Makhanlal Chaturvedi National University of Journalism and Communication Bhopal

SESSION 2022 - 2024

PRCTICAL FILE

INTERNET OF THINGS (IOT)

MASTER'S IN COMPUTER APPLICATION

SEMESTER -3rd

SUBMITTED TO:	Mr. Praveen (Sir)
SUBMITTED BY:	Mohd Soyal
ENROLLMENT NO. :	AY1020999037
SUBMISSION DATE:	11/12/2023

SIGNATURE
REMARK

INTERNET OF THINGS - PRACTICAL INDEX

QUESTION 1.

- **Gas detector using node MCU.**

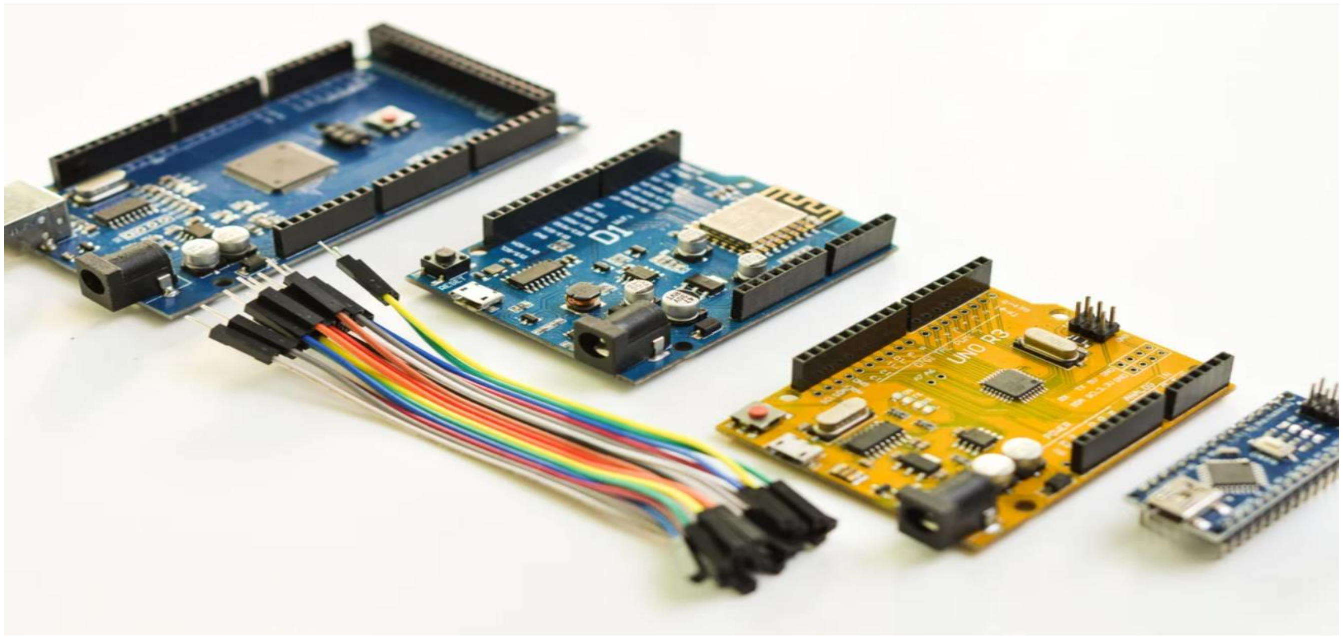
Gas detector using node MCU

- ➢ Gas detector using node MCU:- Here we use an MQ-2 gas sensor that detects the LPG, gases. If there is a gas in the air the resistance between two electrodes of the gas sensor gets increased according to the gas presence. We use node MCU as a microcontroller or a wifi development board that reads the analog values from the gas sensor. In the code, we set the gas value [200]. The node MCU sends these analog values to the Smartphone to the Blynk application. when the gas value gets increased above 200 the app shows the notification "Gas Detected".

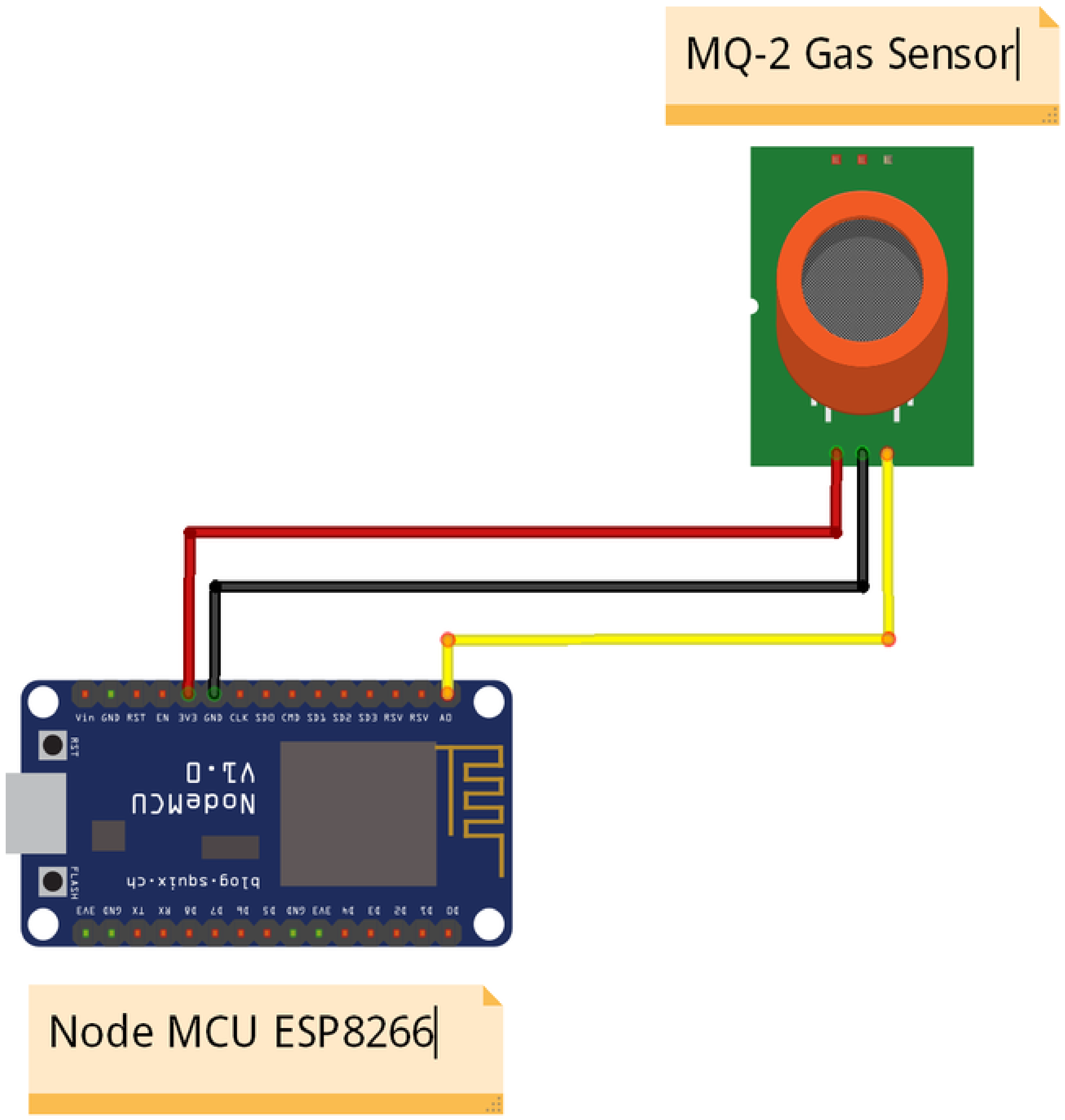
- **Step 1: MATERIAL REQUIRED**

➢ To make this project we need some components:-

1. NodeMcu ESP8266
2. MQ-2 Gas Sensor
3. Breadboard
4. Jumper Wires


D1
UNO R3

Step 2: CIRCUIT DIAGRAM



- Refer to the above circuit diagram to make a connection.

