Project Report

Version <1.0>

Air Quality Monitor using Arduino

Project Guide: Dr. Kamlesh Sharma, Associate Professor, CSE, FET
Ms. Neha Batra, Assistant Professor, CSE, FET

Members: Aniket
1/17/FET/BCC/045

College Name: Manav Rachna International Institute of Research and Studies (MRIIRS)

Department: Faculty of Engineering and Technology (FET)

Table of Contents

1. Introduction

1.1 Purpose

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 World Health Organization estimates that 4.6 million people die each year from causes directly attributable to air pollution. Even being aware of air particulates and toxins wage a difficult battle many people often don't realize that their towns, let alone their own homes, could be filled with tainted air.

However, today in the age of powerful IoT technologies it is possible that any person can combat bad air quality. Sensors and microcontrollers have gotten so small, and so simple to implement, that practically anyone can install any type of monitor or sensor in their very own home.

1.2 Existing system

Keeping the effects of air pollution in mind such as respiratory diseases, declining mortality rate and rise in the earth's temperature many systems and facilities have been developed to monitor the status of air pollution.

The air quality in current time is monitored either through the domestic air monitoring systems or through data provided by meteorological agencies collected by weather stations in the form of news feeds on TV, internet or radio broadcasts.

Most of the people still rely to the latter way to monitor their surrounding air because of the big size and high cost of domestic air purifying and monitoring devices. The problem regarding air quality monitoring is not with the accuracy of data provided by the weather agencies. The actual problems encountered are: -

The data provided is instantaneously after a regular interval say an hour or a few and is not real-time.

The data is collected through limited monitoring stations located at places such as public parks or bus stands. So, the data which you view may belong to your city, but it may not belong to your locality.

1.3 Proposed system

There is a strong need to monitor air quality in real-time whether in home or office. Our proposed system offers a light weight, compact and low-cost device that monitors the quality of air in real-time and sends the status to android device.

The proposed system is an IOT based device that collects the data through various sensors and processes them using an Arduino microcontroller board to determine the quality of air and then send the results to mobile via networking module via internet.

1.4 Hardware/Software requirement

Hardware Requirement:

- Arduino Uno Development Board
- Breadboard
- Jumper Wires
- DHT11 (Temperature and Humidity Sensor)
- MQ135 (Gas Sensor)
- Power Supply
- ESP8266 (Wi-Fi Module)

Software Requirement:

- Arduino IDE
- Text Editor
- Android Studio

1.5 Project scope

- Develop a low cost and energy efficient device to monitor air quality.
- Develop a compact and light weight air quality monitoring device that fits regular households and offices.
- To accurately monitor air quality of surrounding environment.
- Provide real-time status to the user on mobile device using IO.

