AIR QUALITY MONITORING

Phase 5: Project Documentation & Submission

In this part you will document your project and prepare it for submission.
Document the Air Quality Monitoring project and prepare it for submission

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 how the real-time air quality monitoring system can raise public awareness about air quality and health impacts.

Submission

- ❖ Share the GitHub repository link containing the project's code and files.
- ❖ Provide instructions on how to replicate the project, set up IoT devices, develop the data-sharing platform, and integrate them using Python.
- ❖ Include example outputs of IoT device data transmission and platform UI.

Objectives

- ❖ There are few objectives that need to be achieved at the end of this project. The objectives of this project are:
- ❖ To design a low cost and portable air pollutant index (API) using particle and gas sensor; ii) To integrate the sensor, Arduino microcontroller and GSM module to form a complete API system.
- ❖ To transmit and receive Air pollution index (API) data via short message service (SMS) using GSM.

IoT Device Setup:

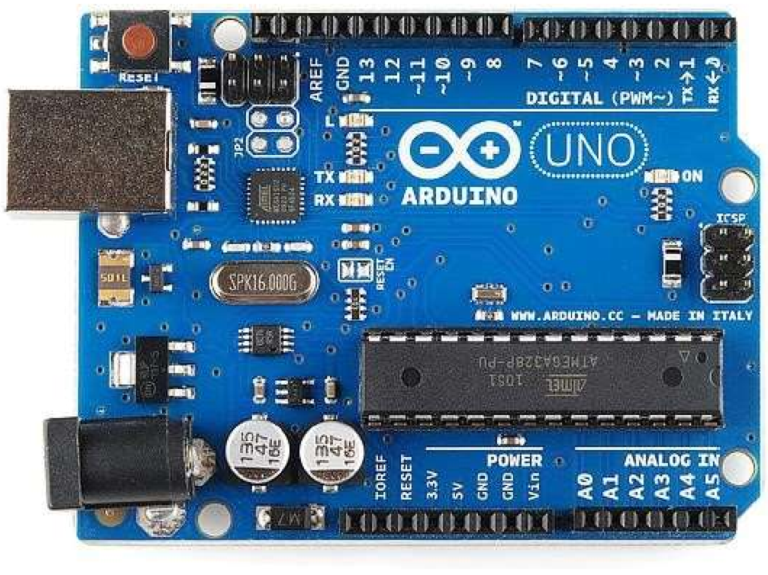
Hardware Requirement

- Arduino Uno
- Wi-Fi module ESP8266
- 16x2 LCD
- MQ135 Gas sensor
- MQ 7 LPG gas sensor
- Buzzer

Arduino UNO

The Arduino Uno R3 is a microcontroller board based on a removable, dual-inlinepackage (DIP) ATmega328 AVR microcontroller. It has 20 digital input/output pins (of which 6 can be used as PWM outputs and 6 can be used as Analog inputs). Programs can be loaded on to it from the easy-to-use Arduino computer program. The Arduino has an extensive support community, which makes it a very easy way to get started working with embedded electronics. The R3 is the third, and latest, revision of the Arduino Uno.

Arduino can be used to communicate with a computer, another Arduino board or other microcontrollers. The Atmega328P microcontroller provides UART TTL (5V) serial communication which can be done using digital pin 0 (Rx) and digital pin 1 (Tx). An Atmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The Atmega16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. There are two RX and TX LEDs on the Arduino board which will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (not for serial communication on pins 0 and 1). A Software Serial library allows for serial communication on any of the Uno’s digital pins. The Atmega328P also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus.



Arduino UNO

The 14 digital input/output pins can be used as input or output pins by using pinMode(), digitalRead() and digitalWrite() functions in 9rduino programming. Each pin operate at 5V and can provide or receive a maximum of 40mA current, and has an internal pull-up resistor of 2050 Kohms which are disconnected by default. Out of these 14 pins, some pins have specific functions as listed below:

- Serial Pins 0 (Rx) and 1 (Tx): Rx and Tx pins are used to receive and transmit TTL serial data. They are connected with the corresponding Atmega328P USB to TTL serial chip.
- External Interrupt Pins 2 and 3**:** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- PWM Pins 3, 5, 6, 9 and 11**:** These pins provide an 8-bit PWM output by using analogWrite() function.
- SPI Pins 10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK): These pins are used for SPI communication.
- In-built LED Pin 13: This pin is connected with a built-in LED, when pin 13 is HIGH – LED is on and when pin 13 is LOW, its off.

- Along with 14 Digital pins, there are 6 analog input pins, each of which provide 10 bits of resolution, i.e., 1024 different values. They measure from 0 to 5 volts, but this limit can be increased by using AREF pin with analog Reference () function.
- Analog pin 4 (SDA) and pin 5 (SCA) also used for TWI communication using Wire library.

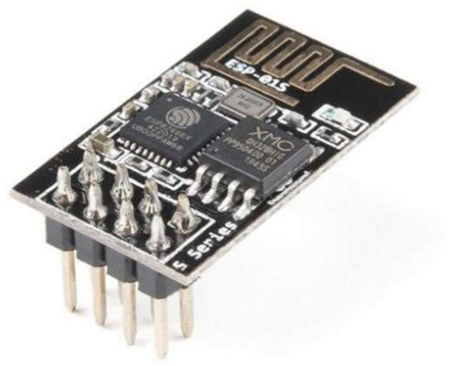
Pin Description:

Pin Category	Pin Name	Details
Power	Vin, 3.3V, 5V, GND	Vin: Input voltage to Arduino when using an external power source. 5V: Regulated power supply used to power microcontroller and other components on the board. 3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA. GND: ground pins.
Reset	Reset	Resets the microcontroller.
Analog Pins	A0 – A5	Used to provide analog input in the range of 0-5V
Input/Output Pins	Digital Pins 0 – 13	Can be used as input or output pins.
Serial	0(Rx), 1(Tx)	Used to receive and transmit TTL serial data.
External Interrupts	2, 3	To trigger an interrupt.
PWM	3, 5, 6, 9, 11	Provides 8-bit PWM output.
SPI	10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK)	Used for SPI communication.
Inbuilt LED	13	To turn on the inbuilt LED.
TWI	A4 (SDA), A5 (SCA)	Used for TWI communication.
AREF	AREF	To provide reference voltage for input voltage.