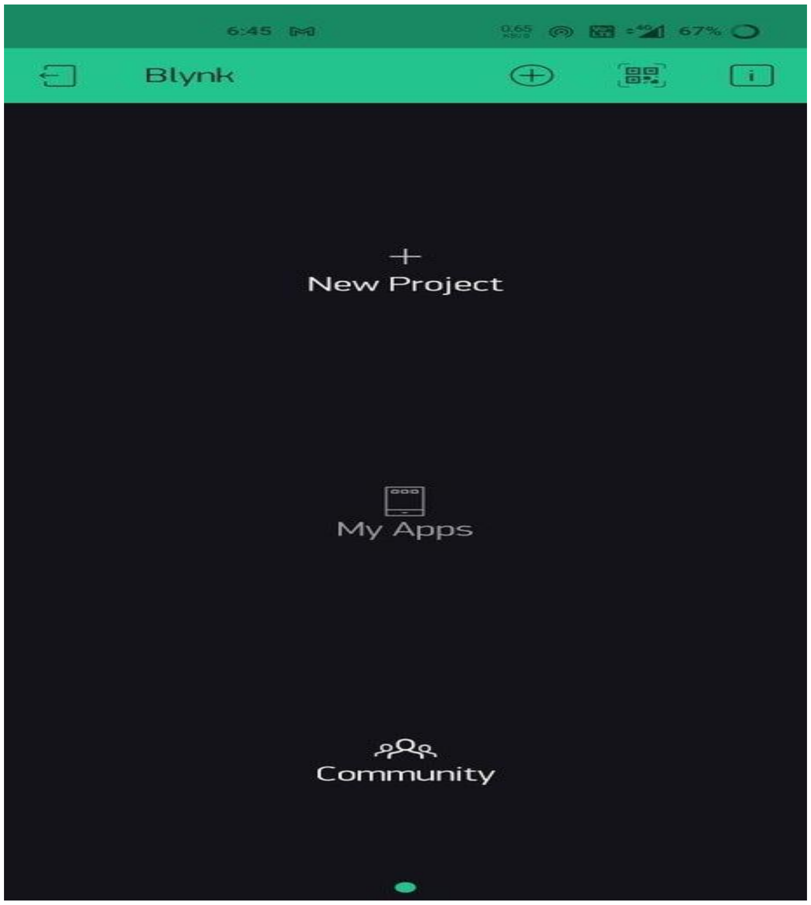
NodeMcu ---- MQ-2 Sensor

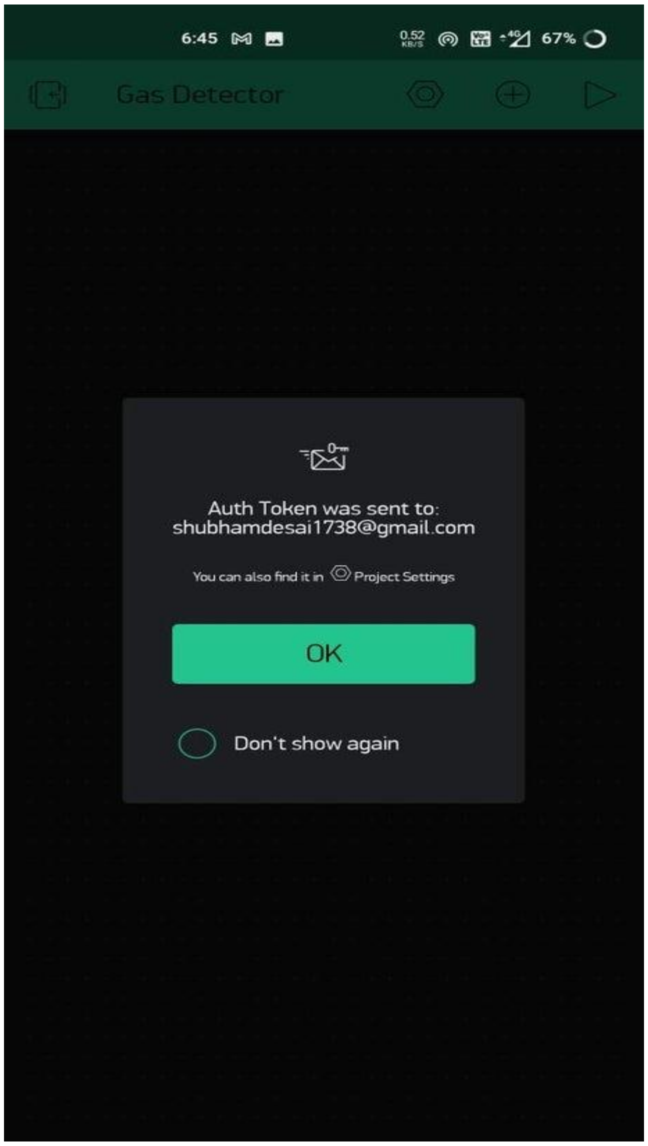
A0 >> A0 (Analog Pin)

G >> GND
3v3 >> VCC
Next PCB
NextPCB is a high-quality PCB Manufacturer. With professional PCB manufacturing capabilities, our PCB engineers with more than 10 years of experience will double-check your engineering files.

NextPCB is certified by IATF16949, ISO9001, ISO14001, UL, CQC, RoHS and REACH; more importantly, we handle the whole process including the PCB prototype, PCB manufacturing, PCB assembly, testing, and final shipment. We are capable of assembling BGA, Micro-BGA, QFN, and other leadless package parts. We also have an online parts shop, you can choose any parts you need.

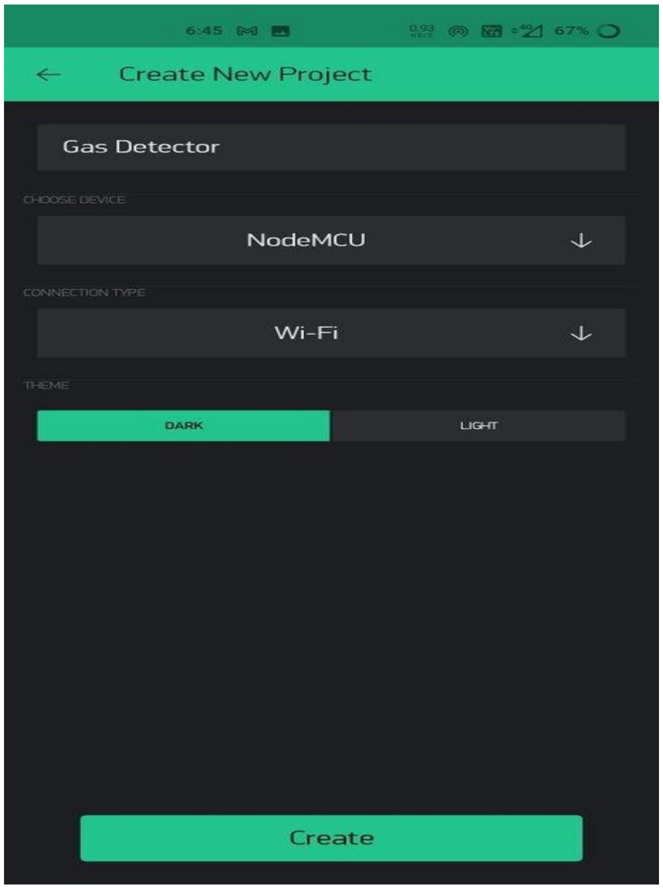
- ## Step 3: BLYNK APP INSTALLATION & SETUP:-





