Project submitted for the Degree of B.Tech in Electronics & Communication Engineering under Maulana Abul Kalam Azad University of Technology

INTERNET OF THINGS (IOT) Based Smoke And Gas Detection Monitoring System

SUBMITTED By

Sayani Paul (24400317018)

Sarbodarshi Mitra (24400317019)

Ankita Kundu (24400317049)

Ankita Ghosh (24400317050)

SUPERVISED BY

Mrs. Tanushree Saha Sarkar

Asst. Professor

Department of Electronics and Communication Engineering

Techno Engineering College Banipur

Banipur, Habra, West Bengal 743233

(JUNE 2021)

CERTIFICATE

This is to certify that the these work titled "**IOT Based Smoke and Gas Detection Monitoring System"** submitted by **Ankita Kundu, Ankita Ghosh, Sarbodarshi Mitra, Sayani Paul** for the partial fulfilment of the requirements of B.TECH degree in Electronics & Communication Engineering, from the institute embodies the work done by them on campus under my supervision.

Date: _____ Signature: _____

Mrs. Tanushree Saha Sarkar

Asst. Prof. & Project Guide

Department of ECE

TECB, Banipur

Date: _____ Signature: _____

Mr.Sanjib Kumar Dhara

Asst. Professor

Coordinator of B.TECH

Department of ECE

TECB, banipur

ACKNOWLEDGEMENT

It is a great privilege for us to express our profound gratitude to our respected teacher Mrs. Tanushree Saha Sarkar, Asst. Professor, Electronics & Communication Engineering, Techno Engineering College Banipur, for her constant guidance, valuable suggestions, supervision and inspiration throughout the course work without which it would have been difficult to complete the work within scheduled time. We are also indebted to the Head of the Department, Mr.Sanjib Kumar Dhara, Electronics & Communication Engineering, Techno Engineering College Banipur for permitting us to pursue the project. We would like to take this opportunity to thank all the respected teachers of this department for being a perennial source of inspiration and showing the right path at the time of necessity.

Ankita Ghosh

Ankita Kundu

Sarbodarshi Mitra

Sayani Paul

SUMMARY

The level of pollution is increasing rapidly due to the factors like industries, urbanization, increasing in population, vehicle use which can affect human health. This proposed IOT Based Air Pollution Monitoring System is used to monitor the Air Quality over a web server using Internet. It will trigger an alarm when the air quality goes beyond a certain level, means when there is sufficient amount of harmful gases present in the air like CO2, smoke. It will show the air quality in PPM on the LCD and as well as on webpage so that air pollution can be monitored very easily. The system uses MQ2 gas sensor for monitoring Air Quality as it detects most harmful gases and can measure their amount accurately and arduino is the brain of this project which controls the entire process. Wi-Fi Module connects the whole process to internet 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.

TABLE OF CONTENT

SL NO.	TOPICS	PAGE NO
1	Introduction	1
2	Objective	2
3	Aim of the project	3
4	Components Required	4
5	Components Explanation	5-15
6	Block Diagram	16
7	Working Principal	17
8	Circuit Diagram	18
9	Circuit Explanation	19-21
10	Result	22-23
11	Source Code & Explanation	24-38
12	Application	39
13	Advantage	40
14	Future Scope	41
15	Conclusion	42
16	Reference	43

List of Figures

SL NO.	CONTENT	PAGE NO.
1	Air pollution	1
2	How IOT Works	3
3	Arduino UNO Pinout	6
4	MQ2 sensor pin Description	7
5	MQ2 sensor connection Diagram	8
6	ESP-12(WiFi Module) pin description	10
7	Buzzer	11
8	Buzzer Pin Description	11
9	Breadboard	12
10	LCD pin Description	13
11	10k potentiometer	13
12	10k potentiometer pin Description	14
13	100 ohm resistor	14
14	Circuit diagram -simulation	18
15	Circuit Diagram -Hardware Implementation	18
16	LCD connection with Arduino	19
17	MQ2 sensor connection with Arduino	20
18	Arduino with LCD connection-Hardware Implementation	21
19	Output-Before gas detection	22
20	Output-After gas detection	22
21	Various parameters being displayed onThingspeak.com	23

List of Table

SL NO.	CONTENT	PAGE NO.
1.	Components Required	4
2.	Arduino Uno Pin Description	5-6
3.	MQ-2 Sensor Pin Description	8
4.	Wi-Fi Module Esp-12E Pin Description	9
5.	Buzzer Pin Description	11
6.	Potentiometer Pin Description	14

INTRODUCTION

Air pollution is the biggest problem of every nation, whether it is developed or developing. Health problems have been growing at faster rate especially in urban areas of developing countries where industrialization and growing number of vehicles leads to release of lot of gaseous pollutants. Harmful effects of pollution include mild allergic reactions such as irritation of the throat, eyes and nose as well as some serious problems like bronchitis, heart diseases, pneumonia, lung and aggravated asthma. According to a survey, due to air pollution 50,000 to 100,000 premature deaths per year occur in the U.S. alone. Whereas in EU number reaches to 300,000 and over 3,000,000 worldwide. IOT Based Air Pollution Monitoring System monitors the Air quality over a web server using Internet and will trigger an alarm when the air quality goes down beyond a certain threshold level, means when there are sufficient amount of harmful gases present in the air like CO,smoke, and LPG. It will show the air quality in PPM on the LCD and as well as on webpage so that it can monitor it very easily. LPG sensor is added in this system which is used mostly in houses. The system can be installed anywhere but mostly in industries and houses where gases are mostly to be found and gives an alert message when the system crosses threshold limit.

Fig- 1

OBJECTIVE

The drawbacks of the conventional monitoring instruments are their large size, heavy weight and extraordinary expensiveness. These lead to sparse deployment of the monitoring stations. In order to be effective, the locations of the monitoring stations need careful placement because the air pollution situation in urban areas is highly related to human activities (e.g. construction activities) and location-dependent (e.g., the traffic choke-points have much worse air quality than average). IOT Based Air Pollution Monitoring System monitors the Air Quality over a webserver using internet and will trigger an alarm when the air quality goes down beyond a certain level, means when there are amount of harmful gases 2present in the air like CO, smoke, and LPG. The system will show the air quality in PPM on the LCD and as well as on webpage so that it can be monitored very easily. MQ-2 sensor is used for monitoring Air Quality as it detects most harmful gases and can measure their amount accurately. In this IOT project, it can monitor the pollution level from anywhere using your computer or mobile. This system can be installed anywhere and can also trigger some device when pollution goes beyond some level, like we can send alert SMS to the user.

Aim of The Project

We propose an air quality pollution monitoring system that allows us to monitor and check live air quality pollution in an area through IOT. System uses air sensors to sense presence of harmful gases/compounds in the air and constantly transmit this data.

The sensors interact with arduino uno which processes this data and transmits it over the application. This allows authorities to monitor air pollution in different areas and act against it.

HOW IOT WORKS ?

An IoT ecosystem consists of web-enabled smart devices that use embedded systems, such as processors, sensors and communication hardware, to collect, send and act on data they acquire from their environments. IOT devices share the sensor data they collect by connecting to an IOT

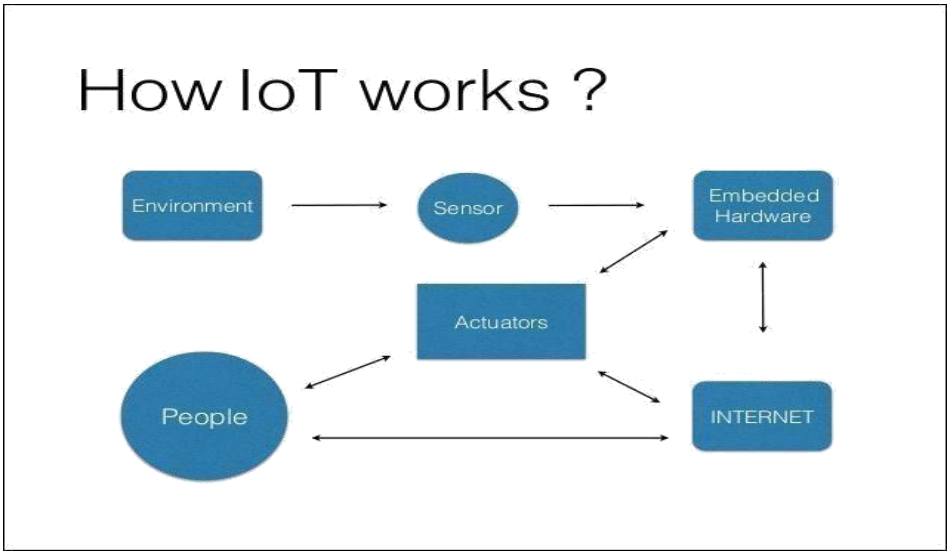


Fig-2

gateway or other edge device where data is either sent to the cloud to be analyzed or analyzed locally. Sometimes, these devices communicate with other related devices and act on the information they get from one another. The devices do most of the work without human intervention, although people can interact with the devices -- for instance, to set them up, give them instructions or access the data.

