Documentation

- Describe the project's objectives, IoT device setup, platform development, and code implementation
- Include diagrams, schematics, and screenshots of the IoT devices and data-sharing platform
- Explain the project in detail

Project objectives:

Air quality monitoring refers to the collection and measurement of ambient air pollution samples. The data from these samples is compared to clean air standards and historical information regarding air quality levels, along with data reflecting its health and environmental impacts, to determine the state of the air.

What is the main objective of air quality monitoring?

Air Quality Monitoring Networks allow the measurement, operation and predictive analysis of the evolution of air pollution in different areas (urban areas, industrial areas, special nature conservation areas, etc.) Some stations are equipped with meteorological sensors and/or noise level meters to measure noise levels

What are the objectives of air quality?

Air quality objectives are used to guide decisions unless they are written specifically into a permit or regulation at which point they become binding requirements. They are typically used to: Assess current or historical air quality.

IOT DEVICE SETUP:

STEP 1 : The VCC pin of the MQ-135 gas sensor module was connected with the VU pin of Node MCU.

STEP 2 : The GND pin of the MQ-135 gas sensor module was connected with the GND pin of Node MCU.

STEP 3 : The Node MCU is powered with a 12V DC via AC-DC adapter for a day.

STEP 4 : The setup was then disconnected.

Air pollution is one of the biggest threats to the present-day environment. Everyone is being affected by air pollution day by day including humans, animals, crops, cities, forests and aquatic ecosystems. Besides that, it should be controlled at a certain level to prevent the increasing rate of global warming. This project aims to design an IOT-based air pollution monitoring system using the internet from anywhere using a computer or mobile to monitor the air quality of the surroundings and environment. There are various methods and instruments available for the measurement and monitoring quality of air.

The IoT-based air pollution monitoring system would not only help us to monitor the air quality but also be able to send alert signals whenever the air quality deteriorates and goes down beyond a certain level. In this system, Node MCU plays the main controlling role. It has been programmed in a manner, such that, it senses the sensory signals from the sensors and shows the quality level via led indicators. Besides the harmful gases (such as CO2, CO, smoke, etc) temperature and humidity can be monitored through the temperature and humidity sensor by this system. Sensor responses are fed to the Node MCU which displays the monitored data in the Thing Speak cloud which can be utilized for analysing the air quality of that area .The following simple flow diagram (as shown in Fig. 1) indicates the working mechanism of the IoT-based Air Pollution Monitoring System.

PLATFORM DEVELOPMENT:

The popularization of IoT devices and the emergence of low-cost sensors that make it possible to accurately measure the concentration of gases and pollutants in the atmosphere are attractive tools to fill the scarcity of information about the quality of the air that people breathe. In this context, the work presented here seeks to contribute to the control of the countless environmental and health impacts of people caused by pollution, because without data that prove the problem, it is difficult for the environmental agencies to act, which may even make it impossible to take actions or impose of limits.

The monitoring solution under development could become a tool for the prevention and control of the disease, as it will make it possible to provide real-time information on air quality that can be used to make decisions that must be taken to minimize impacts.

CODE IMPLEMENTATION:

SOFTWARE CODE for Calibration of MQ135 Sensor:

```
void setup ()

{

Serial begin (9600); //Baud rate
```

```
Pin Mode (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 sensor Value=0.0; //Define variable for analog readings

Serial print ("Sensor Reading = ");

Serial print ln (analog Read(A0));

For (int x = 0 : x < 500 ; x++) //Start for loop

 {

Sensor Value = sensor Value + analog Read(A0); //Add analog values of sensor 500 times
```

```
 }
Sensor Value = sensor Value/500.0; //Take average of readings
Sensor volt = sensor Value*(5.0/1023.0); //Convert average to voltage
RS air = ((5.0*1.0)/sensor volt)-1.0; //Calculate RS in fresh air
 R0 = RS air/3.7; //Calculate R0
Serial print ("R0 = "); //Display "R0"
Serial print ln(R0); //Display value of R0
Delay (1000); //Wait 1 second
}
```