1.6 References

1. Md. Abdullah Al Ahasan, Saumendu Roy, A. H. M. Saim, Rozina Akter, Md. Zakir Hossain. “Arduino – Based Real Time Air Quality and Pollution Monitoring System”, 2018.
2. Rajkumar, M. Newlin, M. S. Shruthi and V. Venkatesa Kumar. “IOT Based Smart System forControlling CO_2 Emission”, 2017.
3. World Health Organization. “Air Pollution”, World Health Organization,2020.
4. Ashish M. Husain 1 , Tazrin Hassan Rini * 2 , Mohammed Ikramul Haque 3 and Md. Rakibul Alam on Air Quality Monitoring: The Use of Arduino and Android, 2016
5. Teddy Surya Gunawan1 , Yasmin Mahira Saiful Munir2 , Mira Kartiwi3 , Hasmah Mansor4 on Design and Implementation of Portable Outdoor Air Quality Measurement System using Arduino, 2013.
6. Vikhyat Chaudhry on Arduair: Air Quality Monitoring, 2013.
7. Sherin Abraham, Xinrong Li on A Cost-Effective Wireless Sensor Network System for Indoor Air Quality Monitoring Applications, 2014.
8. Liu, S., Xia, C., & Zhao, Z. (2016, October). A low-power real-time air quality monitoring system using lpwan based on lora. In 2016 13th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT) (pp. 379-381).IEEE
9. Desai, N. S., & Alex, J. S. R. (2017, March). IoT based air pollution monitoring and predictor system on Beagle bone black. In 2017 Interna-tional Conference on Nextgen Electronic Technologies: Silicon to Software (ICNETS2) (pp. 367-370).IEEE.
10. Balestrini, M., Diez, T., Marshall, P., Gluhak, A., & Rogers, Y. (2015). IoT community technologies: leaving users to their own devices or orchestration of engagement?. EAI Endorsed Transactions on Internet of Things, 1.
11. C.V. Saikumar, M. Reji, P.C. &KishorerajaIOT Based Air Quality Monitoring System International Journal of Pure and Applied Mathematics, 117 (9) (2017)
12. P. Kumar, C. Martani, L. Morawska, L. Norford, R. Choudhary, M. Bell, M. Leac hIndoor air quality and energy management through real-time sensing in commercial buildings Energy and Buildings, 111 (2016).
13. J. Kang, K.I. HwangAcomprehensivereal-timeindoorair-qualitylevelindicator Sustainability, 8 (9) (2016).

2. Overall Description

2.1 Air Quality Monitor using Arduino

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.

Previously we have built the LPG detector using MQ6 sensor and Smoke detector using MQ2 sensor but this time we have used MQ135 sensor which is the best choice for monitoring Air Quality as it can detects most harmful gases and can measure their amount accurately. In this IOT project, you can monitor the pollution level from anywhere using your computer or mobile. We can install this system anywhere and can also trigger some device when pollution goes beyond some level, like we can switch on the Exhaust fan or can send alert SMS/mail to the user.

2.2 Required Components

Hardware Requirement:

- Arduino Uno Development Board
- Breadboard
- Jumper Wires
- DHT11 (Temperature and Humidity Sensor)
- MQ135 (Gas Sensor)
- Power Supply
- ESP8266 (Wi-Fi Module)

Software Requirement:

- Arduino IDE
- Text Editor
- Android Studio

2.3 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.

ESP8266 Wi-Fi module gives your projects access to Wi-Fi or internet. It is a very cheap device and make your projects very powerful. It can communicate with any microcontroller and it is the most leading devices in the IOT platform. Learn more about using ESP8266 with Arduino here.