Pin Description of Arduino UNO

Wi-Fi MODULE ESP8266

The ESP8266 WiFi Module is a self-contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to your WiFi network. The ESP8266 is capable of either hosting an application or offloading all WiFi networking functions from another application processor. Each ESP8266 module comes pre-programmed with an AT command set firmware, meaning, you can simply hook this up to your Arduino device and get about as much WiFi-ability as a WiFi Shield offers (and that's just out of the box)! The ESP8266 module is an extremely cost effective board with a huge, and ever growing, community.



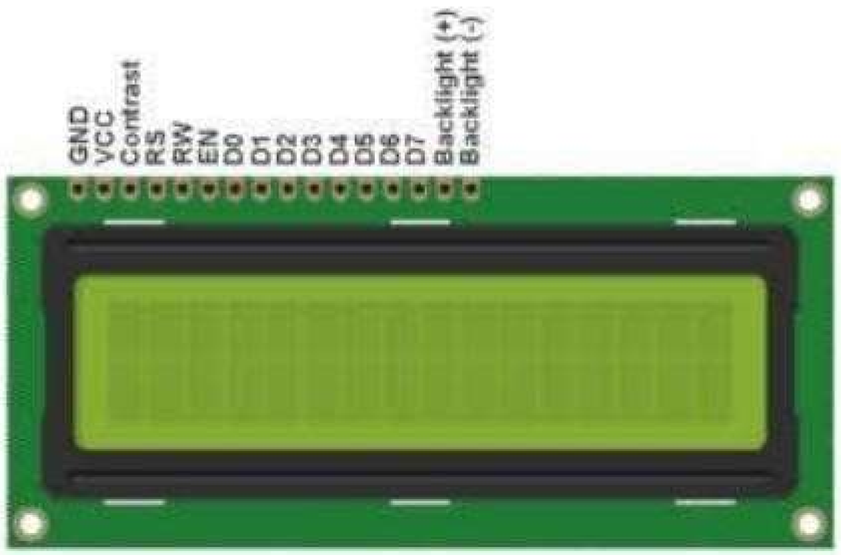
Wi-Fi MODULE ESP8266

This module has a powerful enough on-board processing and storage capability that allows it to be integrated with the sensors and other application specific devices through its GPIOs with minimal development up-front and minimal loading during runtime. Its high degree of on-chip integration allows for minimal external circuitry, including the front-end module, is designed to occupy minimal PCB area. The ESP8266 supports APSD for VoIP applications and Bluetooth co-existance interfaces, it contains a self-calibrated RF allowing it to work under all operating conditions, and requires no external RF parts.

There is an almost limitless fountain of information available for the ESP8266, all of which has been provided by amazing community support. In the *Documents* section below you will find many resources to aid you in using the ESP8266, even instructions on how to transform this module into an IoT (Internet of Things) solution.

16x2 LCD

The term LCD stands for liquid crystal display. It is one kind of electronic display module used in an extensive range of applications like various circuits & devices like mobile phones, calculators, computers, TV sets, etc. These displays are mainly preferred for multisegment light-emitting diodes and seven segments. The main benefits of using this module are inexpensive; simply programmable, animations, and there are no limitations for displaying custom characters, special and even animations, etc.



16x2 LCD Display

A liquid-crystal display (LCD) is a flat-panel display or other electronically modulated optical device that uses the light-modulating properties of liquid crystals. Liquid crystals do not emit light directly, instead using a backlight or reflector to produce images in color or monochrome. [1] LCDs are available to display arbitrary images (as in a general-purpose computer display) or fixed images with low information content, which can be displayed or hidden, such as preset words, digits, and seven-segment displays. The 16×2 LCD pinout is shown below: -

- Pin1 (Ground/Source Pin): This is a GND pin of display, used to connect the GND terminal of the microcontroller unit or power source.
- Pin2 (VCC/Source Pin): This is the voltage supply pin of the display, used to connect the supply pin of the power source.
- Pin3 (V0/VEE/Control Pin): This pin regulates the difference of the display, used to connect a changeable POT that can supply 0 to 5V.
- Pin4 (Register Select/Control Pin): This pin toggles among command or data register, used to connect a microcontroller unit pin and obtains either 0 or 1(0 = data mode, and 1 = command mode).
- Pin5 (Read/Write/Control Pin): This pin toggles the display among the read or writes operation, and it is connected to a microcontroller unit pin to get either 0 or 1 (0 = Write Operation, and 1 = Read Operation).
- Pin 6 (Enable/Control Pin): This pin should be held high to execute Read/Write process, and it is connected to the microcontroller unit & constantly held high.

- Pins 7-14 (Data Pins): These pins are used to send data to the display. These pins are connected in two-wire modes like 4-wire mode and 8-wire mode. In 4-wire mode, only four pins are connected to the microcontroller unit like 0 to 3, whereas in 8-wire mode, 8-pins are connected to microcontroller unit like 0 to 7.
- Pin15 (+ve pin of the LED): This pin is connected to +5V
- Pin 16 (-ve pin of the LED): This pin is connected to GND.

Pin Description:

PIN NO.	FUNCTION	NAME
1.	Ground (0V)	Ground
2.	Supply voltage; 5V (4.7V – 5.3V)	Vcc
3.	Contrast adjustment; through a variable resistor	Vee
4.	Selects command register when low; and data register when high	Register Select
5.	Low to write to the register; High to read from the register	Read/write
6.	Sends data to data pins when a high to low pulse is given	Enable
7.	8-bit data pins	DB0
8.		DB1
9.		DB2
10.		DB3
11.		DB4
12.		DB5
13.		DB6
14.		DB7
15.	Backlight VCC (5V)	Led+
16.	Backlight Ground (0V)	Led -

Pin Description of 16x2 LCD Display

MQ-135 GAS SENSOR

The MQ 135 Air Quality Detector Sensor Module for Arduino has lower conductivity in clean air. When the target combustible gas exists, the conductivity of the sensor is higher along with the gas concentration rising.

Convert change of conductivity to the corresponding output signal of gas concentration. The MQ135 gas sensor has a high sensitivity to Ammonia, Sulphide, and Benzene steam, also sensitive to smoke and other harmful gases.

It is with low cost and suitable for different applications such as harmful gases/smoke detection. The Air quality sensor detects ammonia, nitrogen oxide, smoke, CO2, and other harmful gases. The air quality sensor has a small potentiometer that permits the adjustment of the load resistance of the sensor circuit. The 5V power supply is used for air quality sensor.