Auth Token was sent to:
shubhamdesai1738@gmail.com

You can also find it in Project Settings



Create New Project

Gas Detector

CHOOSE DEVICE

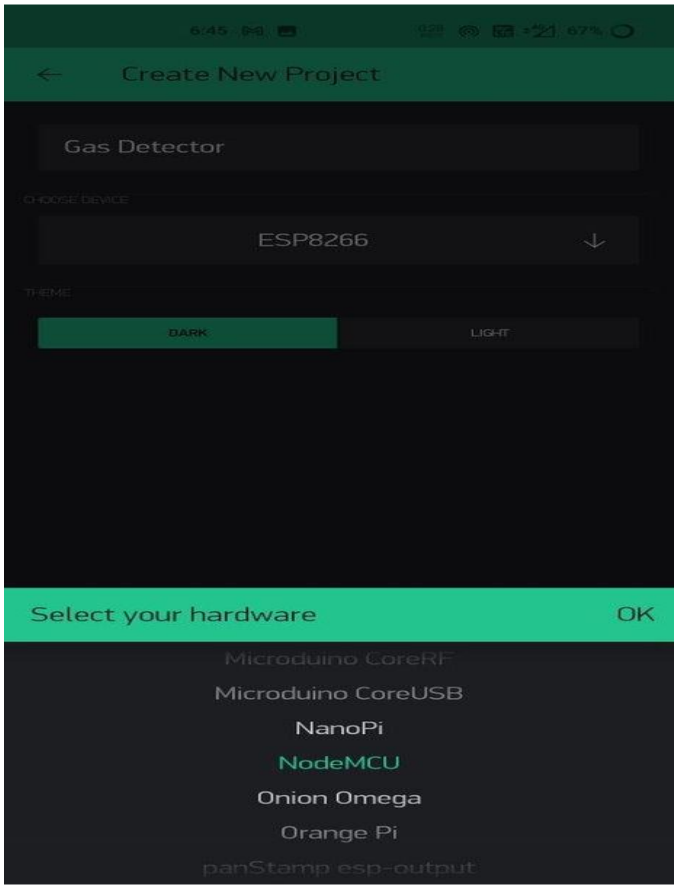
NodeMCU

CONNECTION TYPE

Wi-Fi

THEME

DARK LIGHT


Create New Project

Gas Detector

CHOOSE DEVICE

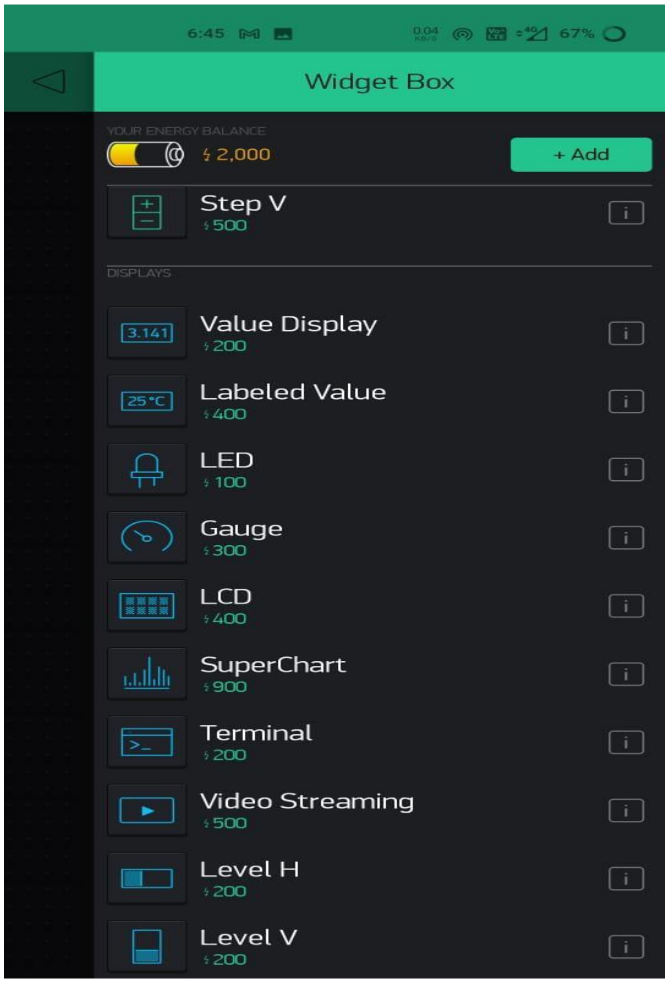
ESP8266

THEME

DARK LIGHT

Select your hardware OK

Microduino CoreRF
Microduino CoreUSB
NanoPi
NodeMCU
Onion Omega
Orange Pi
panStamp esp-output