4.3 Execution of the Main Program

```
#include < ESP8266WiFi.h >
#include < DHT h >
#include < Thing Speak h >
DHT dht (D5, DHT11);
#define LED_GREEN D2
#define LED_YELLOW D3
#define LED_RED D4
#define MQ_135 A0
int ppm=0;
float m = -0.3376; //Slope
float b = 0.7165; //Y-Intercept
float R0 = 3.12; //Sensor Resistance in fresh air from previous code
WiFi   Client
long my Channel   Number = 123456; // Channel id
```

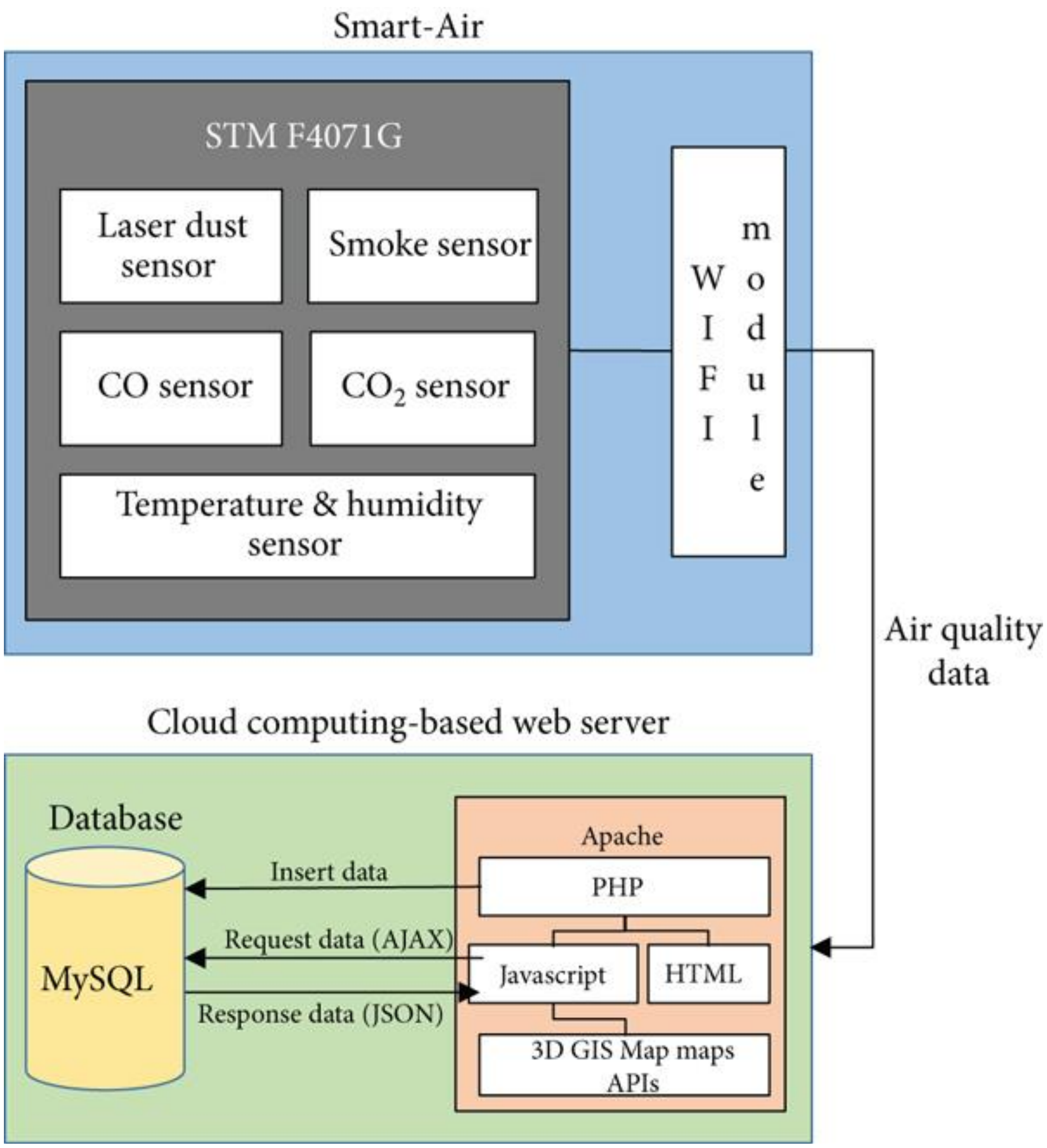
```
 char my Write API Key [] = "API   Key";
void setup () {
 // put your setup code here, to run once:
Serial begin (9600);
Pin Mode (LED GREEN OUTPUT);
Pin Mode (LED YELLOW OUTPUT);
```