MQ-135 Gas Sensor

Pin Description:

1. The VDD power supply 5V DC

2, GND, used to connect the module to system ground

3. DIGITAL OUT, you can also use this sensor to get digital output from this pin, by setting a threshold value using the potentiometer

4. ANALOG OUT, this pin outputs 0-5V Analog voltage based on the intensity of the gas.

MQ-7 GAS SENSOR

Sensitive material of MQ-7 gas sensor is SnO_2, which with lower conductivity in clean air. It make detection by method of cycle high and low temperature, and detect CO when low temperature (heated by 1.5V).

MQ-7 Gas Sensor

The sensor’s conductivity is more higher along with the gas concentration rising. When high temperature (heated by 5.0V), it cleans the other gases adsorbed under low temperature. Please use simple electro-circuit, convert change of conductivity to correspond output signal of gas concentration. MQ-7 gas sensor has high sensitivity to Carbon Monoxide. The sensor could be used to detect different gases contains CO, it is with low cost and suitable for different application.

BUZZER

The buzzer consists of an outside case with two pins to attach it to power and ground. ...
When current is applied to the buzzer it causes the ceramic disk to contract or expand.
Changing the This then causes the surrounding disc to vibrate. That's the sound that you hear.

Buzzer

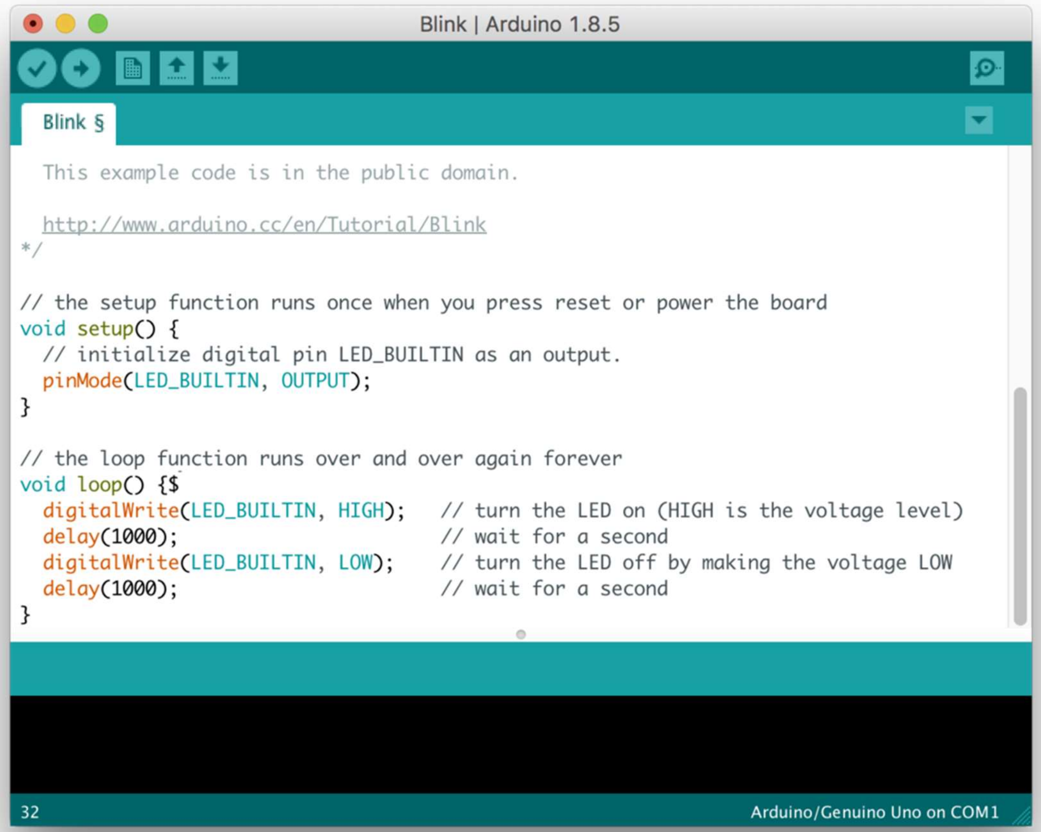
Platform development:

Software Requirements

In our project we specially need here a software to upload the code to Arduino which is following below:

Arduino IDE

The Arduino Integrated Development Environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++. It is used to write and upload programs to Arduino compatible boards, but also, with the help of third-party cores, other vendor development boards.
The source code for the IDE is released under the GNU General Public License, version 2. The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub *main()* into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware. By default, avrdude is used as the uploading tool to flash the user code onto official Arduino boards.
Arduino IDE is a derivative of the Processing IDE, however as of version 2.0, the Processing IDE will be replaced with the Visual Studio Code-based Eclipse Theia IDE framework.



```
  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {$
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Interface of Arduino IDE Software

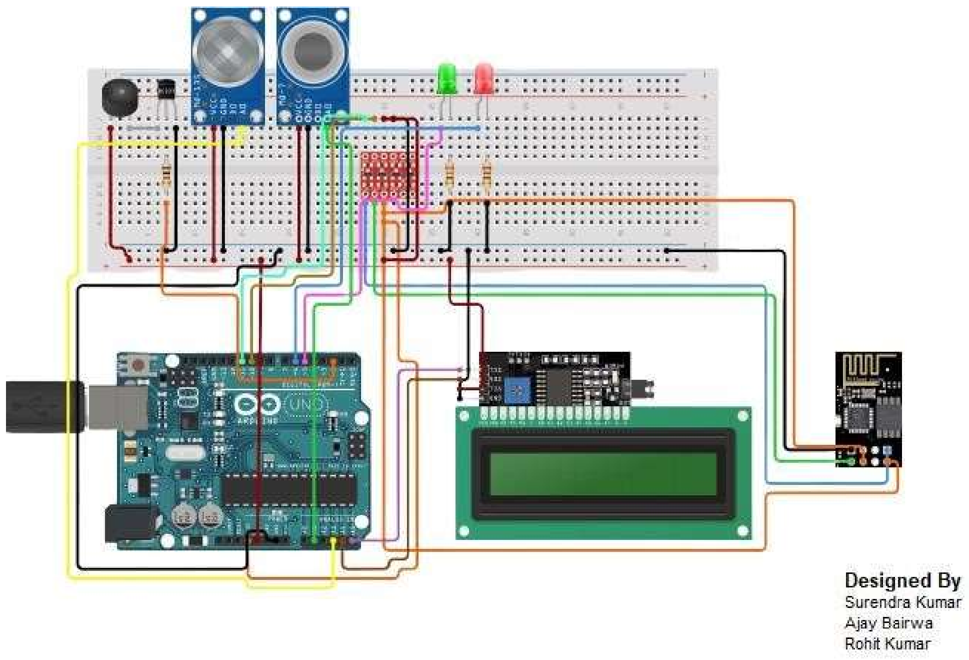
With the rising popularity of Arduino as a software platform, other vendors started to implement custom open source compilers and tools (cores) that can build and upload sketches to other microcontrollers that are not supported by Arduino's official line of microcontrollers.

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50
- Cross-platform - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
- Simple, clear programming environment - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.
- Open source and extensible software - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.
- Open source and extensible hardware - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

Circuit Diagram

The circuit diagram of Air Quality Monitoring System is given below diagram:



Circuit Diagram of Air Pollution Monitoring System

Circuit Explain

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.

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 ohms 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.

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 "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

In our project, we use Arduino IDE Software to upload the code to Arduino UNO which is given below in details.

CODE:

```
//----------------------------------------------------------------------------------------------------------
//                                          LIBRARIES
//----------------------------------------------------------------------------------------------------------
#include <SoftwareSerial.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16,2);

//----------------------------------------------------------------------------------------------------------
//                                          DEFINES VARIABLES
//----------------------------------------------------------------------------------------------------------
 int buz = 5;        //buzzer connected to pin 7 int fan = 6         // fan connected to pin 6
 int greenled = 8;   //green led connected to pin 8 int redled = 9;     //red led connected to pin 9

SoftwareSerial esp8266(2, 3);   // Rx, Tx

//========== MQ135 Sensor variable int gas_sensor = A0;    //Sensor pin
float m = -0.353;       //Slope  float c = 0.711;        //Y-Intercept  float R0 = 23.30;
//Sensor Resistance in fresh air from previous code 21.30

//========== MQ7 Sensor variable int
CO_sensor = A1; //Sensor pin  float m1
= -0.67; //Slope
float c1 = 1.34; //Y-Intercept  float R01 =
5.80; //Sensor Resistance 4.80
```

```
//-----------------------------------------------------------------------------------------------------------
//                                             WI-FI AND CHANNEL DETAILS
//-----------------------------------------------------------------------------------------------------------
String apiKey = "8UZGITPS1BK2WOOX";  // "Thingspeak API key"
String mySSID = "Xav!er";       // WiFi SSID
String myPWD = "12345678"; // WiFi Password

//-----------------------------------------------------------------------------------------------------------
//                                             SETUP
//-----------------------------------------------------------------------------------------------------------
void setup() {
  Serial.begin(9600);              // PC to Arduino Serial Monitor   esp8266.begin(115200);
// Arduino to ESP01 Communication
   lcd.init();                     // initialize the lcd
lcd.backlight();   lcd.print(" Air  Pollution ");
lcd.setCursor(0,1);   lcd.print(" Monitor
System ");   delay(4000);   lcd.clear();

  pinMode(buz,OUTPUT);                 // buzzer is connected as Output from Arduino
pinMode(greenled,OUTPUT);            // green led is connected as output from Arduino
pinMode(redled, OUTPUT);      // red led is connected as output from Arduino
pinMode(fan, OUTPUT);   pinMode(gas_sensor, INPUT);
pinMode(CO_sensor,INPUT);



//------------------// Connection of ESP8266 //------------------//
esp8266.println("AT");          // OK   esp8266.println("AT+RST");
//ready

  unsigned char check_connection=0;
unsigned char times_check=0;
Serial.println("Connecting to Wifi");
lcd.print("Connect to Wifi.");
delay(1000);   lcd.clear();

  String cmd ="AT+CWJAP="Xav!er","12345678"";     // ""+ mySSID +"",""+ myPWD +"""
esp8266.println(cmd);   delay(5000);

  while(check_connection==0)
  {
    if(esp8266.find("OK"))
    {
        Serial.println("WIFI CONNECTED.");
lcd.print("WIFI CONNECTED.");
delay(2000);       lcd.clear();       break;
    }
    times_check++;
if(times_check>3)
     {
        times_check=0;
        Serial.println("Trying to Reconnect..");
lcd.setCursor(0,0);       lcd.print("    Trying to
");       lcd.setCursor(0,1);       lcd.print("
Reconnect...  ");       esp8266.println(cmd);
delay(5000);       lcd.clear();
      }
    }
    //-----------------// Connection of ESP8266 //-----------------//

  esp8266.println("AT+CWMODE=1");
esp8266.println("AT+CIFSR");   delay(1000);
 }
//---------------------------------------------------------------------------------------------------------
//                                             SETUP
//---------------------------------------------------------------------------------------------------------



//---------------------------------------------------------------------------------------------------------
//                                           MAIN LOOP
//---------------------------------------------------------------------------------------------------------
 void loop() {