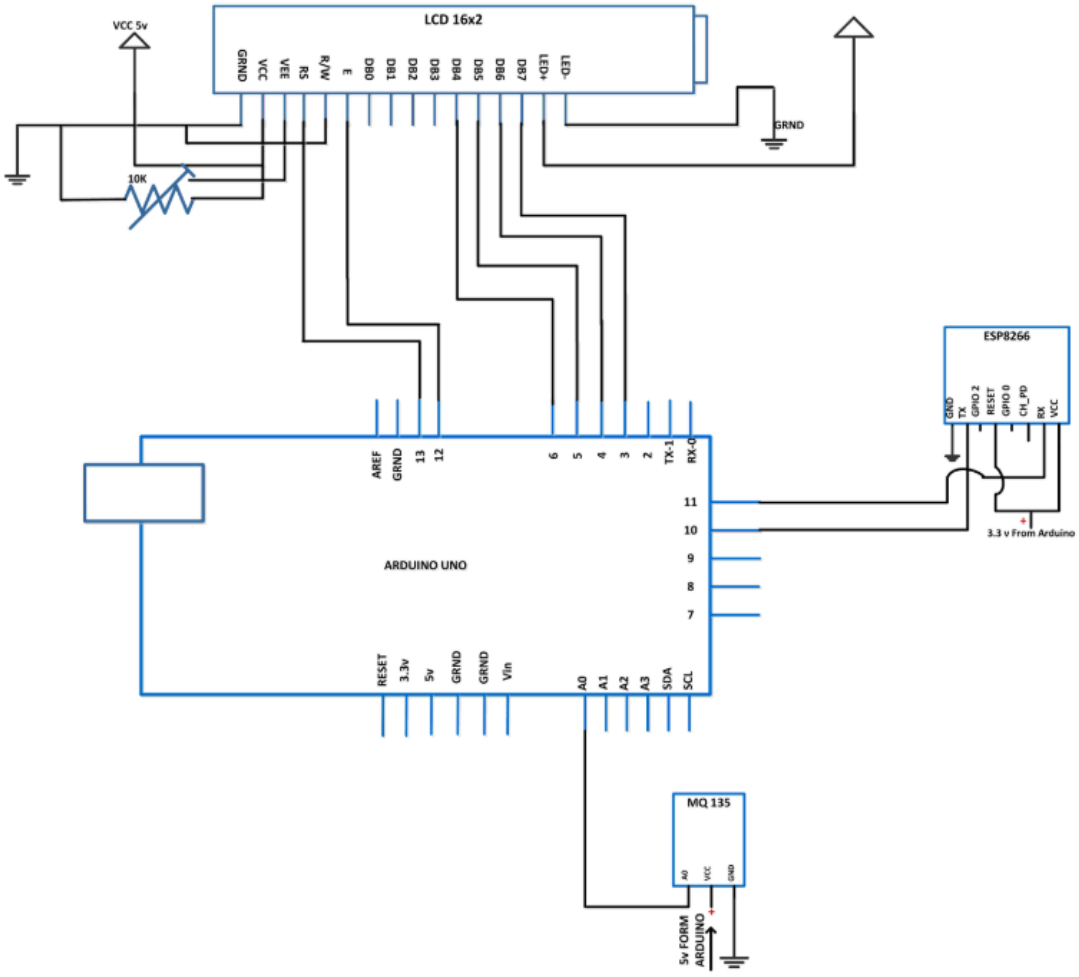

Figure.1: Archietecture of Air Pollution Monitor Arduino

Then we will connect the MQ135 sensor with the Arduino. Connect the VCC and the ground pin of the sensor to the 5V and ground of the Arduino and the Analog pin of sensor to the A0 of the Arduino.

Connect a buzzer to the pin 8 of the Arduino which will start to beep when the condition becomes true.

In last, we will connect LCD with the Arduino. The connections of the LCD are as follows

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.