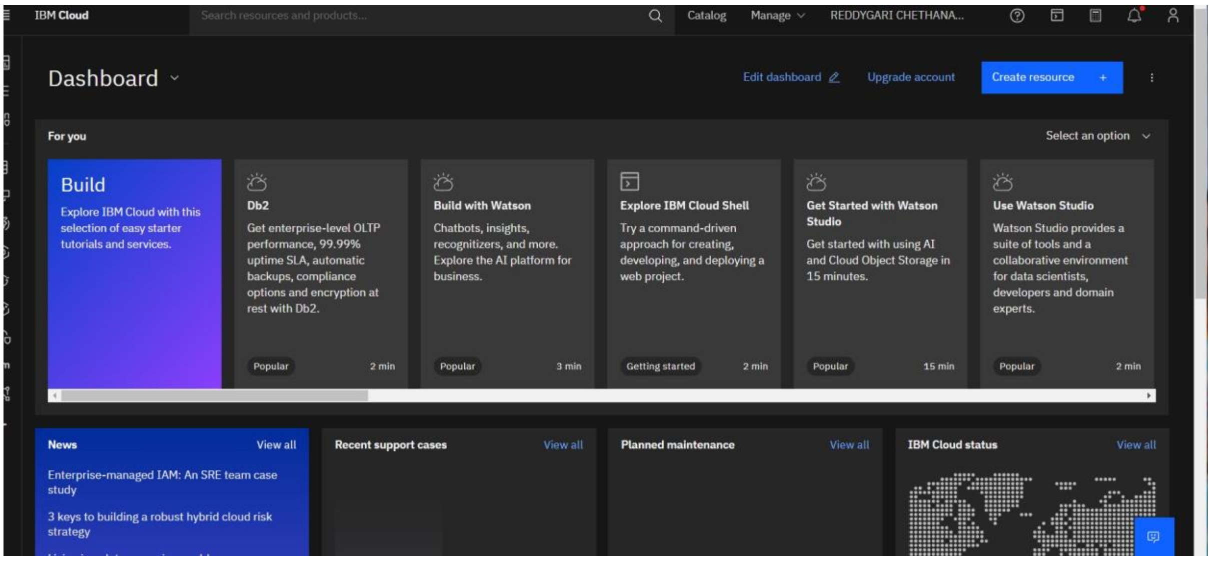
```
pin Mode (LED RED OUTPUT);
pin Mode (MQ_135, INPUT);
WiFi begin (" WiFi Name", " WiFi Password");
While (WiFi status () != WL_CONNECTED)
 {
Delay (200);
Serial print (".");
 }
Serial print ln ();
Serial print ln ("Node MCU is connected!");
Serial print ln (WiFi local IP ());
Begin ();
Thing Speak begin(client);
}
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 sensor Value;//Variable to store the analog values from MQ-135
 float h;
 float t;
 float ppm log; //Get ppm value in linear scale according to the ratio value
 float ppm; //Convert ppm value to log scale
 h = read Humidity ();
delay (4000);
 t = read Temperature ();
delay (4000);
sensor Value = analog Read (gas sensor); //Read analog values of sensor
sensor volt = sensor Value*(5.0/1023.0); //Convert analog values to voltage
RS gas = ((5.0*1.0)/sensor volt)-1.0; //Get value of RS in a gas
 ratio = RS gas/R0; // Get ratio RS gas/RS air
```

```
ppm log = (log10(ratio)-b)/m; //Get ppm value in linear scale according to the ratio value
 ppm = pow (10, ppm log); //Convert ppm value to log scale
Serial print ln ("Temperature: " + (String) t);
Serial print ln ("Humidity: " + (String) h);
```

```
Serial print ln ("Our desired PPM = "+ (String) ppm);
Thing Speak write Field (my Channel Number, 1, t, my Write API Key);
Delay (20000);
Thing Speak write Field (my Channel Number, 2, h, my Write API Key);
Delay (20000);
Thing Speak write Field (my Channel Number, 3, ppm, my Write API Key);
Delay (20000);
 if(ppm<=100)
 {
Digital Write (LED GREEN HIGH);
Digital Write (LED YELLOW LOW);
Digital Write (LED RED LOW);
 }
 else if(ppm<=200)
 {
Digital Write (LED GREEN LOW);
Digital Write (LED YELLOW HIGH);
Digital Write (LED RED LOW);
 }
 else
 {
Digital Write (LED GREEN LOW);
Digital Write (LED YELLO LOW);
Digital Write (LED RED HIGH);
 }
Delay (2000);}
```

