starts at 48*/
     String webpage = "<h1>IOT Air Pollution Monitoring System</h1>";
       webpage += "<p><h2>";
       webpage+= " Air Quality is ";
       webpage+= air_quality;
       webpage+=" PPM";
       webpage += "<p>";
     if (air_quality<=1000)
{
  webpage+= "Fresh Air";
}
else if(air_quality<=2000 && air_quality>=1000)
{
  webpage+= "Poor Air";
}
else if (air_quality>=2000 )
{
webpage+= "Danger! Move to Fresh Air";
}
webpage += "</h2></p></body>";
     String cipSend = "AT+CIPSEND=";
     cipSend += connectionId;
     cipSend += ",";
     cipSend +=webpage.length();
     cipSend +="\r\n";

     sendData(cipSend,1000,DEBUG);
     sendData(webpage,1000,DEBUG);

     cipSend = "AT+CIPSEND=";
     cipSend += connectionId;

     cipSend += ",";

     cipSend +=webpage.length();

     cipSend +="\r\n";


     String closeCommand = "AT+CIPCLOSE=";

     closeCommand+=connectionId; // append connection id

     closeCommand+="\r\n";


     sendData(closeCommand,3000,DEBUG);

    }

  }

lcd.setCursor (0, 0);

lcd.print ("Air Quality is ");

lcd.print (air_quality);

lcd.print (" PPM ");

lcd.setCursor (0,1);

if (air_quality<=1000)

{

lcd.print("Fresh Air");

digitalWrite(8, LOW);

}

else if( air_quality>=1000 && air_quality<=2000 )

{

lcd.print("Poor Air, Open Windows");

digitalWrite(8, HIGH );

}

else if (air_quality>=2000 )

{
```

```
lcd.print("Danger! Move to Fresh Air");
digitalWrite(8, HIGH);   // turn the LED on
}
lcd.scrollDisplayLeft();
delay(1000);
}
String sendData(String command, const int timeout, boolean debug)
{
    String response = "";
    esp8266.print(command); // send the read character to the esp8266
    long int time = millis();
    while( (time+timeout) > millis())
    {
      while(esp8266.available())
      {
        // The esp has data so display its output to the serial window
        char c = esp8266.read(); // read the next character.
        response+=c;
      }
    }
    if(debug)
    {
      Serial.print(response);
    }
    return response;
}
```