//========== MQ135 Sensor variable
  float sensor_volt;         //Define variable for sensor voltage    float
RS_gas;              //Define variable for sensor resistance     float
ratio;              //Define variable for ratio
  float sensorValue = analogRead(gas_sensor);   //Read analog values of sensor
sensor_volt = sensorValue*(5.0/1024.0);      //Convert analog values to voltage
RS_gas = ((5.0*10.0)/sensor_volt)-10.0;      //Get value of RS in a gas   ratio =
RS_gas/R0;                         // Get ratio RS_gas/RS_air
```

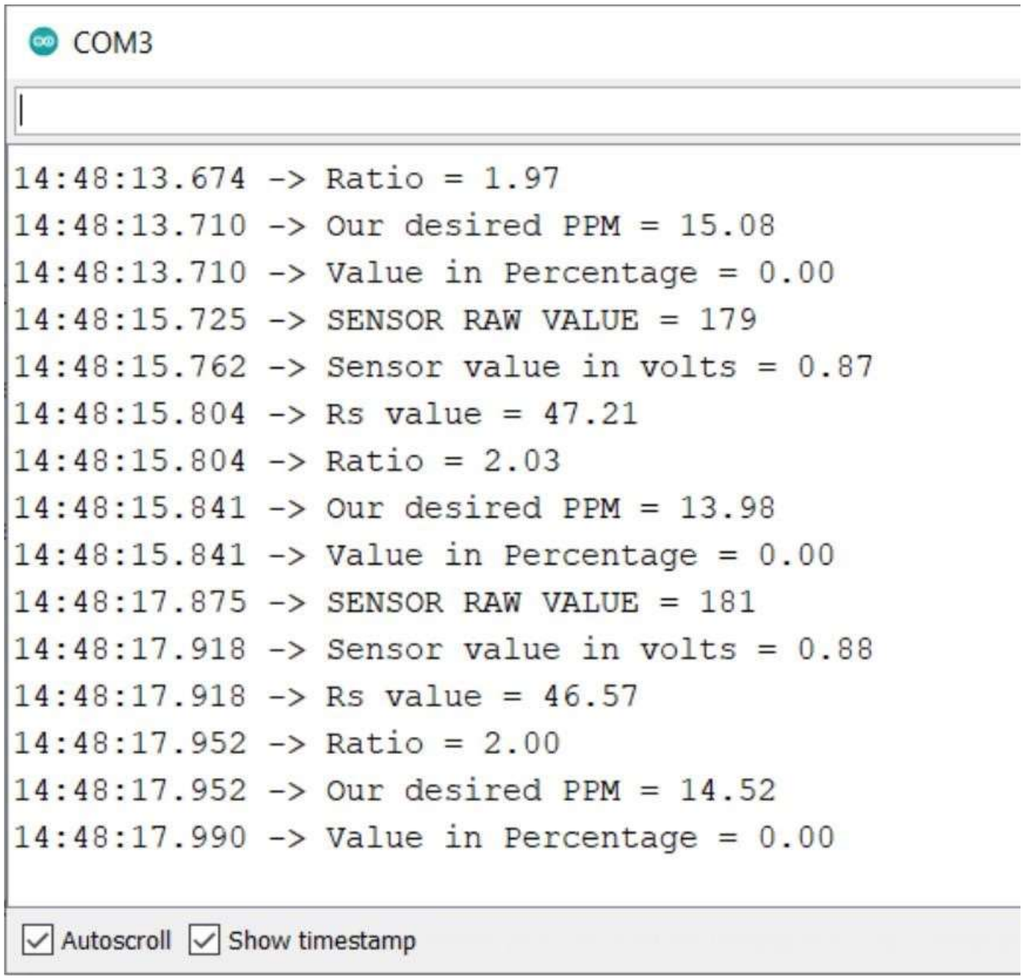
```
  double ppm_log = (log10(ratio)-c)/m;         //Get ppm value in linear scale according to the
the ratio value     double ppm = pow(10, ppm_log);              //Convert ppm value to log scale


  lcd.setCursor(0,0);            // set cursor of lcd to 1st row and 1st column
  lcd.print("CO2: ");            // print message on lcd   lcd.print(ppm);
// print value of MQ135


  //========== MQ7 Sensor variable
  float sensor_volt1;             //Define variable for sensor voltage    float RS_gas1;
//Define variable for sensor resistance     float ratio1;                    //Define variable
for ratio   float sensorValue1 = analogRead(CO_sensor);    //Read analog values of
sensor
  sensor_volt1 = sensorValue1*(5.0/1024.0);           //Convert analog values to voltage
  RS_gas1 = ((5.0*10.0)/sensor_volt1)-10.0;           //Get value of RS in a gas
  ratio1 = RS_gas1/R01;                                // Get ratio RS_gas/RS_air
  double ppm_log1 = (log10(ratio1)-c1)/m1;      to    //Get ppm value in linear scale according
the the ratio value

  double ppm1 = pow(10, ppm_log1);                    //Convert ppm value to log scale


  lcd.setCursor(0,1);              // set cursor of lcd to 1st row and 1st column
lcd.print("CO PPM = ");          // print message on lcd   lcd.print(ppm1);                //
print value of MQ7   delay(5000);



  //------Sending Data to ESP8266--------//
  esp8266.println("AT+CIPMUX=0");           // To Set MUX = 0 for single wifi and MUX=1 for
multiple


  // TCP connection  AT+CIPSTART=4,"TCP","184.106.153.149",80
  String cmd = "AT+CIPSTART="TCP","";      // TCP connection with https://thingspeak.com
server   cmd += "184.106.153.149";                    // IP addr of api.thingspeak.com   cmd +=
"",80";                                // Port No. = 80


  esp8266.println(cmd);                          // Display above Command on PC
Serial.println(cmd);                   // Send above command to Rx1, Tx1
if(esp8266.find("ERROR"))                        // If returns error in TCP connection
  {
    Serial.println("TCP Connection Error");   // Display error msg to PC
return;                            // Try TCP connection   }
```

```
 //prepareGETstringGET /update?api_key=8UZGITPS1BK2WOOX&field1=0.5&field2=5.0

String getStr = "GET /update?api_key=";
getStr += apiKey;   getStr +="&field1=";
getStr += ppm;   getStr +="&field2=";
getStr += ppm1;   getStr += "rnrn";

 cmd = "AT+CIPSEND=";                  // send data length    cmd
+= String(getStr.length());

 esp8266.println(cmd);                 // Send Data length command to Tx1, Rx1
Serial.println(cmd);                   // Display Data length on PC   esp8266.print(getStr);

 if(esp8266.find("SEND OK"))                    // If prompt opens //verify connection with cloud   {
  Serial.println("Pushed whole data TO CLOUD");  // Display confirmation msg to PC
Serial.println(getStr);                        // Display GET String on PC     lcd.clear();
lcd.setCursor(0,0);                            // set cursor of lcd to 1st row and 1st column
lcd.print("Upload to Cloud.");

   //------Check condition for buzzer and LED--------//     if
(ppm >= 10 | ppm1 >= 10) {       digitalWrite(greenled,
LOW);
    digitalWrite(buz, HIGH);
digitalWrite(redled, HIGH);
digitalWrite(fan, HIGH);
lcd.setCursor(0,1);       lcd.print("Polluted
Air ಠ_ಠ");       Serial.println("Alert!!!");
delay(2000); // wait 2000ms       lcd.clear();
  }

  else {       digitalWrite(greenled,
HIGH);       digitalWrite(redled, LOW);
digitalWrite(buz, LOW);
digitalWrite(fan, LOW);
lcd.setCursor(0,1);       lcd.print("
Normal Air ◉‿◉ ");
Serial.println("Normal");
delay(2000); // wait 500ms

lcd.clear();