COMPONENTS REQUIRED

1. **Hardware Requirement:-** 1) MQ2 Gas sensor 2) Arduino Uno 3) Wi-Fi module ESP12E 4) 16x2 LCD 5) Breadboard 6) 10K potentiometer 7) 100 ohm resistors 8) Buzzer 9) Connecting Wires 10) LED

2. **Software Requirement:-** 1) Arduino 1.8.13 Software 2) Embedded C Language 3)ThingSpeak IOT platform 4) Blynk IOT platform.

SL NO.	COMPONENTS	QUANTITY
1.	Arduino Uno	1
2.	MQ2 Gas Sensor	1
3.	WiFi Module ESP8266	1
4.	LCD Screen	1
5.	Breadboard	1
6.	10 k Potentiometer	1
7.	Buzzer	1
8.	100 ohm Resistor	3
9.	Connecting Wires	As Required
10.	LED	2

COMPONENT EXPLANATION

ARDUINO UNO :-

Arduino Uno is a microcontroller board based on 8-bit ATmega328P microcontroller. Along with ATmega328P, it consists other components such as crystal oscillator, serial communication, voltage regulator, etc. to support the microcontroller. Arduino Uno has 14 digital input/output pins (out of which 6 can be used as PWM outputs), 6 analog input pins, a USB connection, A Power barrel jack, an ICSP header and a reset button.

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.

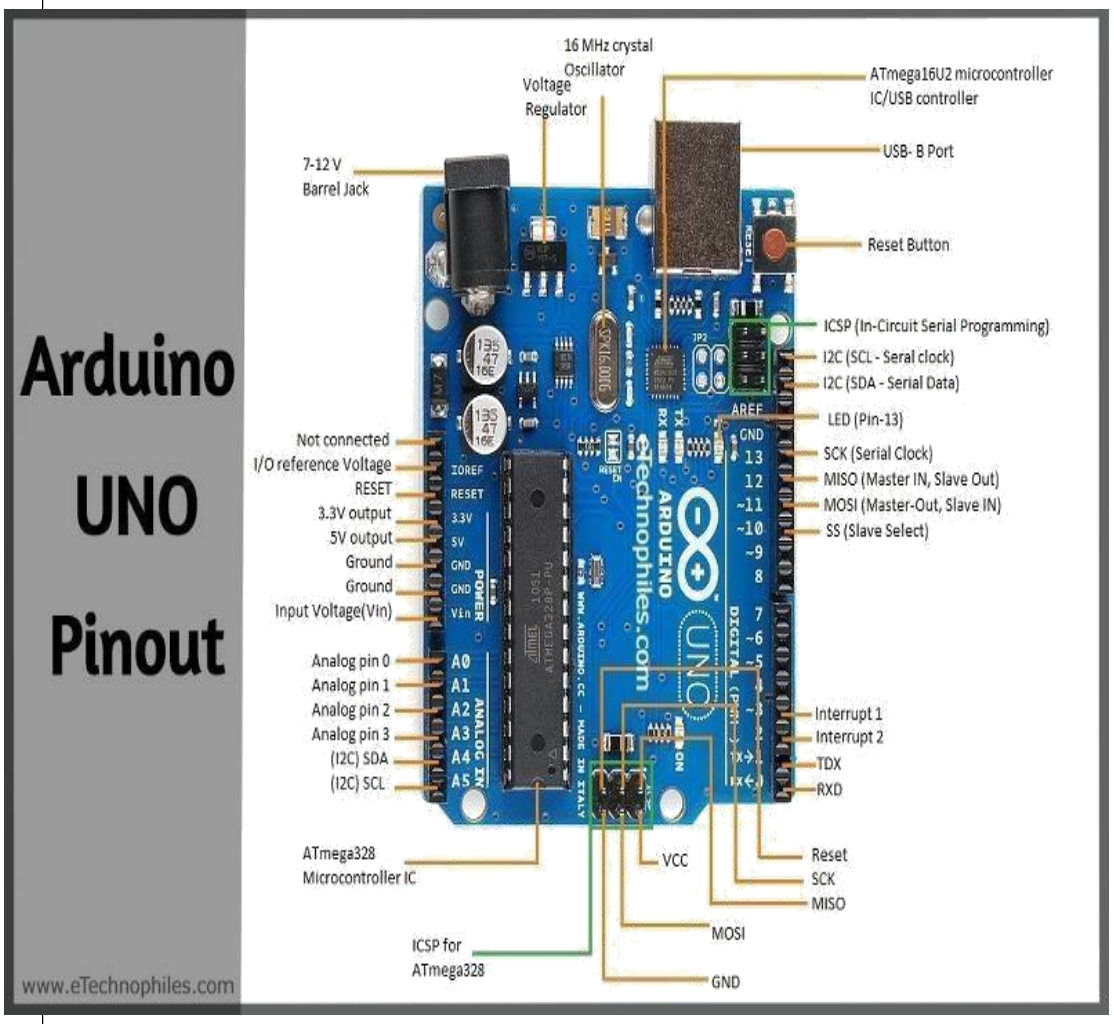


Fig-3

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs -light on a sensor, a finger on a button, or a Twitter message -and turn it into an output -activating a motor, turning on an LED, publishing something online.

All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

MQ-2 SENSOR :-

MQ2 gas sensor is an electronic sensor used for sensing the concentration of gases in the air such as LPG, propane, methane, hydrogen, alcohol, smoke and carbon monoxide.

MQ2 gas sensor is also known as chemiresistor. It contains a sensing material whose resistance changes when it comes in contact with the gas. This change in the value of resistance is used for the detection of gas.

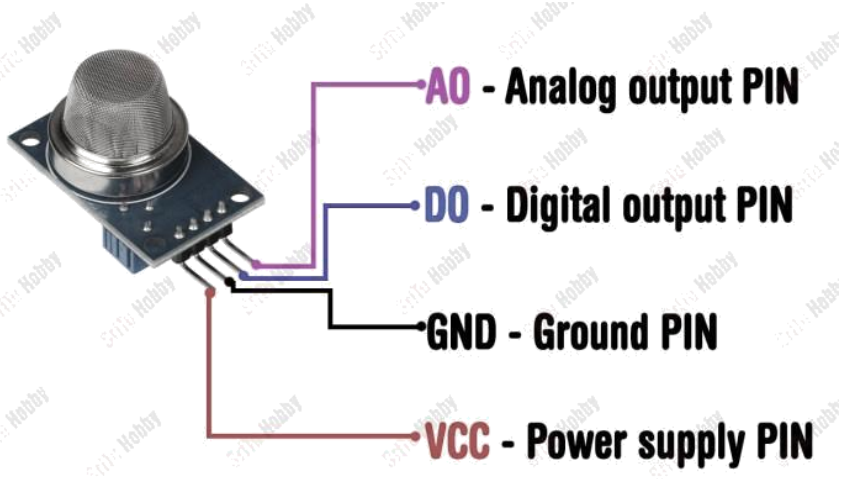
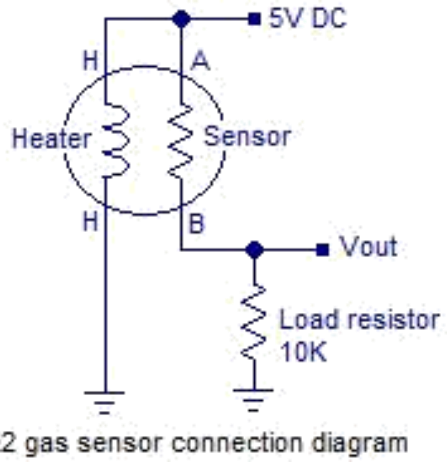


Fig-4

MQ2 is a metal oxide semiconductor type gas sensor. Concentrations of gas in the gas is measured using a voltage divider network present in the sensor. This sensor works on 5V DC voltage. It can detect gases in the concentration of range 200 to 10000ppm.

Pin Description:

1	Vcc	This pin powers the module, typically the operating voltage is +5V
2	Ground	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



MQ2 gas sensor connection diagram

Fig-5

WIFI MODULE ESP12E :-

ESP-12E is a miniature **Wi-Fi module** present in the market and is used for establishing a wireless network connection for microcontroller or processor. The core of ESP-12E is **ESP8266EX**, which is a high integration wireless SoC (System on Chip). It features ability to embed Wi-Fi capabilities to systems or to function as a standalone application. It is a low cost solution for developing IoT applications.

ESP12E Pin Configuration

The ESP-12E module has twenty two pins and we will describe function of each pin below.