6:45 0.04 KB/S 67%

Widget Box

YOUR ENERGY BALANCE

2,000

+ Add

Step V
500

DISPLAYS

Value Display
200

Labeled Value
400

LED
100

Gauge
300

LCD
400

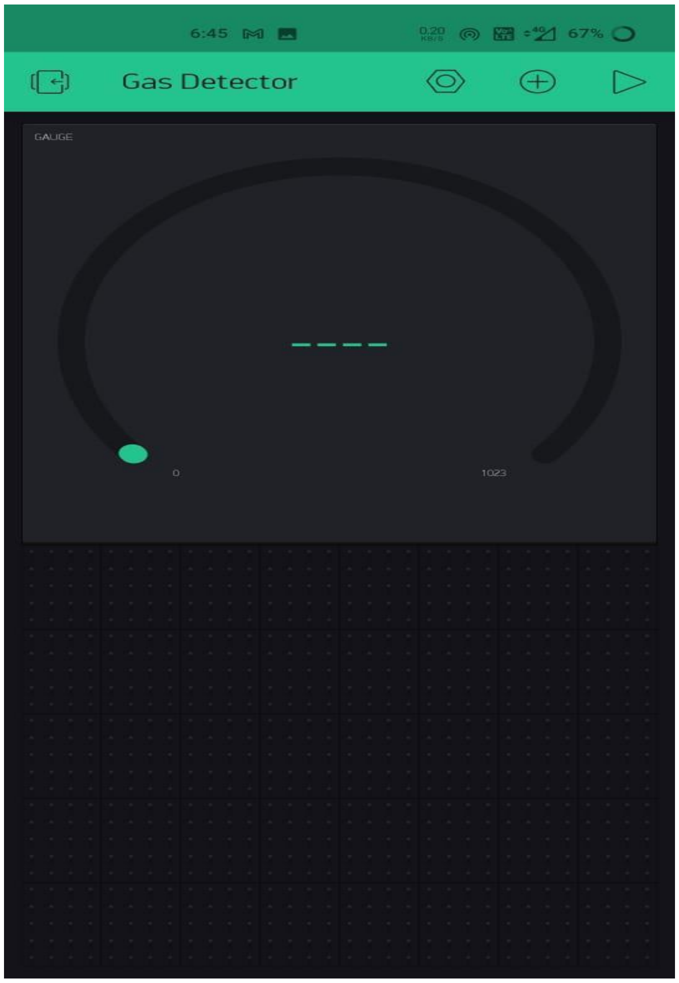
SuperChart
900

Terminal
200

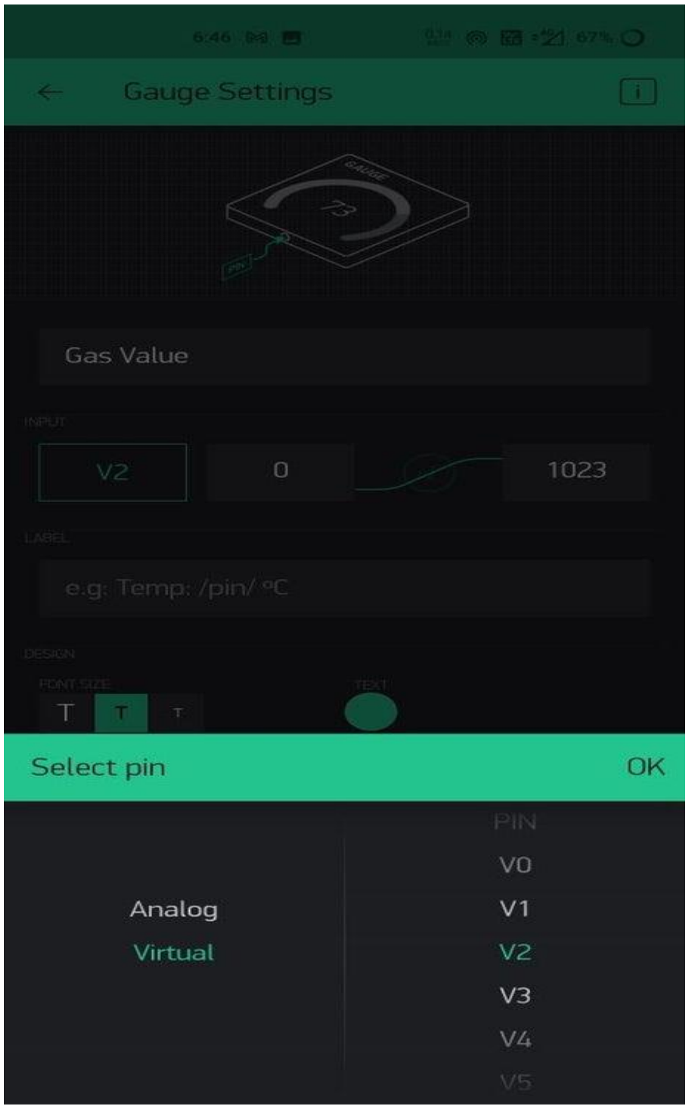
Video Streaming
500

Level H
200

Level V
200


6:45
67%
Gas Detector
GAUGE

0
1023



Gauge Settings

Gas Value

INPUT

V2 | 0 | 1023

LABEL

e.g: Temp: /pin/ ºC

DESIGN

FONT SIZE

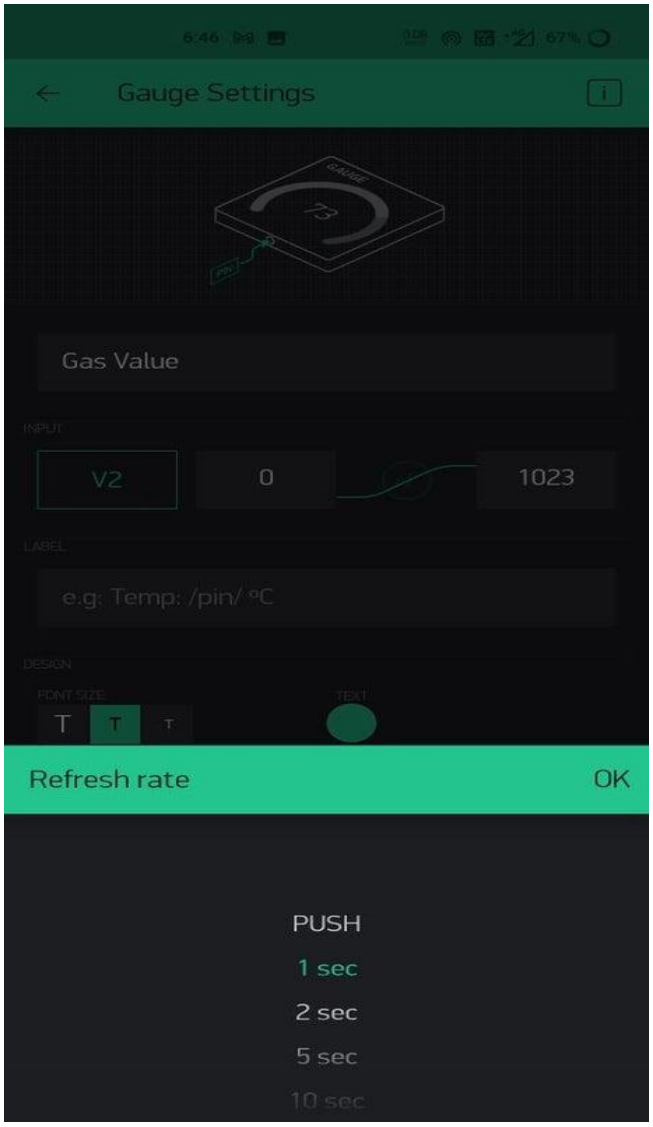
TEXT

Select pin OK

PIN

Analog
Virtual

V0
V1
V2
V3
V4
V5



Gauge Settings

Gas Value

INPUT

V2 0 1023

LABEL

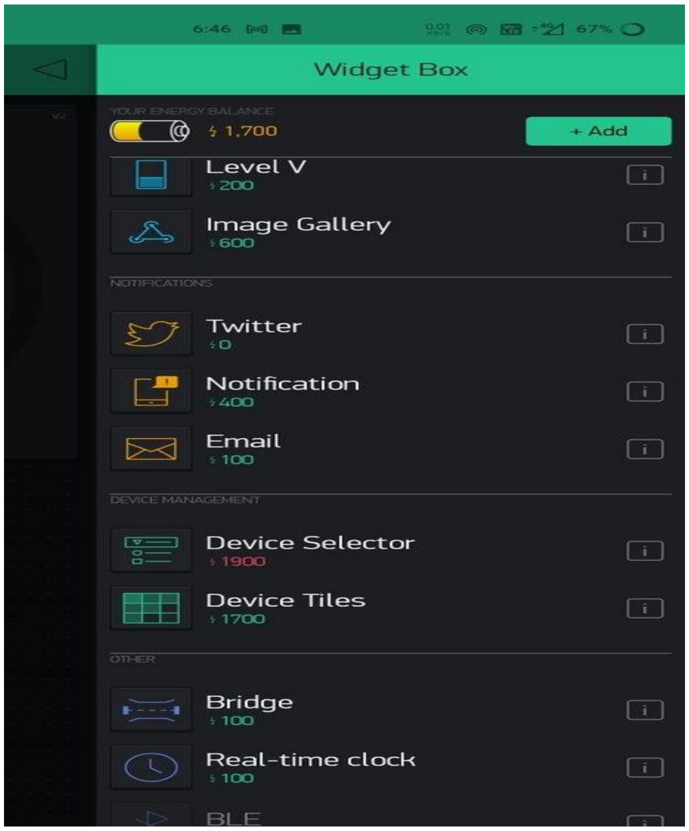
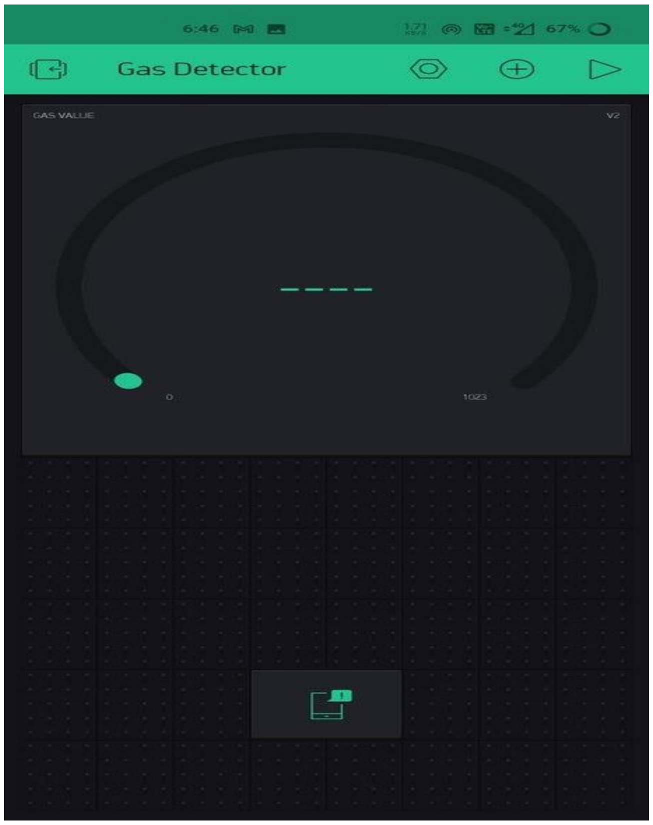
e.g: Temp: /pin/ ºC

DESIGN

FONT SIZE

TEXT

Refresh rate OK


6:46
Widget Box
YOUR ENERGY BALANCE
1,700
+ Add
Level V
200
Image Gallery
600
NOTIFICATIONS
Twitter
0
Notification
400
Email
100
DEVICE MANAGEMENT
Device Selector
1900
Device Tiles
1700
OTHER
Bridge
100
Real-time clock
100
BLE

6:46
Gas Detector
GAS VALUE
V2

0
1023

➢ Let's Setup the BLYNK app:-

1. Install Blynk App from google play store.
2. Create Account On Blynk.
3. Create New Project.
4. You will get Token on your E-Mail.
5. Give Name to Project.
6. Select Hardware (NodeMcu) & Connection Type (WIFI)& click on Create Button.
7. Add gauge using Widget Box.
8. Gave name to the Gauge as "Gas Value", Select Pin > Virtual > V2, Push > 1Sec.
9. Add Notification using Widget box.
10. Now Application Setup is done.