DIAGRAMS:


Smart-Air
STM F4071G
Laser dust sensor
Smoke sensor
CO sensor
CO_2 sensor
Temperature & humidity sensor
WIFI module
Air quality data
Cloud computing-based web server
Database
MySQL
Insert data
Request data (AJAX)
Response data (JSON)
Apache
PHP
Javascript
HTML
3D GIS Map maps APIs



Build
Explore IBM Cloud with this selection of easy starter tutorials and services.

Db2
Get enterprise-level OLTP performance, 99.99% uptime SLA, automatic backups, compliance options and encryption at rest with Db2.

Build with Watson
Chatbots, insights, recognitizers, and more. Explore the AI platform for business.

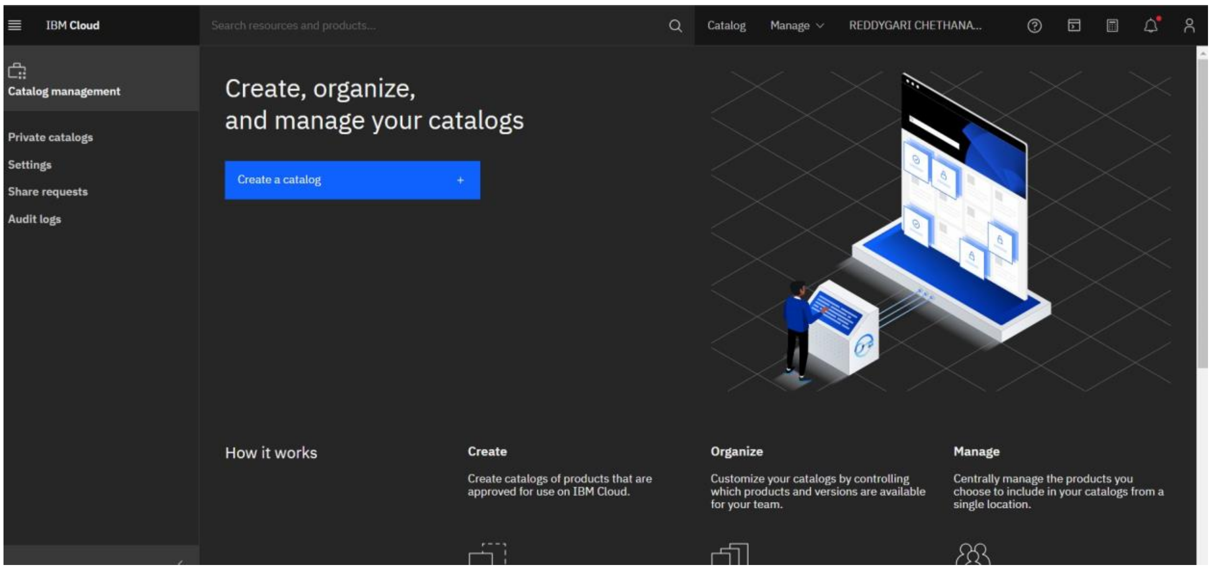
Explore IBM Cloud Shell
Try a command-driven approach for creating, developing, and deploying a web project.

Get Started with Watson Studio
Get started with using AI and Cloud Object Storage in 15 minutes.