Pin	Name	Description
1	RST	Reset Pin of the module
2	ADC	Analog Input Pin for 10-bit ADC (0V to1V)
3	EN	Module Enable Pin (Active HIGH)
4	GPIO16	General Purpose Input Output Pin 16
5	GPIO14	General Purpose Input Output Pin 14
6	GPIO12	General Purpose Input Output Pin 12
7	GPIO13	General Purpose Input Output Pin 13
8	VDD	+3.3V Power Input
9	CS0	Chip selection Pin of SPI interface
10	MISO	MISO Pin of SPI interface
11	GPIO9	General Purpose Input Output Pin 9
12	GPIO10	General Purpose Input Output Pin 10
13	MOSI	MOSI Pin of SPI interface
14	SCLK	Clock Pin of SPI interface
15	GND	Ground Pin
16	GPIO15	General Purpose Input Output Pin 15
17	GPIO2	General Purpose Input Output Pin 2
18	GPIO0	General Purpose Input Output Pin 0
19	GPIO4	General Purpose Input Output Pin 4
20	GPIO5	General Purpose Input Output Pin 5
21	RXD0	UART0 RXD Pin
22	TXD0	UART0 TXD Pin

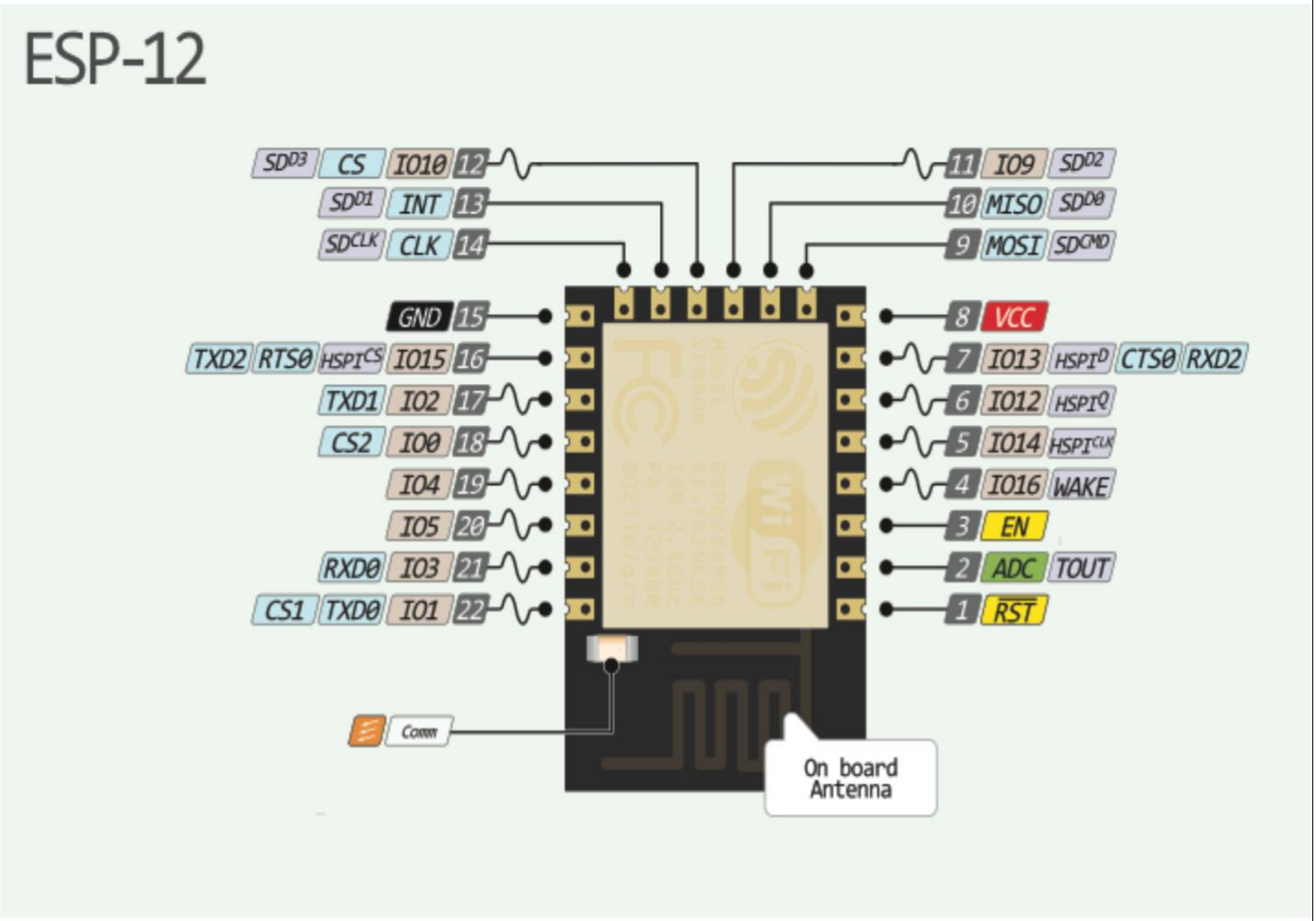


Fig-6

BUZZER :-

A **buzzer** is a small yet efficient component to add sound features to our project/system. It is very small and compact 2-pin structure hence can be easily used on breadboard. A Buzzer or beeper is an audio signaling device. Whenever the air pollution goes above the threshold level the Buzzer starts beeping indicating Danger.

Fig-7

Buzzer Pin Configuration

Pin Number	Pin Name	Description
1	Positive	Identified by (+) symbol or longer terminal lead. Can be powered by 6V DC
2	Negative	Identified by short terminal lead. Typically connected to the ground of the circuit

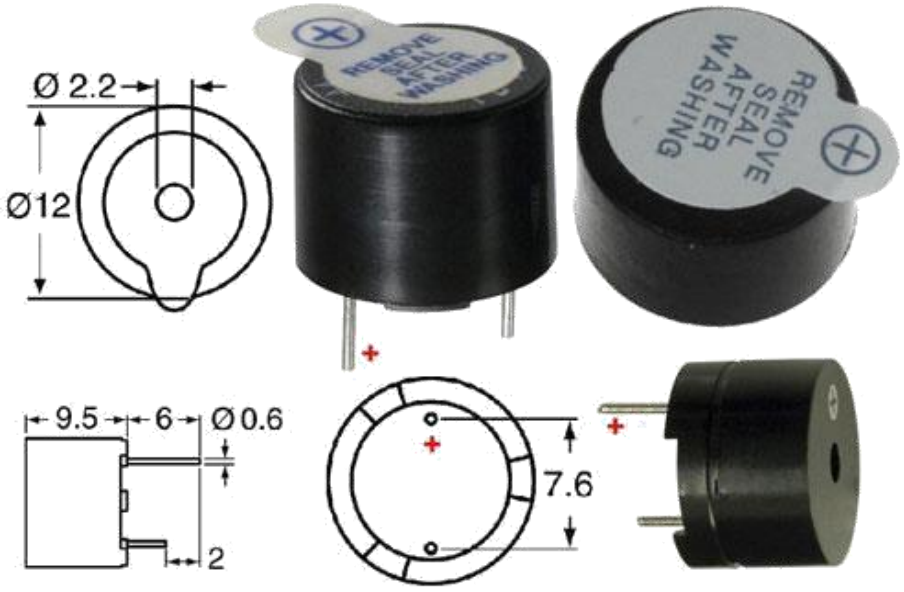


Fig-8

BREADBOARD :-

A breadboard is a rectangular plastic board with a bunch of tiny holes in it. These holes let you easily insert electronic components to **prototype** (meaning to build and test an early version of) an electronic circuit, like this one with a battery, switch, resistor, and an LED (light-emitting diode)

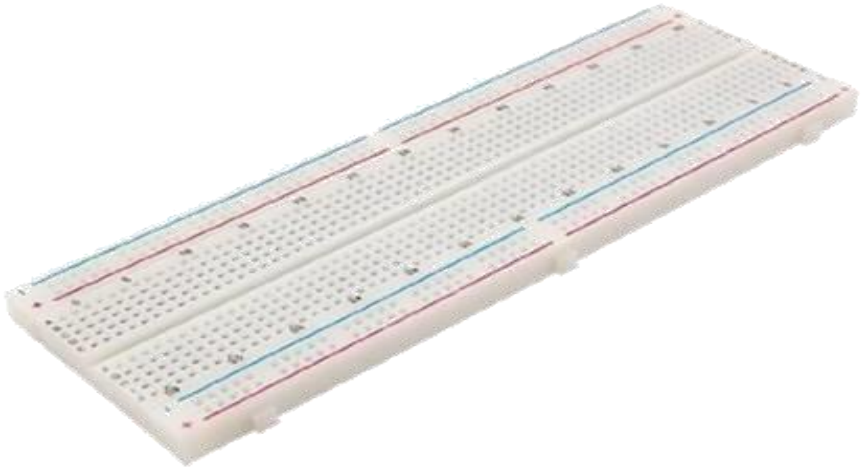
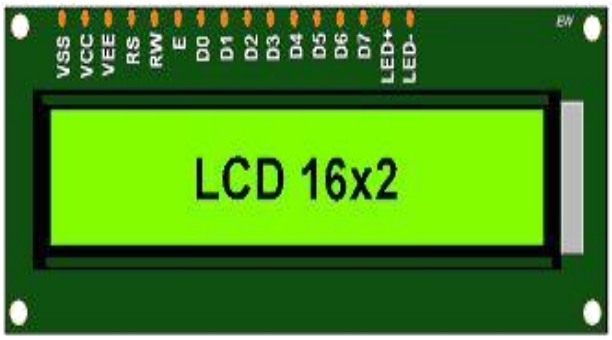


Fig-9

LCD DISPLAY :-

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 multi-segment 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.Here WE have used a basic (16x2) 16 character by 2 line display. Black text on Green background. It is used to indicate the Air in PPM



No.	PIN	Function
1	VSS	Ground
2	VCC	+5 Volt
3	VEE	Contrast control 0 Volt: High contrast.