   }
  //------Check condition for buzzer and LED--------//
 }   else
 {
   esp8266.println("AT+CIPCLOSE");         // Send Close Connection command to Rx1, Tx1
   Serial.println("Uploading Error.");
   Serial.println("AT+CIPCLOSE");          // Display Connection closed command on PC
lcd.clear();     lcd.setCursor(0,0);     lcd.print("Uploading Error.");
    //------Check condition for buzzer and LED--------//     if
(ppm >= 10 | ppm1 >= 10) {       digitalWrite(greenled,
LOW);       digitalWrite(buz, HIGH);
digitalWrite(redled, HIGH);       digitalWrite(fan, HIGH);
lcd.setCursor(0,1);       lcd.print("Polluted Air ಠ_ಠ");
     Serial.println("Alert!!!");       delay(2000);
// wait 2000ms       lcd.clear();
   }     else {       digitalWrite(greenled,
HIGH);       digitalWrite(redled, LOW);
digitalWrite(buz, LOW);
digitalWrite(fan, LOW);
lcd.setCursor(0,1);       lcd.print("
Normal Air ◉‿◉ ");
Serial.println("Normal");
delay(2000); // wait 500ms
lcd.clear();
   }
    //------Check condition for buzzer and LED--------//
 }
 //------Sending Data to ESP8266--------//



 // thingspeak free version needs 15-20 sec delay between every push
//  delay(15000);                          // wait for 16sec
 }
```

The above entire code is needed to upload to the Arduino UNO using Arduino IDE Software.

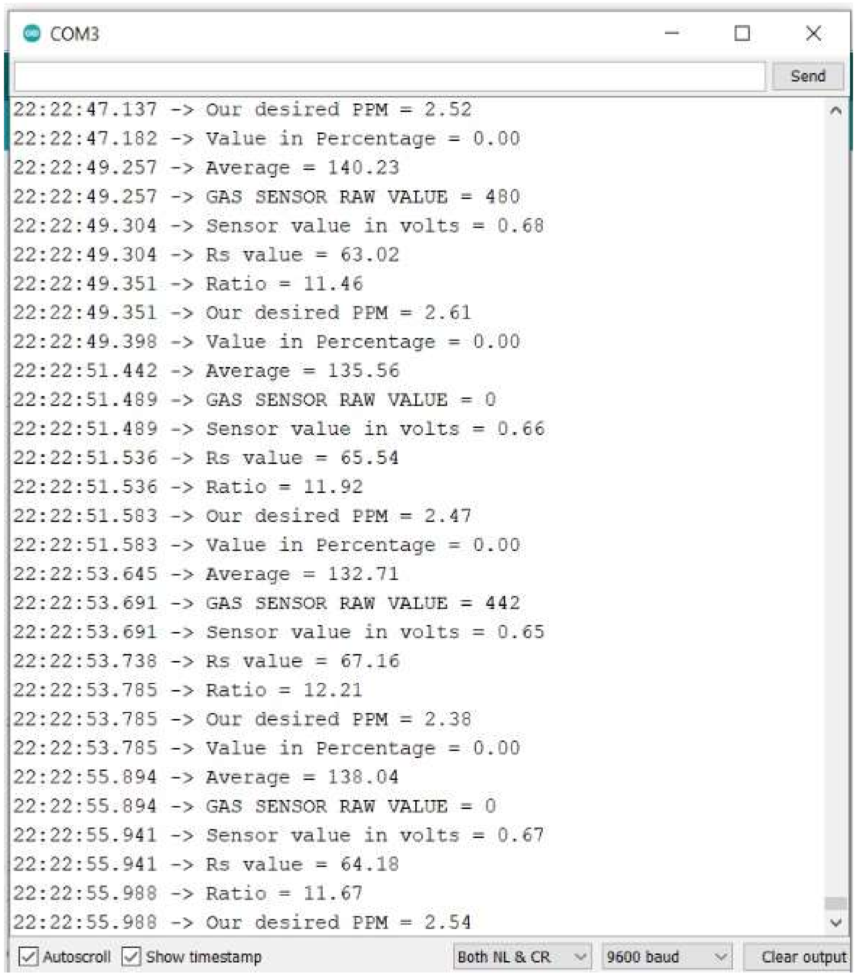
Results

After the testing the system we got results which are following below:



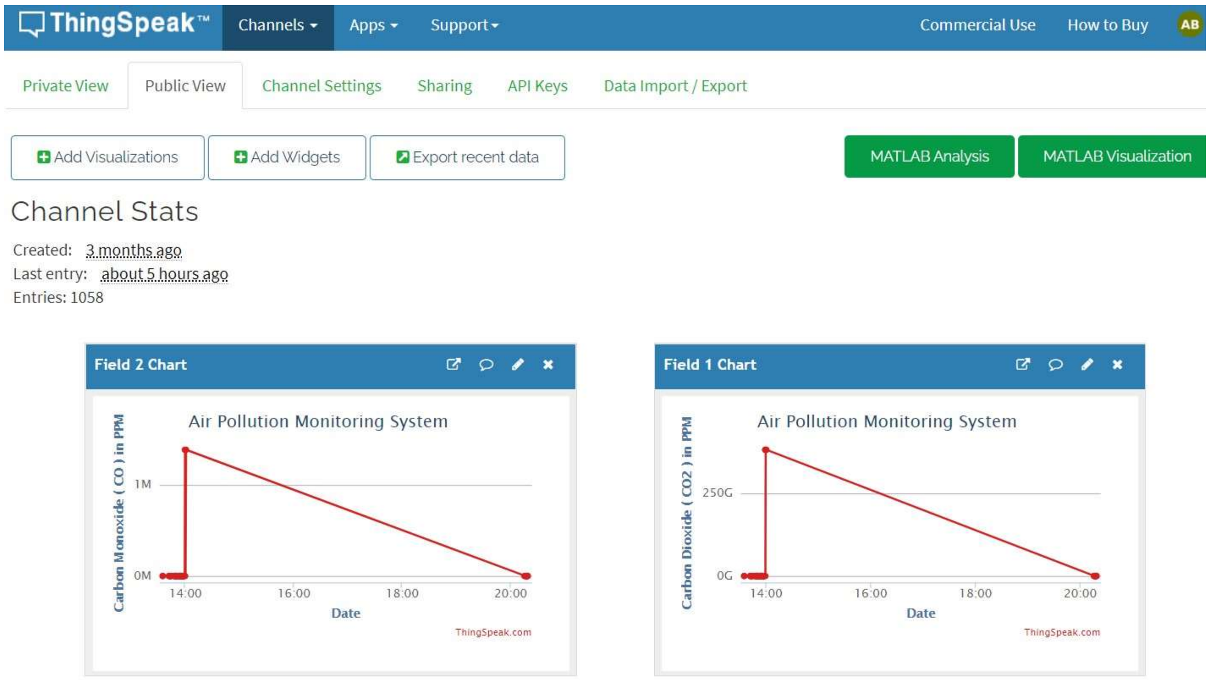
```
14:48:13.674 -> Ratio = 1.97
14:48:13.710 -> Our desired PPM = 15.08
14:48:13.710 -> Value in Percentage = 0.00
14:48:15.725 -> SENSOR RAW VALUE = 179
14:48:15.762 -> Sensor value in volts = 0.87
14:48:15.804 -> Rs value = 47.21
14:48:15.804 -> Ratio = 2.03
14:48:15.841 -> Our desired PPM = 13.98
14:48:15.841 -> Value in Percentage = 0.00
14:48:17.875 -> SENSOR RAW VALUE = 181
14:48:17.918 -> Sensor value in volts = 0.88
14:48:17.918 -> Rs value = 46.57
14:48:17.952 -> Ratio = 2.00
14:48:17.952 -> Our desired PPM = 14.52
14:48:17.990 -> Value in Percentage = 0.00
```

MQ135 Gas Sensor Data