Use Watson Studio
Watson Studio provides a suite of tools and a collaborative environment for data scientists, developers and domain experts.

Enterprise-managed IAM: An SRE team case study

3 keys to building a robust hybrid cloud risk strategy



Create, organize, and manage your catalogs

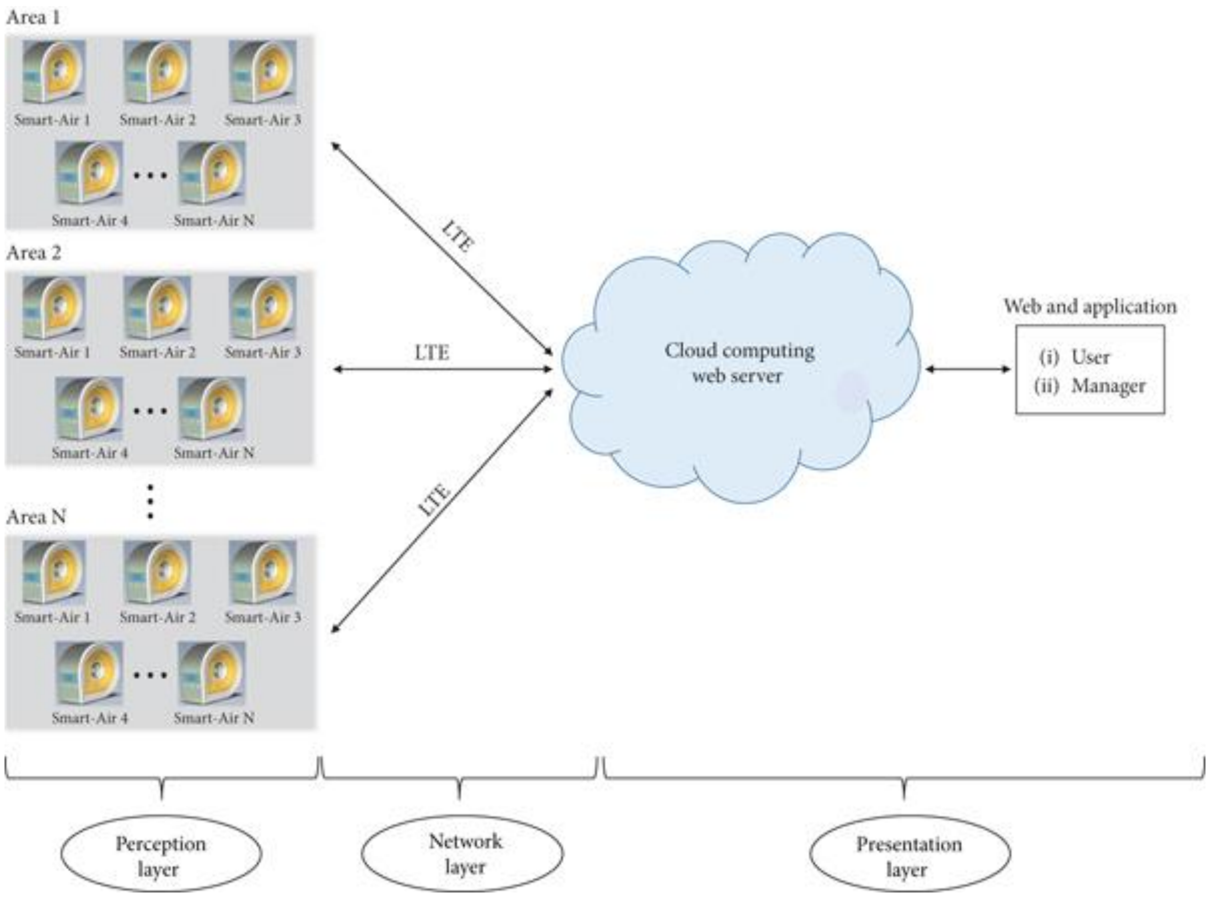
How it works

Create
Create catalogs of products that are approved for use on IBM Cloud.

Organize
Customize your catalogs by controlling which products and versions are available for your team.