No.	PIN	Function
4	RS	Register Select 0: Commannd Reg. 1: Data Reg.
5	RW	Read / write 0: Write 1: Read
6	E	Enable H-L pulse
7-14	D0 - D7	Data Pins D7: Busy Flag Pin
15	LED+	+5 Volt
16	LED-	Ground

Fig-10

10k Potentiometer :-

The potentiometer is used to control the screen contrast of the LCD. In our project we used 10K potentiometer but other than 10k value will work too.

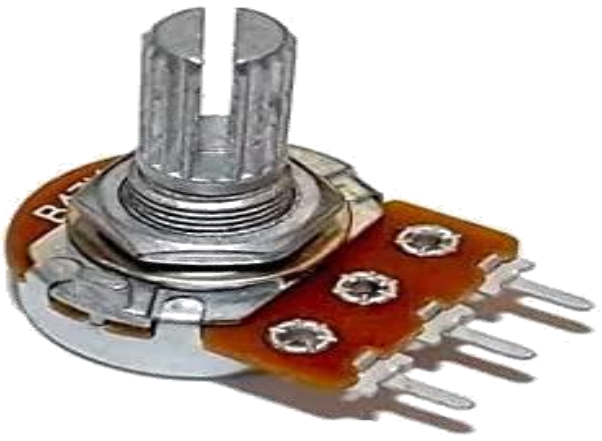


Fig-11

Potentiometer Pin Configuration :-

Pin No.	Pin Name	Description
1	Fixed End	This end is connected to one end of the resistive track
2	Variable End	This end is connected to the wiper, to provide variable voltage
3	Fixed End	This end is connected to another end of the resistive track

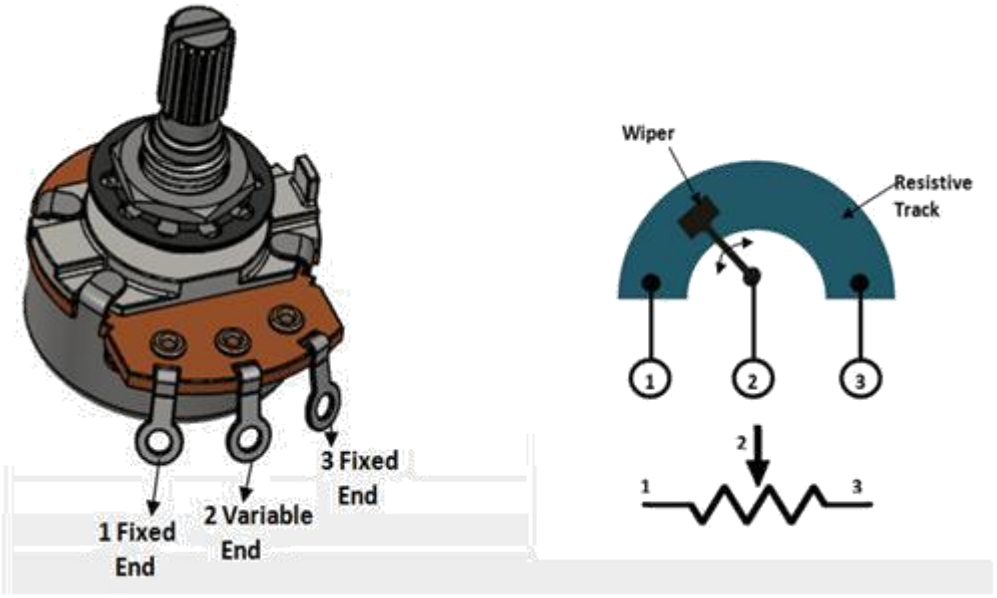


Fig-12

100 Ohm RESISTOR :-

A 100 ohm resistor will conduct .01 amps or 10 milliamps if 1 volt is connected across it or 0.1 amp or 100 milliamps if 10 volts is connected across it. Ohms law I= V/R

They come in many shapes and sizes depending on the maximum power they can dissipate. Top is a standard 1/4 watt resistor. Below is a surface mount resistor and bottom are some 20 watt resistors. (Power in Watts = V x I) power = volts across the resistor by the current flowing through the resistor in amps . this cause heat, so a larger resistor made of more heat resistant material dissipates more heat to the air.

Fig-13

Arduino IDE:-

The Arduino Integrated Development Environment (IDE) is the main text editing program used for Arduino programming. It is where we typing up our code before uploading it to the board . Arduino code is referred to as sketches.

Here we have used the latest version of arduino 1.8.13. And embedded C language is used to write our code.

ThingSpeak IOT Platform :-

The project is based on ThingSpeak cloud computing. ThingSpeak is an open source IoT application and API to store and retrieve data from things using HTTP protocol over the Internet via LAN. It enables the creation of sensor-logging applications, location-tracking applications and a social network of things with status updates. That means, when we send data from the sensors to ThingSpeak at regular intervals, it creates, store and display data in a trend automatically. Eight data per channel. Each channel can take eight data signals from different devices. This means, using ThingSpeak API, one can upload eight data per channel, which are eventually gathered, logged and put into trend data by ThingSpeak.

Blynk IOT Platform :-

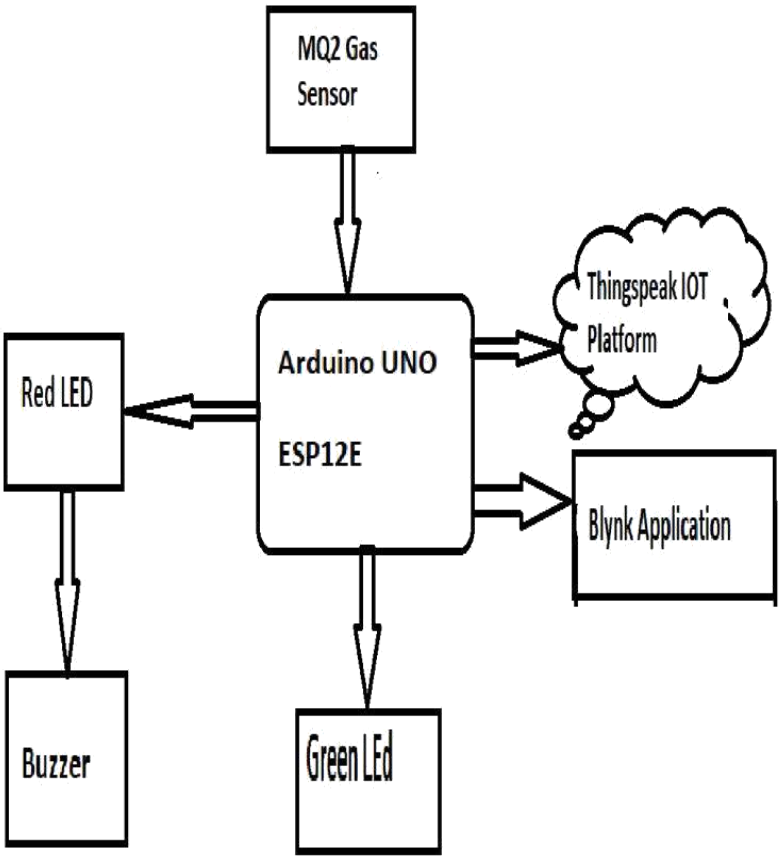
Blynk works over the Internet. This means that it can control hardware remotely, it can display sensor data, it can store data, visualize it and do many other things.

There are three major components in the platform:

- **Blynk App** - Allows us to create amazing interfaces for your projects using various widgets we provide.
- **Blynk Server** - responsible for all the communications between the smartphone and hardware. We can use our Blynk Cloud or run our private Blynk server locally. It's open-source, could easily handle thousands of devices and can even be launched on a Raspberry Pi.
- **Blynk Libraries** - for all the popular hardware platforms - enable communication with the server and process all the incoming and outcoming commands.

After downloading the Blynk app, we can create a project dashboard and arrange buttons, sliders, graphs, and other widgets onto the screen. Using the widgets, we can turn pins on and off or display data from sensors. This app is available for both Android and IOS.

Block Diagram



HOW IT WORKS

The MQ2 sensor can sense LPG, propane, methane, hydrogen, alcohol, smoke and carbon monoxide, 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). MQ2 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 MQ2 sensor, it is explained in detail in “Code Explanation” section below.

The Analog Output pin of the sensor is connected to the A0 pin of the Arduino. The analog output voltage from the sensor can be assumed directly proportional to the concentration of CO2 gas in PPM under standard conditions. The analog voltage is sensed from the sensor and converted to a digital value in range from 0 to 1023 by the inbuilt ADC channel of the controller. The digitized value is hence equal to the gas concentration in PPM.