```
22:22:47.137 -> Our desired PPM = 2.52
22:22:47.182 -> Value in Percentage = 0.00
22:22:49.257 -> Average = 140.23
22:22:49.257 -> GAS SENSOR RAW VALUE = 480
22:22:49.304 -> Sensor value in volts = 0.68
22:22:49.304 -> Rs value = 63.02
22:22:49.351 -> Ratio = 11.46
22:22:49.351 -> Our desired PPM = 2.61
22:22:49.398 -> Value in Percentage = 0.00
22:22:51.442 -> Average = 135.56
22:22:51.489 -> GAS SENSOR RAW VALUE = 0
22:22:51.489 -> Sensor value in volts = 0.66
22:22:51.536 -> Rs value = 65.54
22:22:51.536 -> Ratio = 11.92
22:22:51.583 -> Our desired PPM = 2.47
22:22:51.583 -> Value in Percentage = 0.00
22:22:53.645 -> Average = 132.71
22:22:53.691 -> GAS SENSOR RAW VALUE = 442
22:22:53.691 -> Sensor value in volts = 0.65
22:22:53.738 -> Rs value = 67.16
22:22:53.785 -> Ratio = 12.21
22:22:53.785 -> Our desired PPM = 2.38
22:22:53.785 -> Value in Percentage = 0.00
22:22:55.894 -> Average = 138.04
22:22:55.894 -> GAS SENSOR RAW VALUE = 0
22:22:55.941 -> Sensor value in volts = 0.67
22:22:55.941 -> Rs value = 64.18
22:22:55.988 -> Ratio = 11.67
22:22:55.988 -> Our desired PPM = 2.54
```

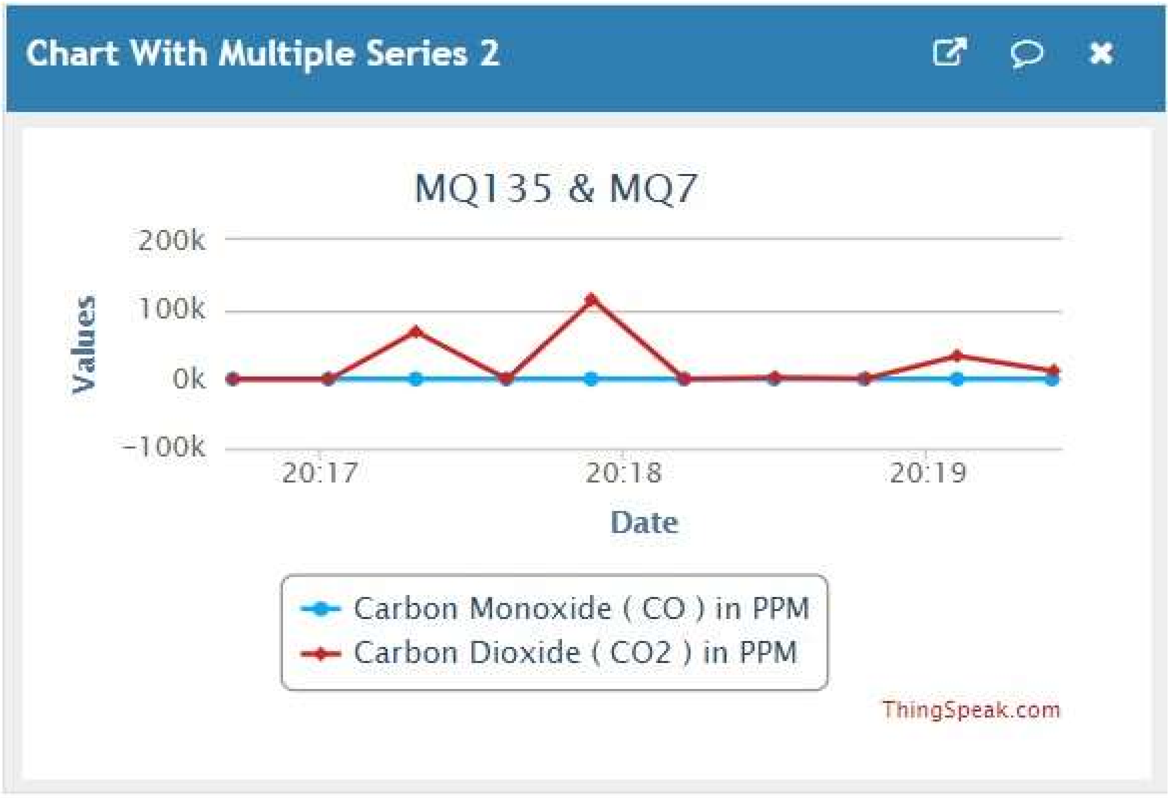
MQ7 Gas Sensor Data



Channel Stats

Created: 3 months ago
Last entry: about 5 hours ago
Entries: 1058

ThinkSpeak Sever Data



Thingspeak Server Data

Health Impacts:

The Sources of India's Polluted Air

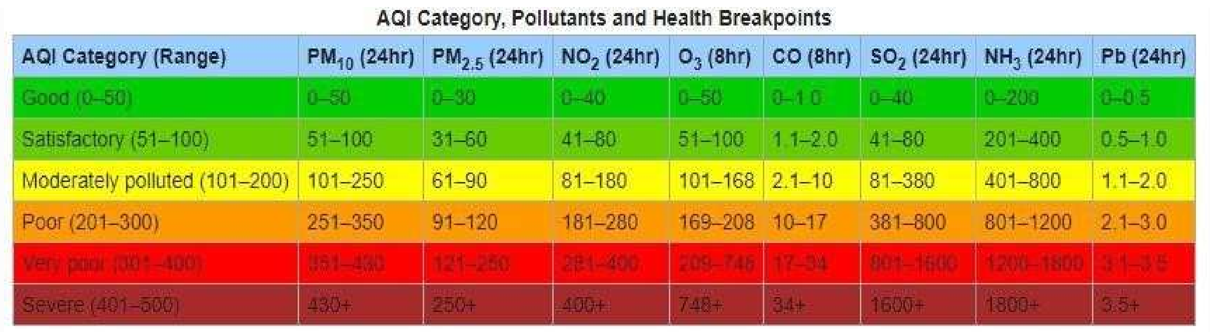
Fuel used for domestic stoves is usually made from a wet mixture of pieces of wood, dried leaves, hay and dried animal dung. This is fashioned into discs and dried in the sunshine. When it is burned in the stoves or chullas, it produces smoke and other pollutants five times higher than if coal were burnt. It is thought that in excess of 100 million households use these stoves up to 3 times a day, 7 days a week. Electricity or other clean fuels are not available in many remote areas. Even in cities where electricity is available, it is traditional to use these types of stove and 24 percent of city pollutants are attributed to such habits.

Some Indian auto-rickshaws and taxis run on fuel that has been adulterated by other, cheaper ingredients. This is a common occurrence in all of South Asia. The taxation system in India exacerbates this situation because gasoline carries a much higher rate of tax than diesel.

This, in turn, carries a higher level than kerosene because kerosene is intended to be used as a cooking fuel. Other volatile liquids such as lubricants and solvents carry little or no tax and therefore make ideal ingredients to mix with the higher-priced fuels. To a low wage earner, this adulteration can save as much as 30 per cent over the period of one month.

Traffic congestion is a huge problem in India’s large cities and towns due to the number of cars trying to use what roads are available. Other factors include a lack of intra-city divided highways and traffic accidents due to the chaotic conditions on India’s roads due to the lackadaisical enforcement of the laws. Because of the bottle-necks created by junctions, traffic remains at a standstill with the engines idling. Monitoring stations near some of the large intersections record noticeably higher figures than those recorded elsewhere Dust produced from the demolition and subsequent building of new properties contributes to the poor quality of air in the city. During the dry season, dust is blown in from the desert-dry countryside and deposited in the city when the wind pressure drops as it travels over the buildings.

The air quality in the capital of Delhi always drops to the “severe” category during the winter months. Primarily, this is due to the practice of burning the stubble after the harvest to prepare for the planting of next season’s crop. It is reported that this alone is responsible for 32 per cent of Delhi’s PM2.5 particulate matter. At 292micrograms per cubic metre, the figure is 5 times higher than the World Health Organisation’s recommended safe limit. Weather conditions play an important part in the dispersal of airborne particles through both wind and rain.