3. Specific Requirements

13.1 Use Case Reports

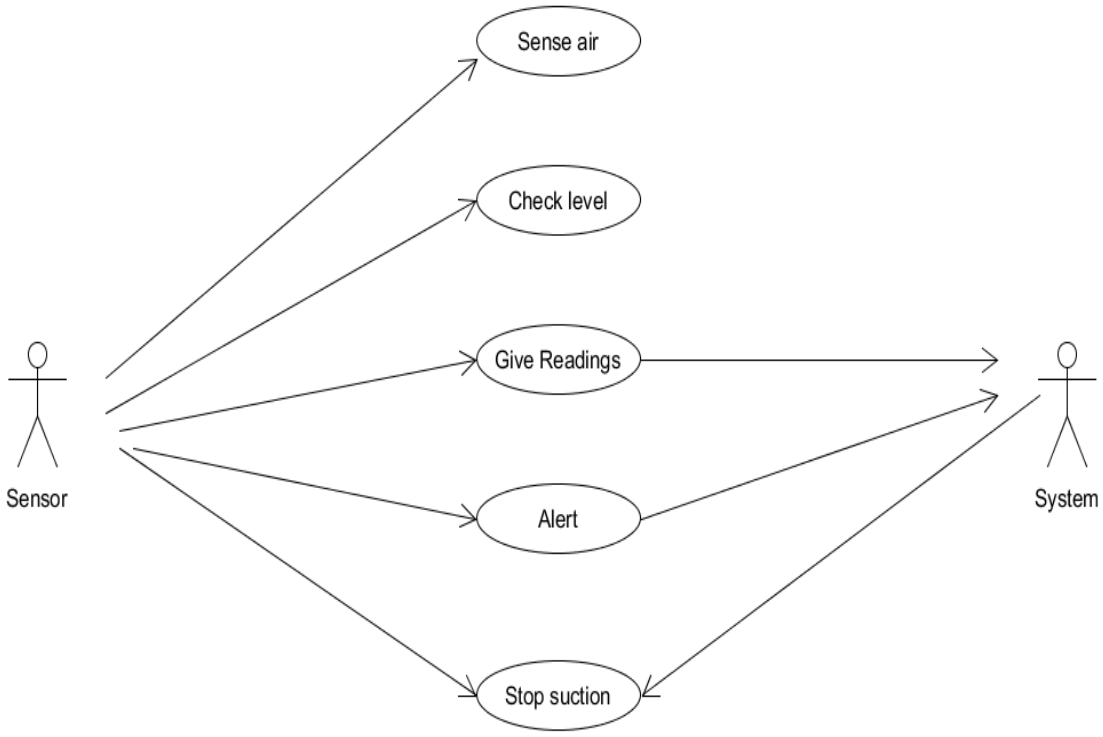

Figure.2: Use Case Diagram of Air Quality Working by System

Actors :

a. Sensor : It senses the air quality MQ135 sensor.
b. System: It is our arduino system in which data is forwarded.

Use Cases:

a. Sense Air: It will sense the current level of the air.
b. Check level: It will check the level of dust particles.
c. Give Readings: then according to that give readings to the arduino board to the system.
d. Alert: The alert use case is for the alerting to the system.
e. Stop suction: After the reading the sensor stops the suction of the dust particles.

Working:

At first, the system starts then the Sensor starts and take the appropriate readings and send it to the Arduino Uno system. Then the readings are stored and the suction becomes stop and readings are stored in the system database.