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 500 PPM, then the LCD and webpage will display “Normal”. Whenever the value will increase 500 PPM, then the buzzer will start beeping, the red led will illuminate and the LCD and webpage will display “Alert”.

The Wi-Fi module is configured to connect with the ThingSpeak IOT platform. ThingSpeak is an IOT analytics platform service that allows to aggregate, visualize and analyze live data streams in the cloud. ThingSpeak provides instant visualizations of data posted by the IOT devices to ThingSpeak server. With the ability to execute MATLAB code in ThingSpeak one can perform online analysis and processing of the data as it comes in.

The Wi-Fi module is also connected with Blynk IOT platform. With the help of this we can easily monitor the air quality through the mobile. And if the fetched parameter surpasses the threshold level then it will also give us notification even though the mobile screen is turned off.

CIRCUIT DIAGRAM

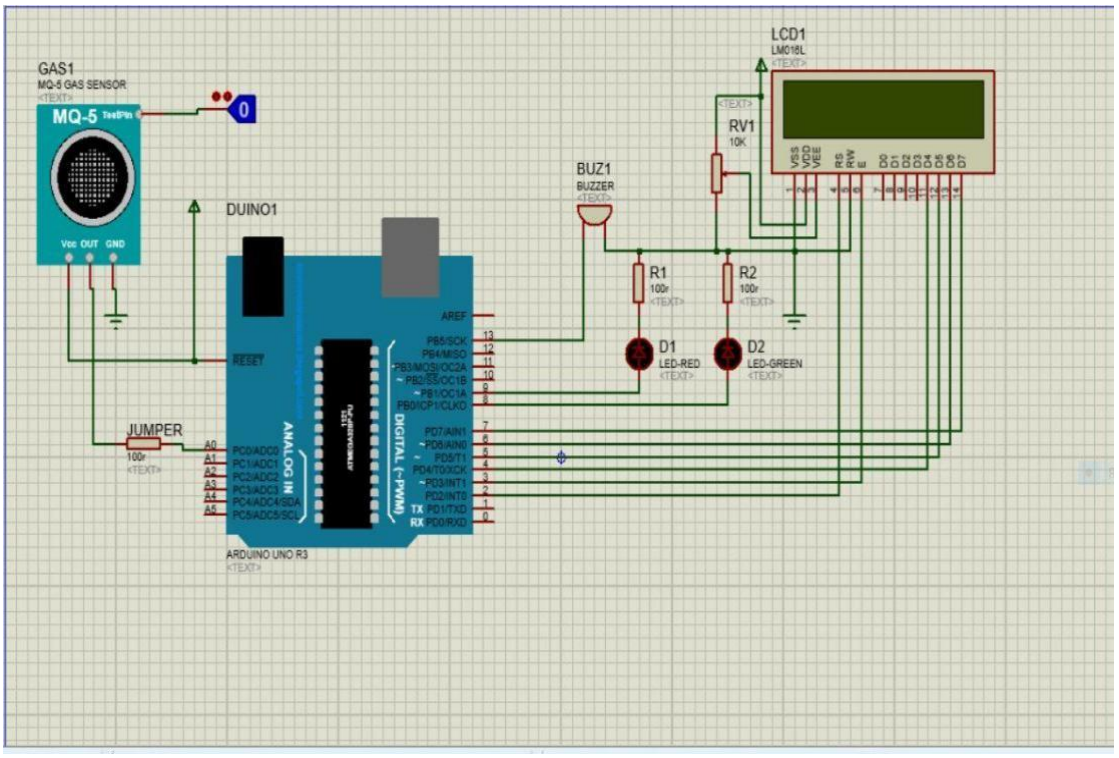


Fig-14

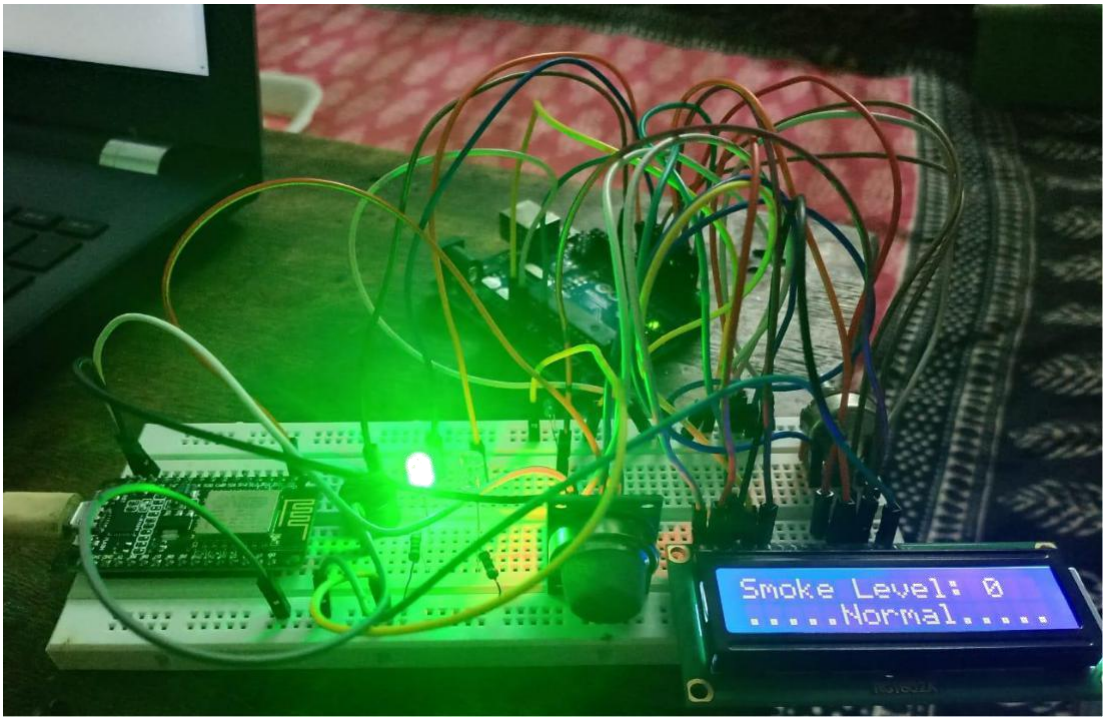


Fig-15

CIRCUIT DIAGRAM EXPLANATION

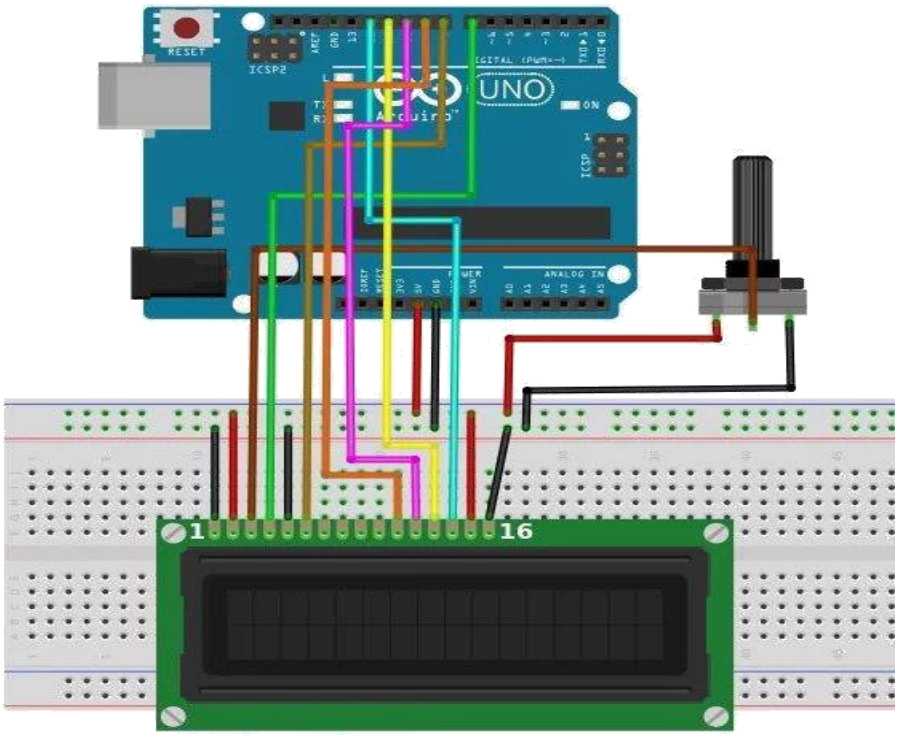


Fig-16

In last, we will connect LCD with 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.