Another major contributor to the air pollution was the Badarpur Thermal Power Station. This was built in 1973 and produced a mere 8 per cent of Delhi’s electricity yet was accountable for
80-90 per cent of the particulate matter. In November 2017 during “The Great Smog of Delhi” it was temporarily shut down to alleviate the smog but restarted the following February. However, due to the amount of pollution it created, it was shut down permanently in late 2018.



AQI Category, Pollutants and Health Breakpoints

AQI Category (Range)	PM_{10} (24hr)	$PM_{2.5}$ (24hr)	NO_2 (24hr)	O_3 (8hr)	CO (8hr)	SO_2 (24hr)	NH_3 (24hr)	Pb (24hr)
Good (0–50)	0–50	0–30	0–40	0–50	0–1.0	0–40	0–200	0–0.5
Satisfactory (51–100)	51–100	31–60	41–80	51–100	1.1–2.0	41–80	201–400	0.5–1.0
Moderately polluted (101–200)	101–250	61–90	81–180	101–168	2.1–10	81–380	401–800	1.1–2.0
Poor (201–300)	251–350	91–120	181–280	169–208	10–17	381–800	801–1200	2.1–3.0
Very poor (301–400)	351–430	121–250	281–400	209–748	17–34	801–1600	1200–1800	3.1–3.5
Severe (401–500)	430+	250+	400+	748+	34+	1600+	1800+	3.5+

AQI of India

The air quality in the capital of Delhi always drops to the "severe" category during the winter months. Primarily, this is due to the practice of burning the stubble after the harvest to prepare for the planting of next season's crop. It is reported that this alone is responsible for 32 per cent of Delhi's PM2.5 particulate matter. At 292micrograms per cubic metre, the figure is 5 times higher than the World Health Organisation's recommended safe limit. Weather conditions play an important part in the dispersal of airborne particles through both wind and rain. Another major contributor to the air pollution was the Badarpur Thermal Power Station. This was built in 1973 and produced a mere 8 per cent of Delhi's electricity yet was accountable for 80-90 per cent of the particulate matter. In November 2017 during "The Great Smog of Delhi" it was temporarily shut down to alleviate the smog but restarted the following February. However, due to the amount of pollution it created, it was shut down permanently in late 2018.

AQI	**Associated Health Impacts**
Good (0-50)	Minimal impact
Satisfactory (51–100)	discomfort to sensitive people.
Moderately polluted (101–200)	May cause breathing discomfort to people with lung disease such as asthma, and discomfort to people with heart disease, children and older adults.
Poor (201–300)	May cause breathing discomfort to people on prolonged exposure, and discomfort to people with heart disease.
Very poor (301–400)	May cause respiratory illness to the people on prolonged exposure. Effect may be more pronounced in people with lung and heart diseases.
Severe (401–500)	May cause respiratory impact even on healthy people, and serious health impacts on people with lung/heart disease. The health impacts may be experienced even during light physical activity

Pollution's Health Impacts

Links to access GitHub repositories:

GitHub repository link: **https://github.com/kousiksuriya/Air-Quality-Monitoring.git**

Conclusion

Smart air pollution monitoring system that constantly keeps track of air quality in an area and displays the air quality measured on an LCD screen the system helps to create awareness of the Quality of air that one breathes daily. This monitoring device can deliver real-time Measurements of air quality. The system to monitor the air of environment using Arduino microcontroller proposed to improve quality of air. Here the using of MQ135 gas sensor gives the sense of different type of dangerous gas and Arduino is the heart of this project which controls the entire process and LCD is used for the visual Output. The Automatic Air management system is a step forward to contribute a solution to the biggest threat. The air & sound monitoring system overcomes the problem of the highly-polluted areas which is a major issue. It supports the new technology and effectively sup ports the healthy life concept.