3.2 Activity Diagrams

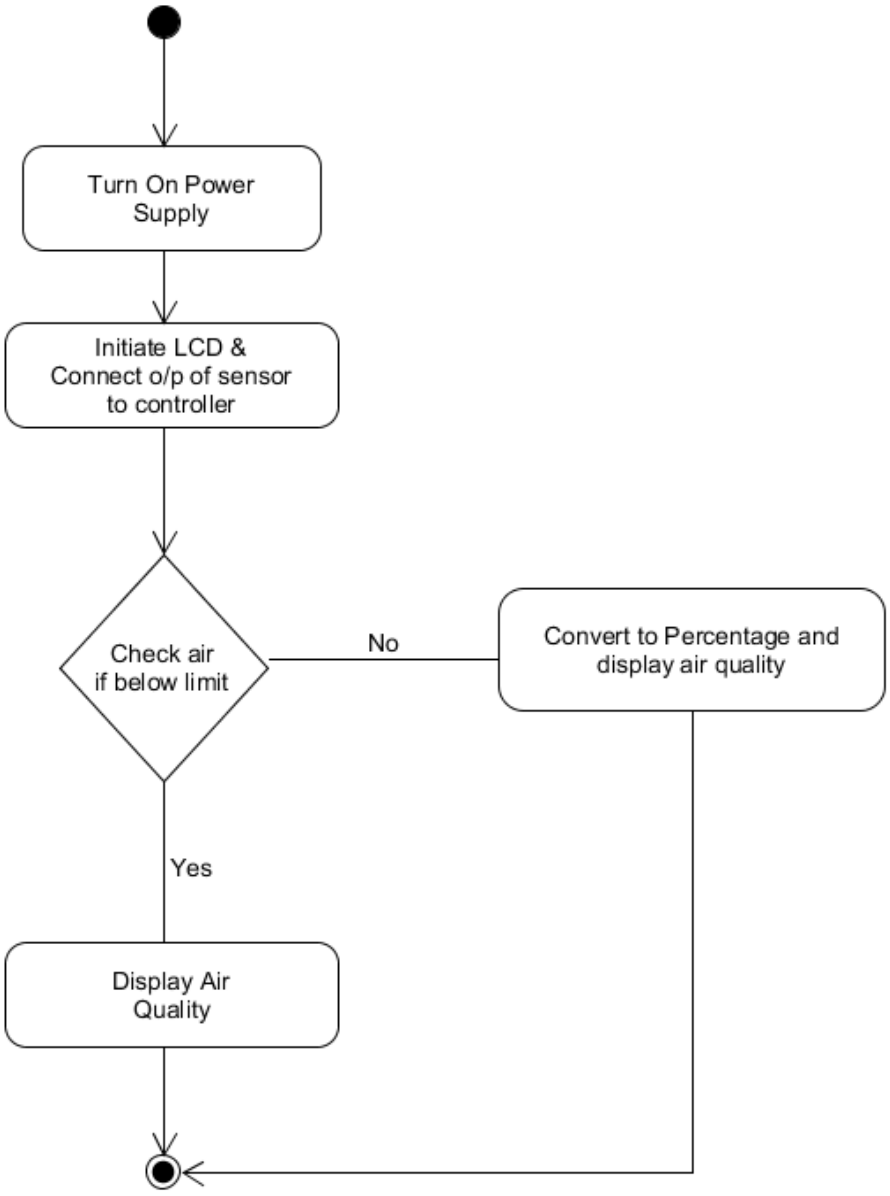

Figure.3: Activity Diagram for Air Quality Monitoring System

Components:

a. Start : It will start the execution.

b. Activities: Turn On power supply, Initiate LCD and Connect output of sensor to controller, convert to percentage and display air quality, display air quality.

c. Decision box: It will check if condition is true or false and return to the function.

d. End: It will end/terminate the activities with and output.

Working:

At first the execution starts from start and then user turns on the power supply to start the machine after that the Lcd light initiate on and off with respect to series of events and connect the output of the sensor to the controller.

The decision box the checks the air quality and if its in the limit the show the ouput to the user and end the activity if not, then it will calculate the percentage according to the ppm and display information to user and ends the activity.

3.3 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.

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 “Good 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”.