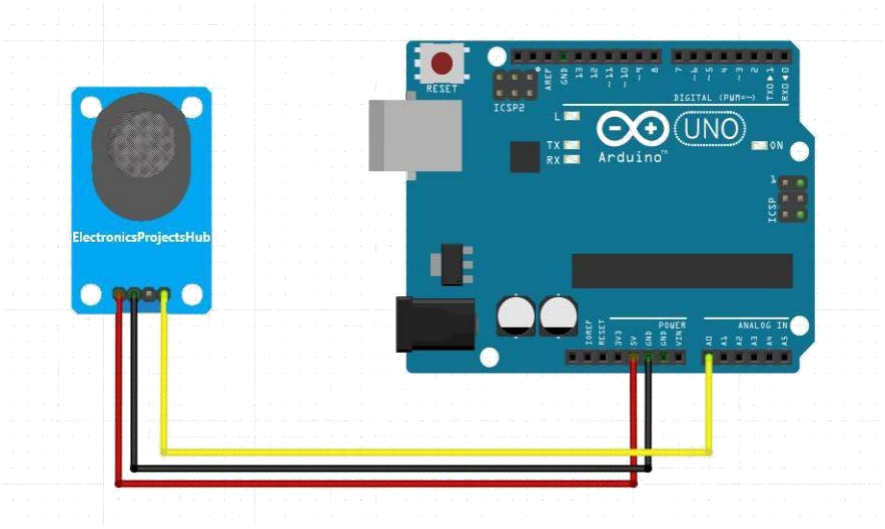


Fig-17

Then we will connect the MQ2 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.

ESP12E Wi-Fi module gives our project 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.

The Reset and VCC pins of the module are connected to the 5V DC while Ground pin is connected to the common ground. And A0 pin connected to the A0 pin Of MQ2 sensor.

As red led runs on 2.0 volt and the green led runs on 2.2 volt ,we need to reduce the voltage of the LED's .So we grounded two LED's using two 100 ohm resistors. When the condition became true the red led illuminate. And in normal condition the red led turns off and the green led turns on.

We connected the buzzer to pin 8 of arduino so that when the threshold limit surpass the buzzer will start beeping.

The Arduino can be powered by connecting it to a USB connection. Since the voltage supply and ground pins of the other modules are connected with the common VCC and ground respectively, the rest of the components draw power from the 5V output of the Arduino board itself.

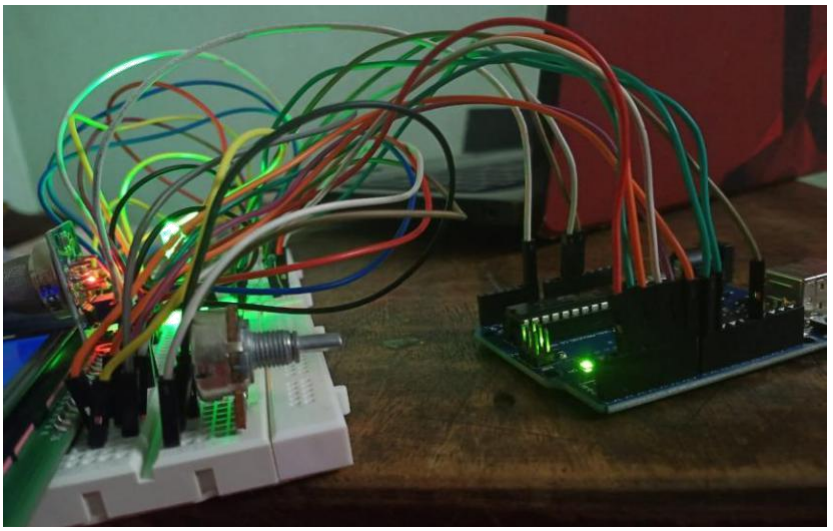


Fig-18

TESTING AND OUTPUT

Before-

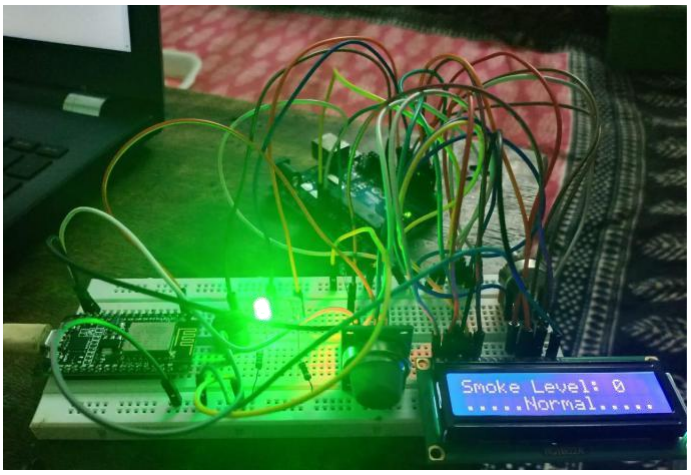
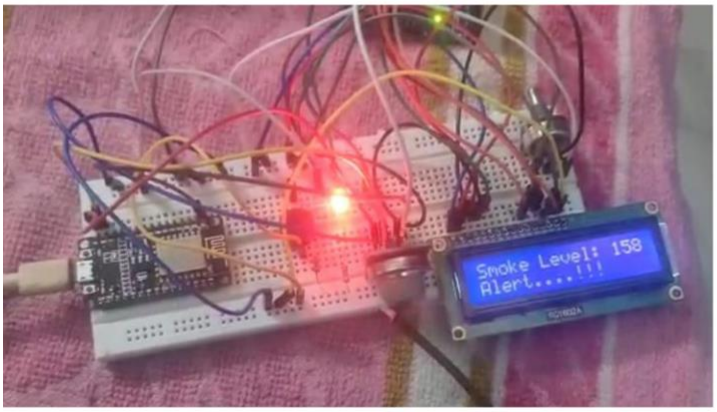


Fig-19

After-



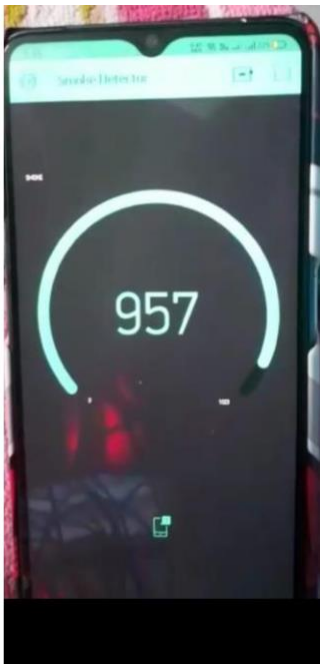
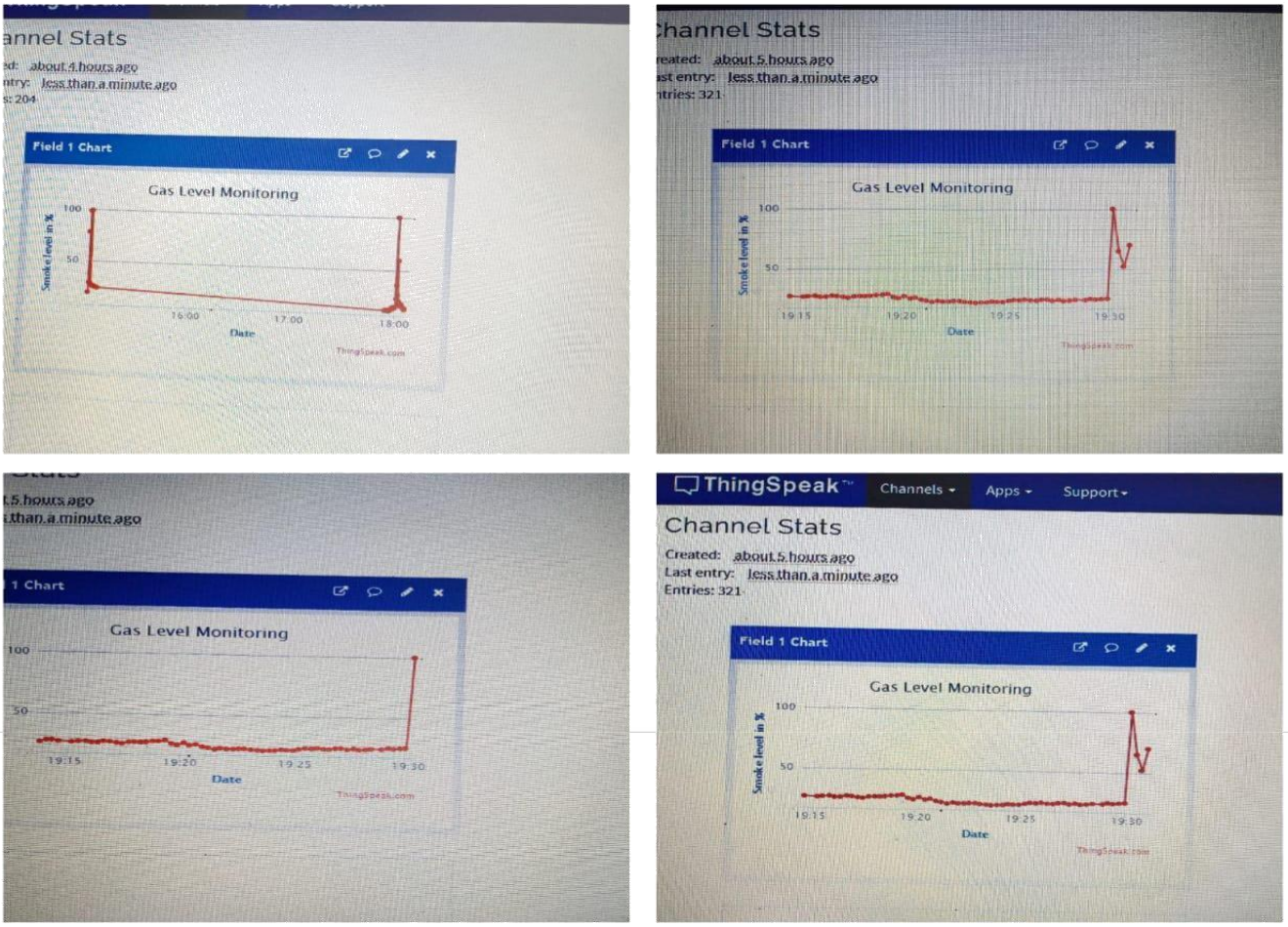
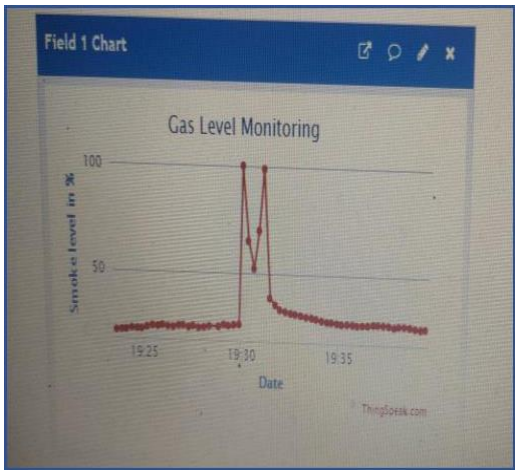