Manage
Centrally manage the products you choose to include in your catalogs from a single location.

When a monitoring area has been determined, the specific types of air pollutants present must be considered. As mentioned above, Smart-Air has an expandable interface such that multiple sensors can be added to the microcontroller. Furthermore, the platform can monitor a large area or many areas simultaneously using multiple Smart-Air devices. Then, each device is classified by area to visualize the data. Each Smart-Air device transmits air quality data to the web server via LTE and automatically indicates the air quality for the specific area by LED color. Moreover, each device can be set to present a unique color of LED through the application or web server



PROJECT IN DETAIL:

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.

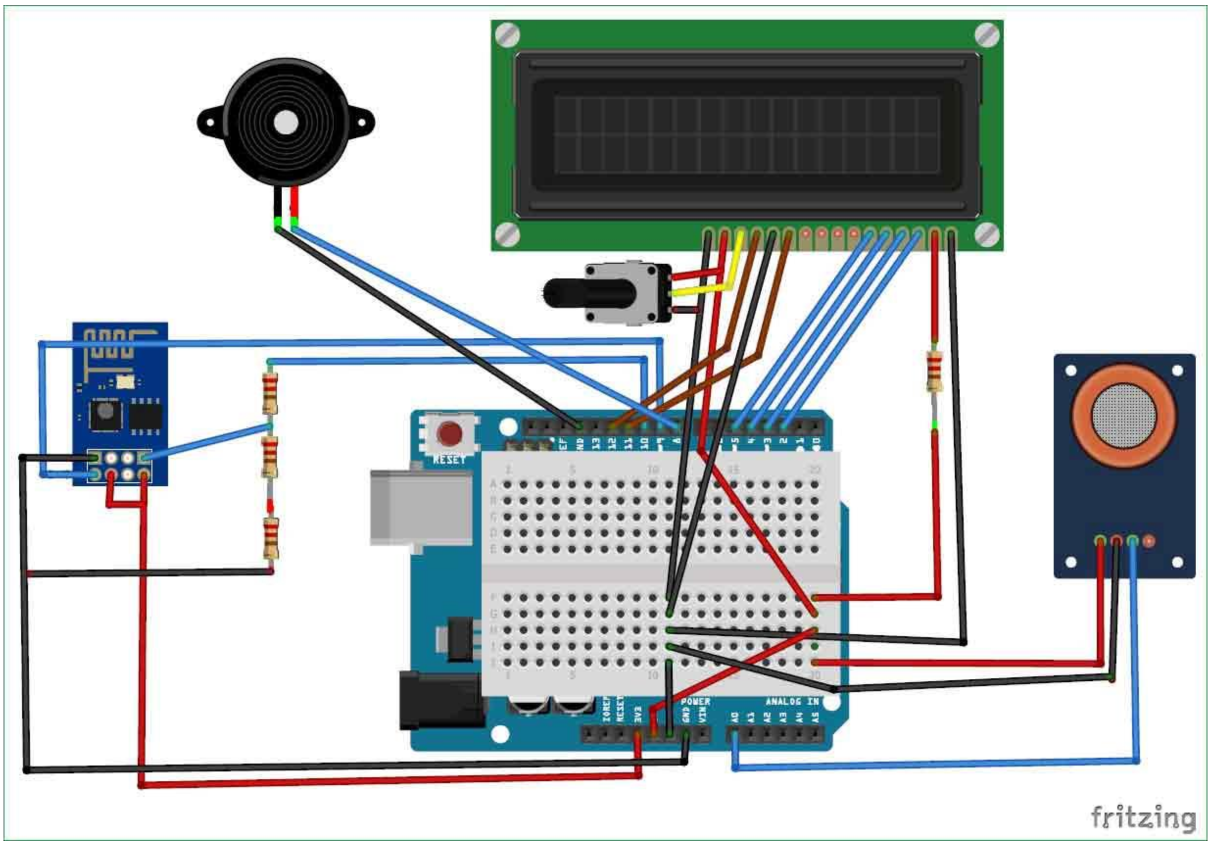
Connect pin 11 (D4) to pin 5 of Arduino.

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”.