• Step 4: ARDUINO CODE

1. Add Blynk library in your Arduino IDE as well as NodeMcu library.
2. Sketch-Include Library-Manage libraries-Type Blynk-Install [Same For another Libraries]
3. Copy Following code & Upload to Arduino Nano

```
/*
* Here is the arduino code for the Project
 * Add following libraries in your Arduino IDE
 * Here is the complete Project making video :-
https://youtu.be/m2QufB-bap8
 */
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
BlynkTimer timer;
#define BLYNK_PRINT Serial    // Comment this out to disable
prints and save space
char auth[] = "Auth Token"; //Enter Authentication code sent
by Blynk on your regested email


char ssid[] = "----------"; // Enter WIFI Name Here
char pass[] = "----------"; // 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 > 200 )
  {
    Blynk.notify("Smoke Detected!");
  }

}
```

Step 5: WORKING & TESTING

QUESTION 2.

- Fire detector using node MCU.

Fire detector using node MCU

Fire alarm systems are very common nowadays and commonly installed in Banks, shops, offices, home etc. They detect the fire and trigger a loud alarm to aware everybody. But what if nobody is there to hear that alarm, like in night time or when nobody is at home. So to inform the authority about any fire incident today we are building a IoT based Fire Alarm system which not only trigger an alarm but also sends a Email alert to concern persons. This method can also be used to inform fire department automatically in case of fire. Here we will use Infrared Flame Sensor to detect the fire and ESP8266 NodeMCU to trigger the alarm and send email with the help of SMTP server. This project can be further extended to make a phone call or send an SMS with the help of GSM module in case of fire.

- **Step1 : COMPONENTS REQUIRED**

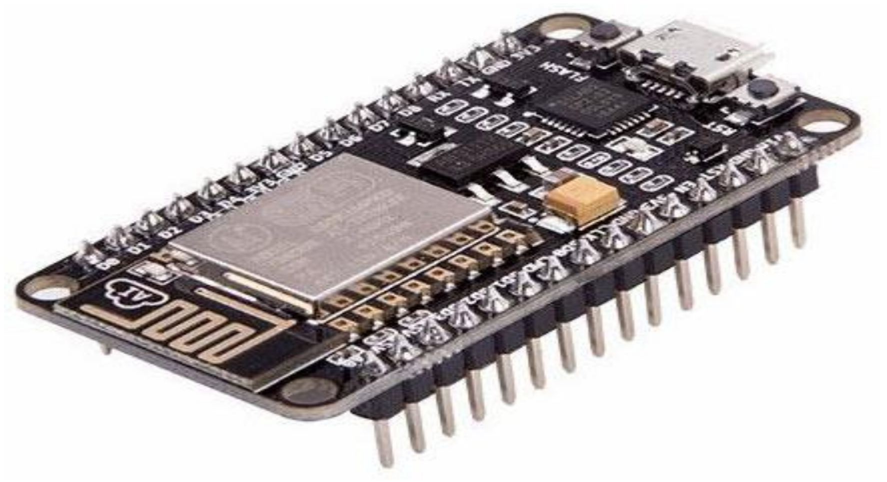
➢ To make this project we need some components:-

1. ESP8266 NodeMCU x 1
2. Flame Sensor x 1
3. MQ2 Smoke Sensor x 1
4. 5V Buzzer x 1

5. Led Any color x 1
6. Transistor BC547 x 1
7. Resistor 1KΩ x 2
8. Jumper cables
9. Breadboard

➢ NodeMCU ESP8266

ESP8266 NodeMCU is an open source IoT platform. It includes firmware which runs on the low cost Wi-Fi enabled ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. It has GPIO, SPI, I2C, ADC, PWM AND UART pins for communication and controlling other peripherals attached to it. On board NodeMCU has CP2102 IC which provides USB to TTL functionality.
In this IoT Fire Alarm, we are using one GPIO pin to get the digital data from the flame sensor.



➢ Flame Sensor

A Flame Sensor is a device that can be used to detect presence of a fire source or any other bright light sources. There are

several ways to implement a Flame Sensor but the module used in this project is an Infrared Radiation Sensitive Sensor.
The module uses a LM393 comparator chip to provide a stable digital output signal. This comparator has a driving ability of 15 mA. This flame detector sensor can be used in different project including fire alarms and other fire detecting devices or projects.
Specifications:

1. LM393 comparator chip
2. Detection Range: 760 nm to 1100 nm
3. Operating Voltage: 3.3 V to 5 V
4. Maximum Output Current: 15 mA
5. Digital Outputs: 0 and 1Detection Angle: about 60 degrees
6. Adjustable sensitivity via potentiometer
7. LED lights indicators: power (red) and digital switching output (green).

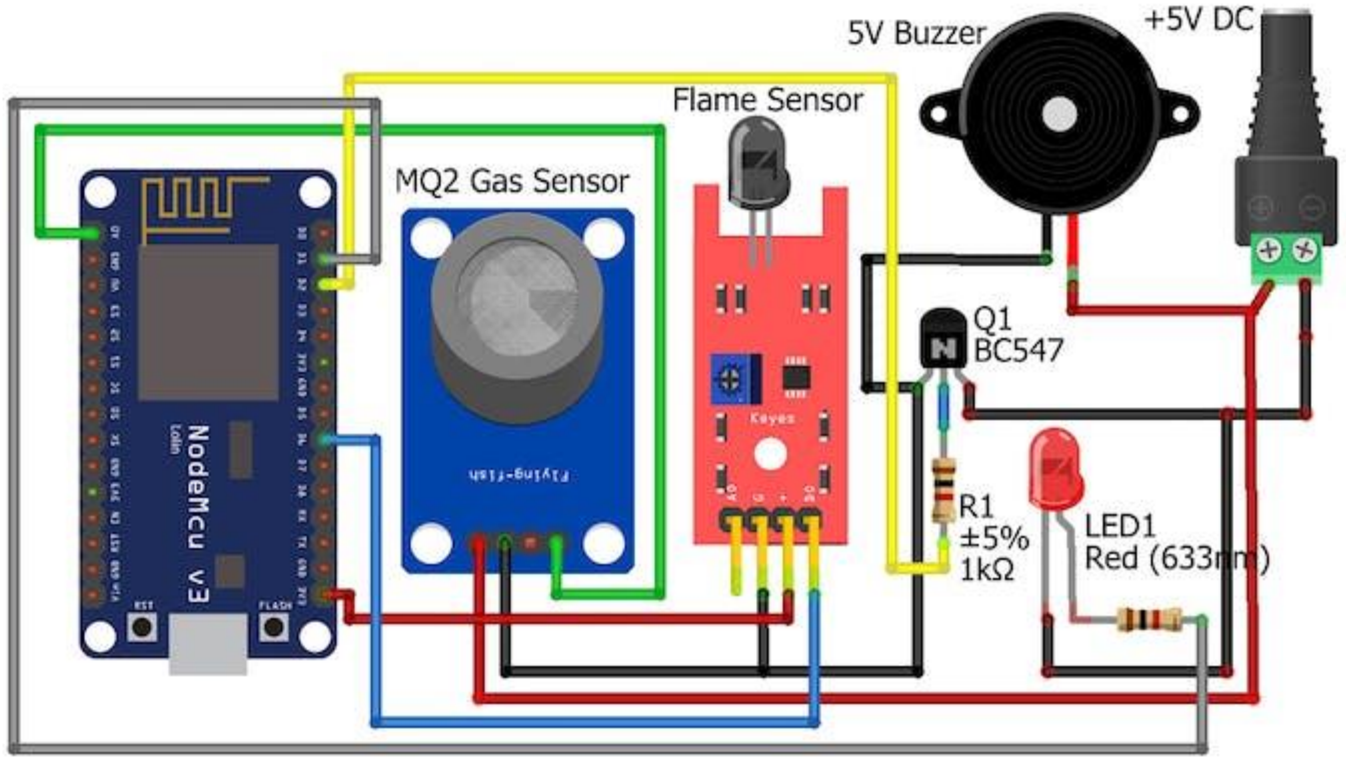
Step 2: Smoke Sensor

In this project we are using a MQ2 smoke sensor.
This sensor can detect various gases including Methane, Butane and LPG. We are using a MQ2 gas sensor module in this project for better interface.