Fig-20





Various parameters being displayed on ThingSpeak.com

Fig-21

SOURCE CODE

Source Code for Arduino:-

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);


int sensorPin = A0; // choose the input pin (for GAS sensor)
int buzzer = 13; // choose the pin for the Buzzer int
G_led = 8; // choose the pin for the Green LED int
R_led = 9; // choose the pin for the Red Led


int read_value; // variable for reading the gaspin status
int set = 50; // we start, assuming Smoke detected


void setup()
{
        pinMode(sensorPin, INPUT); // declare sensor as input


        pinMode(buzzer,OUTPUT); // declare Buzzer as output
        pinMode(R_led,OUTPUT); // declare Red LED as output
        pinMode(G_led,OUTPUT); // declare Green LED as output


        lcd.begin(16, 2);
```

```
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("   WELCOME To   ");
        lcd.setCursor(0,1);
        lcd.print("  GAS Detector  ");
        delay(2000);
        lcd.clear();
}

void loop()
{
        read_value = (analogRead(sensorPin)); // read input value
        read_value = read_value - 500;
        if(read_value<0)
        {
                read_value=0;
        }
        lcd.setCursor(0, 0);
        lcd.print("Smoke Level: ");
        lcd.print(read_value);
        lcd.print("  ");
        if(read_value>set)
        {
```

```
            // check if the Smoke variable is High
            lcd.setCursor(0, 1); lcd.print("Alert....!!! ");
            digitalWrite(buzzer, HIGH); // Turn LED
            on. digitalWrite(R_led, HIGH); // Turn LED
            on. digitalWrite(G_led, LOW); // Turn LED
            off. delay(1000);

        }
        if(read_value<set)
        {
            // check if the Smoke variable is Low
            lcd.setCursor(0, 1);
            lcd.print(".....Normal.....");
            digitalWrite(buzzer, LOW); // Turn LED on.
            digitalWrite(R_led, LOW); // Turn LED on.
            digitalWrite(G_led, HIGH); // Turn LED on.
        }
        delay(1000);
}
```

Source Code Explanation for Arduino:-

- LiquidCrystal Library allows an Arduino board to control LiquidCrystal displays (LCDs) based on the Hitachi HD44780 (or a compatible) chipset, which is found on most text-based LCDs. The library works with in either 4- or 8-bit mode (i.e. using 4 or 8 data lines in addition to the rs, enable, and, optionally, the rw control lines).

  To use this library

  #include <LiquidCrystal.h>

- LiquidCrystal() creates a variable of type LiquidCrystal. The display can be controlled using 4 or 8 data lines. If the former, omit the pin numbers for d0 to d3 and leave those lines unconnected. The RW pin can be tied to ground instead of connected to a pin on the Arduino; if so, omit it from this function's parameters.

  LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

  Here , `lcd` : a variable of type LiquidCrystal.

- A variable is a way of naming and storing a value for later use by the program, such as data from a sensor or an intermediate value used in a calculation. Here we have declared six variables. First is sensorPin to connect the input pin(A0) for GAS sensor, second is buzzer to connect the pin 13 for the Buzzer, Third is G_led to connect the pin 8 for Green LED, fourth is R_led to connect the pin 9 for Red LED, fifth is read_value for reading the gas pin status and the last is set to set the threshold level.

```
int sensorPin = A0;
int buzzer = 13;
int G_led = 8;
int R_led = 9;
int read_value;
int set = 50;
```

- The setup() function is called when a sketch starts. Use it to initialize variables, pin modes, start using libraries, etc. The setup() function will only run once, after each powerup or reset of the Arduino board. And pinMode() Configures the specified pin to behave either as an input or an output. Here we set sensorPin as input and buzzer, R_led, G_led as output.

```
pinMode(sensorPin, INPUT);

pinMode(buzzer,OUTPUT);

pinMode(R_led,OUTPUT);

pinMode(G_led,OUTPUT);
```

- begin() Initializes the interface to the LCD screen, and specifies the dimensions (width and height) of the display. begin() needs to be called before any other LCD library commands.

```
lcd.begin(16, 2);
```

- clear() clears the LCD screen and positions the cursor in the upper-left corner.

```
lcd.clear();
```

- setCursor() position the LCD cursor; that is, set the location at which subsequent text written to the LCD will be displayed. And print() prints text to the LCD. In our code at first we select the first row and print “WELCOME To “ and then select the second row and print “GAS Detector “ . This texts will show for 2sec and after that the LCD screen would be clear.

```
lcd.setCursor(0,0);
lcd.print("   WELCOME To   ");
lcd.setCursor(0,1);
lcd.print(" GAS Detector ");
delay(2000);
```

```
lcd.clear();
```

- After creating a setup() function, which initializes and sets the initial values, the loop() function does precisely what its name suggests, and loops consecutively, allowing program to change and respond. Use it to actively control the Arduino board.

  So, In loop() at first we will read the input value. And for adjustment the value we will subtract it by 500.

```
read_value = (analogRead(sensorPin)); // read input value

read_value = read_value - 500;
```

- Then we will check whether this value come in positive or not. If it is coming negative then we will set it by zero. And also show it on LCD.

```
if(read_value<0)
{
        read_value=0;
}
lcd.setCursor(0, 0);
lcd.print("Smoke Level: ");
lcd.print(read_value);
lcd.print(" ");
```

- If the value is positive then we will check whether it surpassed the threshold level or not. If it goes above threshold level then we will show “Alert…!!!” on LCD and also turn on the Buzzer and Red LED for danger indication. And turn off the Green LED.

```
if(read_value>set)
{
        lcd.setCursor(0, 1);
        lcd.print("Alert....!!!    ");
        digitalWrite(buzzer, HIGH);
        digitalWrite(R_led, HIGH);
```

```
            digitalWrite(G_led, LOW);
            delay(1000);
        }
```

- But if the value is not surpassed the threshold level then we will show "….Normal…."on the LCD. And also keep the Buzzer and Red LED off. And illuminate the Green LED.