3.4 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. lcd.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;
}
```

3.5 Testing and Output

Before uploading the code, make sure that you should 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.

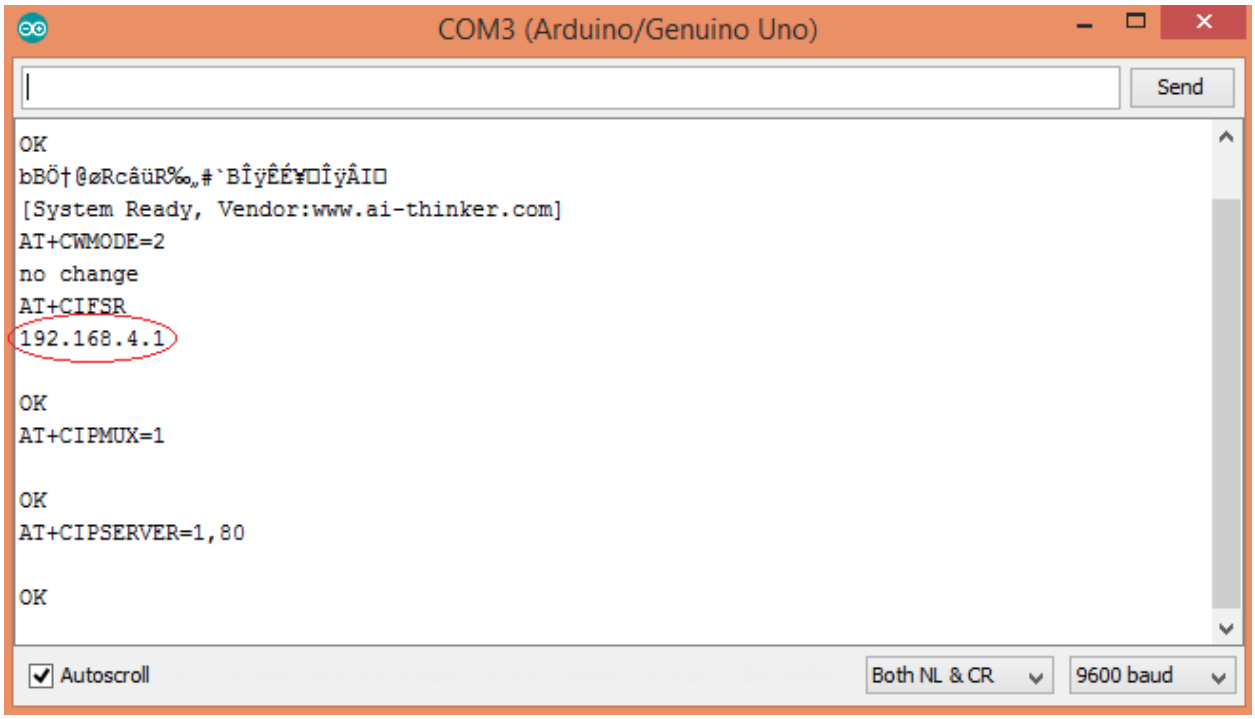

Figure.4: To show the Output on a Webpage

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.

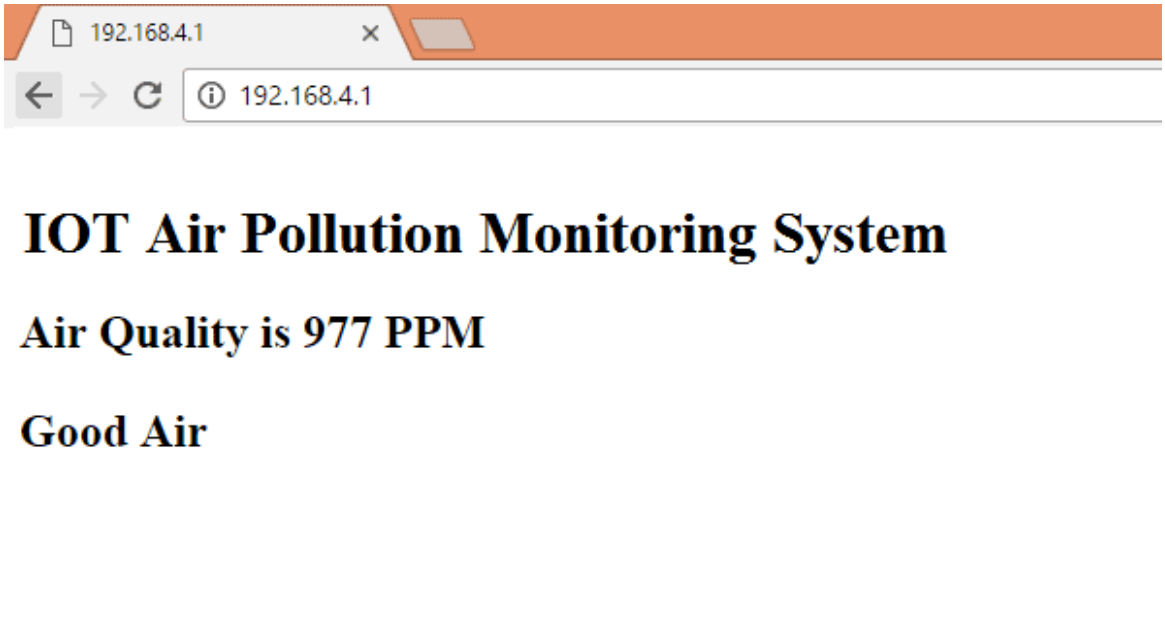

Figure.5: Output shown in the figure

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

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;
}
```