- Specifications:

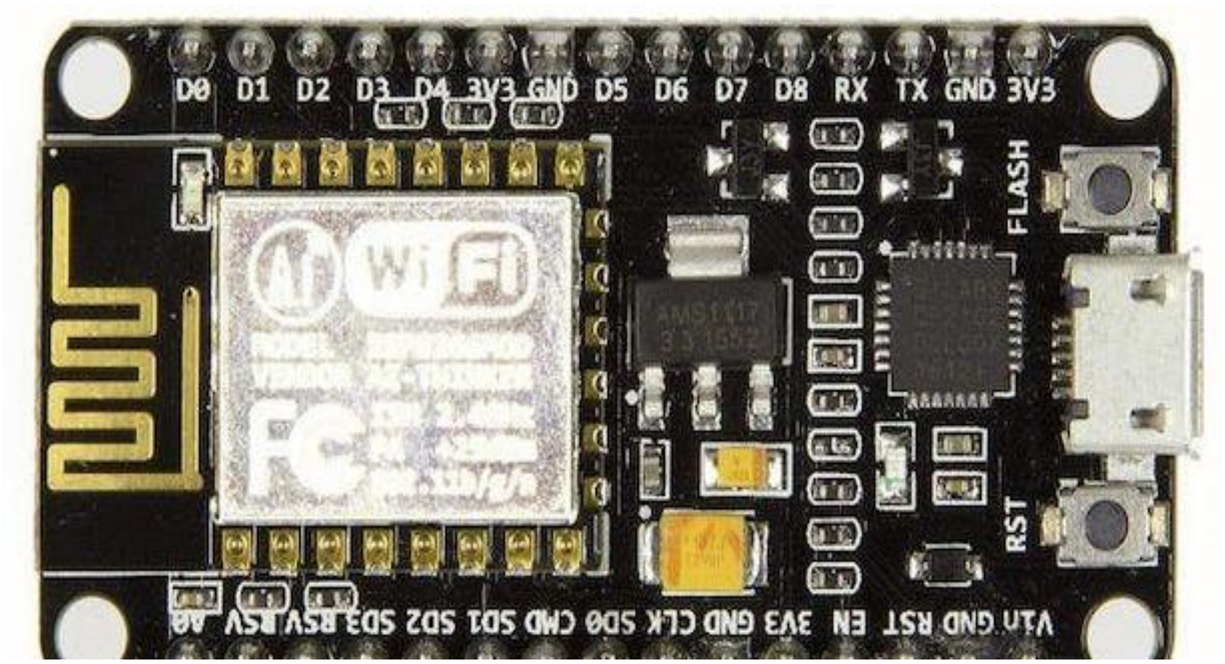
1. Operating Voltage is +5V
2. Can be used to Measure or detect LPG, Alcohol, Propane, Hydrogen, CO and even methane
3. Analog output voltage: 0V to 5V
4. Digital Output Voltage: 0V or 5V (TTL Logic)
5. Preheat duration 20 seconds
6. Can be used as a Digital or analog sensor
7. The Sensitivity of Digital pin can be varied using the potentiometer

Step 3: Connection



- The connection is very simple here all the red and black wired are Vcc and GND connection.
- You can also add an additional relay module to switch on exhaust or any other ac appliance.
- You can modify the code and add it. We are using A0 pin of MQ2 gas sensor and D0 pin of Flame sensor.

- **Step 4: NodeMCU ESP8266**



➢ ESP8266 NodeMCU is an open source IoT platform.

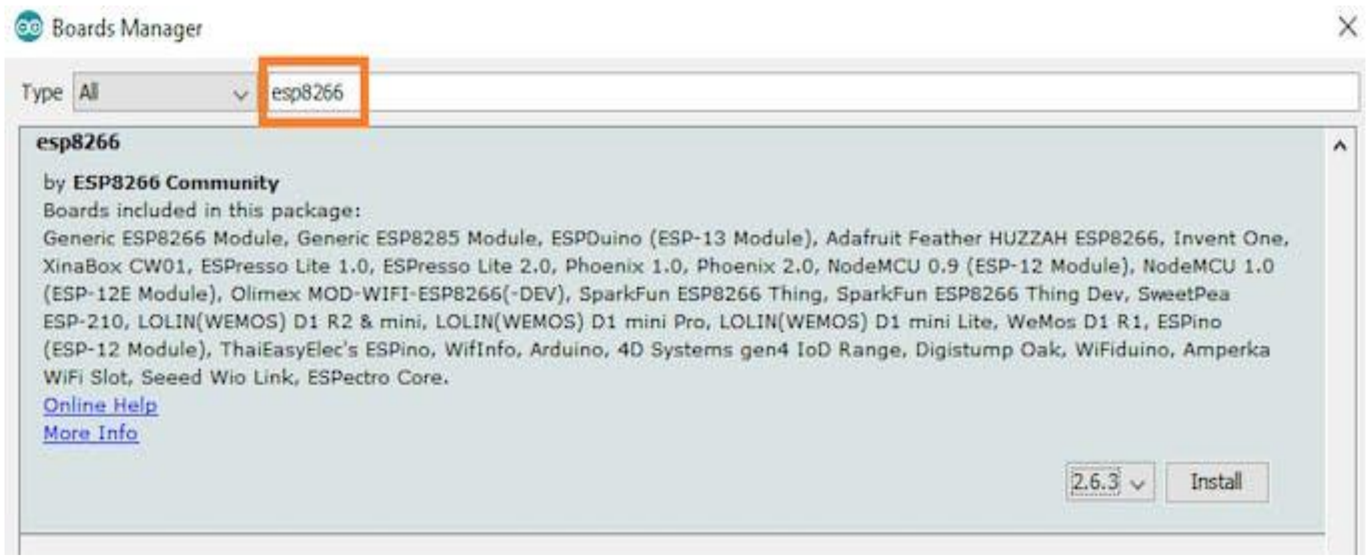
➢ It includes firmware which runs on the low cost Wi-Fi enabled ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module.

➢ It has GPIO, SPI, I2C, ADC, PWM AND UART pins for communication and controlling other peripherals attached to it.

➢ On board NodeMCU has CP2102 IC which provides USB to TTL functionality.

➢ In this IoT Fire Alarm, we are using two GPIO pin to get the digital data from the flame sensor and gas sensor.

Step 5: Required Arduino Libraries



esp8266
by **ESP8266 Community**
Boards included in this package:
Generic ESP8266 Module, Generic ESP8285 Module, ESPDuino (ESP-13 Module), Adafruit Feather HUZZAH ESP8266, Invent One, XinaBox CW01, ESPresso Lite 1.0, ESPresso Lite 2.0, Phoenix 1.0, Phoenix 2.0, NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Olimex MOD-WIFI-ESP8266(-DEV), SparkFun ESP8266 Thing, SparkFun ESP8266 Thing Dev, SweetPea ESP-210, LOLIN(WEMOS) D1 R2 & mini, LOLIN(WEMOS) D1 mini Pro, LOLIN(WEMOS) D1 mini Lite, WeMos D1 R1, ESPino (ESP-12 Module), ThaiEasyElec's ESPino, WifInfo, Arduino, 4D Systems gen4 IoD Range, Digistump Oak, WiFiduino, Amperka WiFi Slot, Seeed Wio Link, ESPectro Core.