```
        if(read_value<set)
        {
            lcd.setCursor(0, 1);
            lcd.print(".....Normal.....");
            digitalWrite(buzzer, LOW); // Turn LED on.
            digitalWrite(R_led, LOW);
            digitalWrite(G_led, HIGH); // Turn LED on.
        }
        delay(1000);
```

Source Code for ESP12 E:-

```
#include <ESP8266WiFi.h>
String apiKey = "3EUWYU9X997ZAZNC"; // Enter your Write API key from ThingSpeak
const char *ssid = "WI-FI id name"; // replace with your Wi-Fi ssid and wpa2 key
const char *pass = "password";
const char *server = "api.thingspeak.com";
WiFiClient client;

void setup()

{

    Serial.begin(115200);
    delay(10);
```

```
        Serial.println("Connecting to ");
        Serial.println(ssid);
        WiFi.begin(ssid, pass);
        while (WiFi.status() != WL_CONNECTED)
        {
                delay(500);
                Serial.print(".");
        }
        Serial.println("");
        Serial.println("Wi-Fi connected");
}
void loop()
{
        float h = analogRead(A0);
        if (isnan(h))
        {
                Serial.println("Failed to read from MQ-2
                sensor!"); return;
        }

        if (client.connect(server, 80)) // "184.106.153.149" or api.thingspeak.com
        {
                String postStr = apiKey;
                postStr += "&field1=";
                postStr += String(h/1023*100);
```

```
            postStr += "r\n";
            client.print("POST /update HTTP/1.1\n");
            client.print("Host: api.thingspeak.com\n");
            client.print("Connection: close\n");
            client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");
            client.print("Content-Type: application/x-www-form-urlencoded\n");
            client.print("Content-Length: ");
            client.print(postStr.length());
            client.print("\n\n");
            client.print(postStr);
            Serial.print("Smoke Level: ");
            Serial.println(h/1023*100);
            Serial.println("Data Send to Thingspeak");
        }
        delay(500);
        client.stop();
        Serial.println("Waiting...");
        // thingspeak needs minimum 15 sec delay between
        updates. delay(1500);
}
```

Source Code Explanation for ESP12 E:-

- In the first line of the code we have wrote **#include<ESP8266WIFI.h>.** By writing this we are including the **ESP8266WIFI** library. This library provides ESP8266 specific Wi-Fi routines that we are calling to connect to the network.

- After that we have declared some variables. In apiKey we assign the Write API key from ThingSpeak. And in *ssid and *pass we have assigned the Wi-Fi ssid and wpa2 key respectively. And these things are dependable to every router. In *server we assign ThingSpeak server address.

- By writing **WiFiClient client** we have initialized the client library.

- The **setup()** function is called when a sketch starts. Use it to initialize variables, pin modes, start using libraries, etc. The **setup()** function will only run once, after each powerup or reset of the Arduino board. And The **serial.begin( )** sets the baud rate for serial data communication. The baud rate signifies the data rate in bits per second. The default baud rate in Arduino is **9600 bps (bits per second**). But we have specified at 115200.

```
Serial.begin(115200);
```

- Serial.println() Prints data to the serial port as human-readable ASCII text. This command can take many forms. Numbers are printed using an ASCII character for each digit. Floats are similarly printed as ASCII digits, defaulting to two decimal places. Bytes are sent as a single character. Characters and strings are sent as is. So, after 10ms it will show “Connecting to _____”(Wi-Fi ssid name).

```
delay(10);
Serial.println("Connecting to ");
Serial.println(ssid);
```

- WiFi.begin() Initializes the Wi-Fi library's network settings and provides the current status.

```
WiFi.begin(ssid, pass);
```

- In below function it will show “….” in every 0.5sec until the Wi-Fi is connected and after connected it will show “Wi-Fi connected”.

```
while (WiFi.status() != WL_CONNECTED)
```

```
{
        delay(500);
        Serial.print(".");
}

Serial.println("");

Serial.println("Wi-Fi connected");
```

- In loop() we are taking a float variable h and assign the input value.

```
float h = analogRead(A0);
```

- Now we will check that whether this value is illegal number or not. If it is illegal number then it will show “Failed to read from MQ-2 sensor!”

```
if (isnan(h))

{

        Serial.println("Failed to read from MQ-2

        sensor!"); return;

}
```

- connect() connects to a specified IP address and port. The return value indicates success or failure. Also supports DNS lookups when using a domain name. After successful connection we will send the right value to the server in percentage form. After sending the data we will show “Data Send to ThingSpeak” and after 0.5sec we close the connection.

```
if (client.connect(server, 80))
{
        String postStr = apiKey;
        postStr += "&field1=";
        postStr += String(h/1023*100);
        postStr += "r\n";
        client.print("POST /update HTTP/1.1\n");
        client.print("Host: api.thingspeak.com\n");
        client.print("Connection: close\n");
        client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");
        client.print("Content-Type:application/x-www-form-
        urlencoded\n");
        client.print("Content-Length: ");
```

```
            client.print(postStr.length());
            client.print("\n\n");
            client.print(postStr);
            Serial.print("Smoke Level: ");
            Serial.println(h/1023*100);
            Serial.println("Data Send to Thingspeak");
        }

        delay(500);

        client.stop();
```

- As ThingsSpeak needs minimum 15 sec to update the value. We also put a delay().

```
            Serial.println("Waiting...");

            delay(1500);
```

Source Code for Blynk Application:-

```
#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>

BlynkTimer timer;

#define BLYNK_PRINT Serial    // Comment this out to disable prints and save space

char auth[] = "Pb8MfjEQrk201hU7cxRPZkhrykQLqlob"; //Enter Authentication code sent by Blynk on your regested email


char ssid[] = "Wi-Fi Name"; // Enter WIFI Name Here

char pass[] = "password"; // Enter WIFI Password Here


int mq2 = A0; // smoke sensor is connected with the analog pin A0

int data = 0;

void setup()
```

```
{
        Serial.begin(115200);
        Blynk.begin(auth, ssid, pass);
        timer.setInterval(1000L, getSendData);
}
void loop()
{
         timer.run(); // Initiates SimpleTimer
         Blynk.run();
}
void getSendData()
{
        data = analogRead(mq2);
         Blynk.virtualWrite(V2, data);
          if (data > 700 )
         {
                 Blynk.notify("Smoke Detected!");
          }
}
```

Source Code Explanation for Blynk Application :-

- First we need to download the Blynk app and the most recent Blynk library . As we have already included the **ESP8266WIFI** library.Now,to make the connection between blynk and wifi module we need to download the Blynk Esp8266 libraries.
- And after making the account the most important thing in Blynk known as “authentication token”. This is the token which sets the communication between the

app and the board.In the code we mentioned the authentication code. It's important to send data in intervals and keep the void loop() as clean as possible.Blynk timer allows you to send data periodically with given intervals not interfering with Blynk library routines.And we also assigned ssid and pass in our code to give the wifi access to our code.As the sensor is connected to the analog pin A0.We need to access this sensor detected data to the Blynk app ,so we assigned the value to the MQ-2.

```
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
BlynkTimer timer;
#define BLYNK_PRINT Serial
 char auth[] = "Pb8MfjEQrk201hU7cxRPZkhrykQLqlob";


char ssid[] = "TP-Link_19C6";
char pass[] = "password";
int mq2 = A0;
int data = 0;
```

- Now for initializing the libraries we used setup().The **serial.begin( )** sets the baud rate for serial data communication. Blynk.begin() Initializes the Blynk library's network settings and provides the current status.And we also set the interval through timer.setInterval.In this function we passed the getSendData function.

```
void setup()
{
        Serial.begin(115200);
        Blynk.begin(auth, ssid, pass);
        timer.setInterval(1000L, getSendData);
}
```

- To initiate the program we need to call Blynk.run() and timer.run() to Initiates SimpleTimer.

```
void loop()
{
        timer.run();
        Blynk.run();
}
```

- So, in this function, it reads the input value. Virtual Pins is a way to exchange any data between your hardware and Blynk app.So,the device can send data to the App using Blynk.virtualWrite(pin, value) and if the data is greater than 700 then we will get notified as Smoke Detected!

```
void getSendData()
{
        data = analogRead(mq2);
        Blynk.virtualWrite(V2, data);
        if (data > 700 )
        {
            Blynk.notify("Smoke Detected!");
        }
}
```

APPLICATION

1) Industrial perimeter monitoring

2) Indoor air quality monitoring.

3) Site selection for reference monitoring stations.

4) To make this data available to the common man.

5) Roadside pollution monitoring.

ADVANTAGES

1) Easy to Install.

2) Sensors are easily available.

3) Simple, compact and easy to handle.

4) Updates On mobile phone directly.

5) Accurate Pollution monitoring.

6) Remote location monitoring.

7) Continuous update of change in percentage of quality.

FUTURE SCOPE

In future the project can be upgraded in more ways.

- Interface more number of sensors to know detail content of all gases present in air.
- Design webpage and upload data on webpage with date and time.
- Interface SD Card to store data.
- Interface GPS module to monitor the pollution at exact location and upload on the webpage for the netizens.

CONCLUSION

An air quality detector is very important because nowadays air pollution is easy to find. For the air pollution which cannot be easy detected by human, it requires a device as a reader of the air quality. By this proposed project, we can avoid air pollution through monitoring the air quality regularly. The system to monitor the air of environment using Arduino microcontroller, IOT Technology is proposed to improve quality of air. With the use of IOT technology enhances the process of monitoring various aspects of environment such as air quality monitoring issue proposed in this paper. Here, using the MQ2 gas sensor gives the sense of different type of dangerous gas and Arduino is the brain of this project which control the entire process. Wi-Fi module (ESP 12E) is the heart of this project which connects the whole process to internet and LCD is used for the visual Output.

REFERENCE

- https://en.wikipedia.org/wiki/ESP8266
- https://create.arduino.cc/projecthub/adithya-tg/control-arduino-uno-using-esp8266-wifi-module-and-blynk-app-504494
- https://www.arduino.cc
- https://roboindia.com/tutorials/blynk-introduction-nodemcu/
- https://whatis.techtarget.com/definition/LCD-liquid-crystal-display
- https://lastminuteengineers.com/mq2-gas-senser-arduino-tutorial/amp/
- https://components101.com/wireless/esp12e-pinout-datasheet