➢ This code need 2 Arduino library. You can download Arduino IDE.

1. ESP8266
2. Blynk

➢ Go to Sketch -> Include Library -> Manage Libraries

➢ Now search for ESP8266 and install

➢ Search for Blynk and insta.

- **Step 6: Code**

➢ Working and Code Explanation

Make the connections as given in the circuit diagram and copy the complete code given at the end of this tutorial. Then upload the code in NodeMCU using Arduino IDE. The flame sensor is connected at D0 pin to give the digital input to the NodeMCU and Buzzer is connected at D1 pin to get digital output from the NodeMCU. If fire is detected by the flame sensor then it gives "0" and NodeMCU turns on the buzzer and send the alert email to the person using SMTP2GO automatically.

As told earlier complete code is given at the end of this tutorial. Here we are explaining the working of code.
First include the required libraries and define variables for Wi-Fi SSID and password to initialize the Wi-Fi connection. Since we need to connect to SMTP server a variable is declared with address of the server. D0 is the pin on NodeMCU for digital input from flame sensor.

```
#include <ESP8266WiFi.h>
const char* ssid = "Enter your ssid";
const char* password = "enter your password";
char server[] = "mail.smtp2go.com";
const int flame = D0;
WiFiClient Client;              //define wifi client as client
```

- Following code is used to connect to Wi-Fi network and displaying the status messages on Serial Monitor.

```
Serial.print("Connecting To: ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi Connected.");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
```

➢ Here we are storing the digital input from the flame sensor in a variable. If the input is "0" (low) means the flame sensor has detected the fire and it sends the email using function sendEmail().

```
int t = digitalRead(flame);
 Serial.println(t);
  if (t==0) {
 digitalWrite(buzz,HIGH);
   sendEmail();
   Serial.print("Mail sent to:");
   Serial.println(" The recipient");
   Serial.println("");
 }
```

➢ **Client.connect()** is the function which takes SMTP Server and SMTP Port to connect to SMTP server. It gives "1" if connection establishes successfully and acknowledge the server with EHLO command.

```
if (Client.connect(server, 2525) == 1)      // connect to smtp
server with port address 2525
 {
   Serial.println(F("connected to server"));
 }
 else
 {
   Serial.println(F("connection failed"));
   return 0;
 }
Client.println("EHLO www.example.com");
```

➢ Now send the encoded user name and password to the server .The user name and the password from the SMTP2GO which have saved in a notepad file are encoded to base64 value. The procedure to encode into base64 value is already explained above.

```
Client.println("AUTH LOGIN");
Serial.println(F("Sending User"));
Client.println("c2VuZGVyQHh5ei5jb20= "); //base64, ASCII
encoded SMTP Username
Serial.println(F("Sending Password"));
Client.println("cGFzc3dvcmQ=");   //base64, ASCII encoded
SMTP Password
```

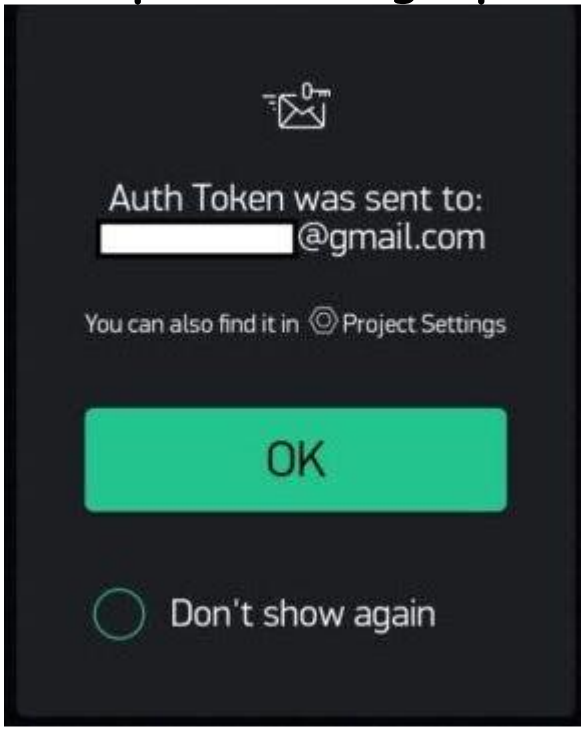
➢ After successful authentication, here the complete email is formed with fields like "To", "From", "Subject", "Body".

```
  Client.println(F("To:  receiver@xyz.com "));
  Client.println(F("From: sender@xyz.com "));
  Client.println(F("Subject: Fire Alarm\r\n"));
  Client.println(F("Attention: Fire Detected.\n"));
```

➢ After sending the data, QUIT command is sent to complete the email.

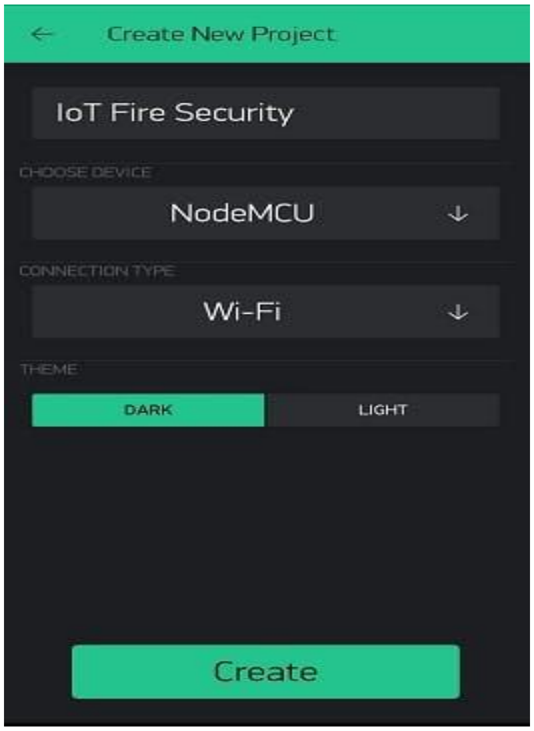
```
Serial.println(F("Sending QUIT"));
Client.println(F("QUIT"));
 if (!emailResp())
 return 0;
Client.stop();
```

- **Step 7: Setting Up a Blynk Project**



Auth Token was sent to:
[redacted]@gmail.com

You can also find it in Project Settings



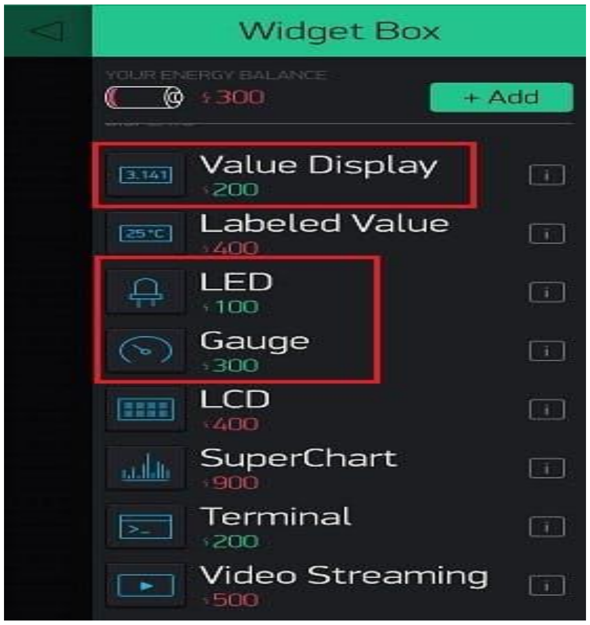
Create New Project

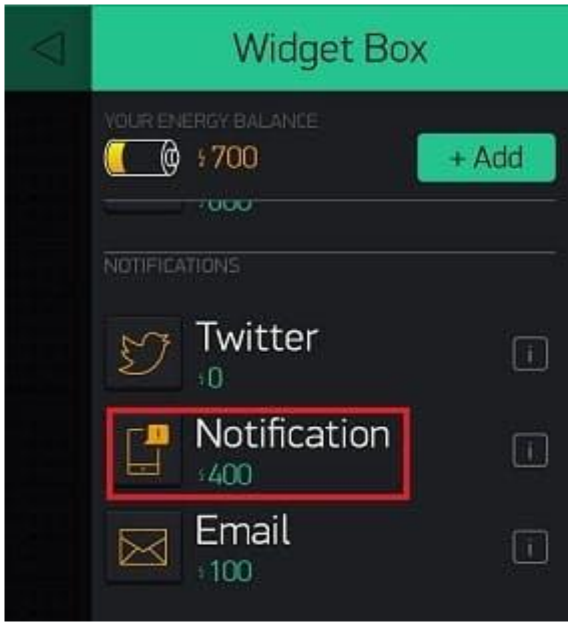
IoT Fire Security

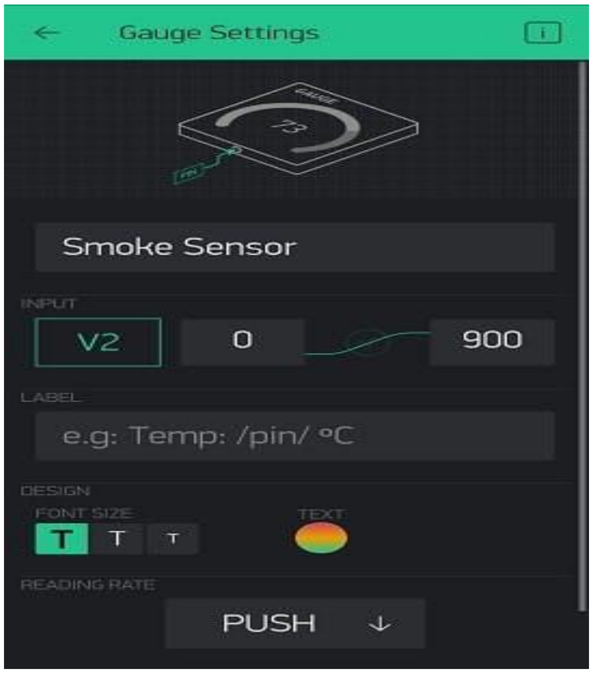
CHOOSE DEVICE: NodeMCU

CONNECTION TYPE: Wi-Fi

THEME: [✓ DARK] LIGHT


Widget Box
YOUR ENERGY BALANCE
300
+ Add
Value Display
200
Labeled Value
400
LED
100
Gauge
300
LCD
400
SuperChart
900
Terminal
200
Video Streaming
500


Widget Box
YOUR ENERGY BALANCE
700
+ Add
NOTIFICATIONS
Twitter
0
Notification
400
Email
100


Gauge Settings

Smoke Sensor

INPUT

V2 0 900

LABEL

e.g: Temp: /pin/ °C

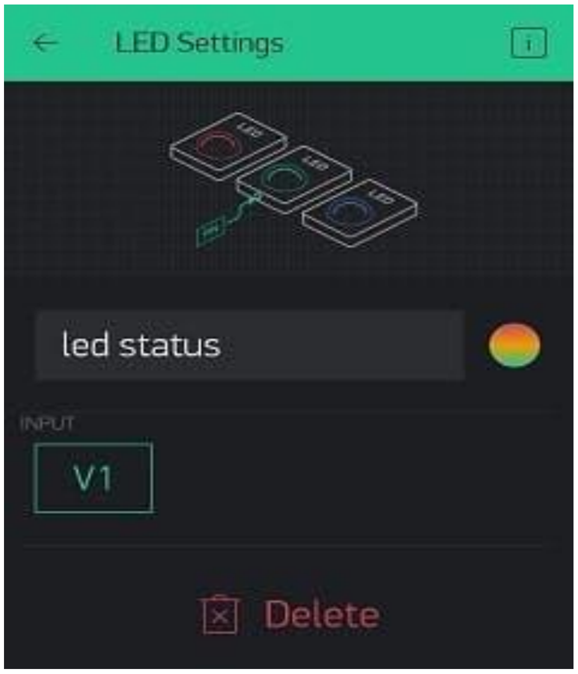
DESIGN

FONT SIZE

TEXT

READING RATE

PUSH

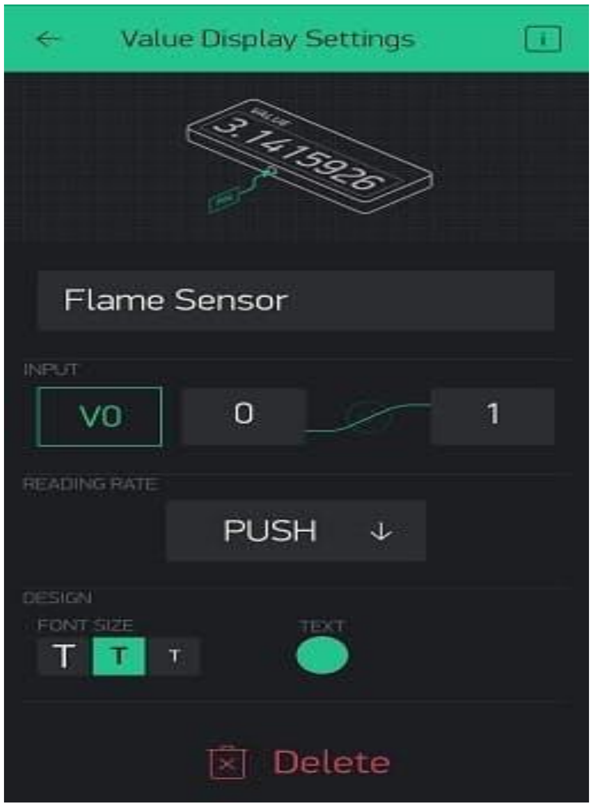

LED Settings

led status

INPUT

V1

Delete



Value Display Settings

Flame Sensor

INPUT

V0 0 1

READING RATE

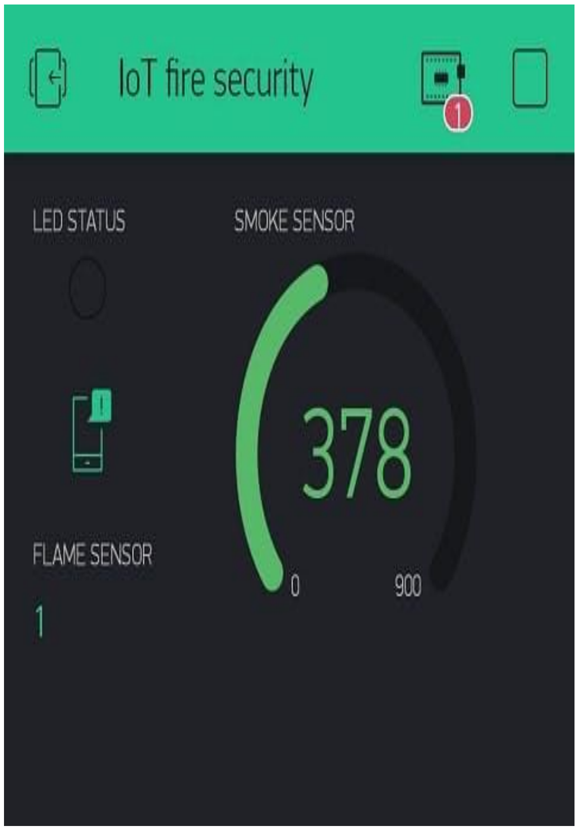
PUSH

DESIGN

FONT SIZE

TEXT

Delete



IoT fire security

LED STATUS

SMOKE SENSOR

378

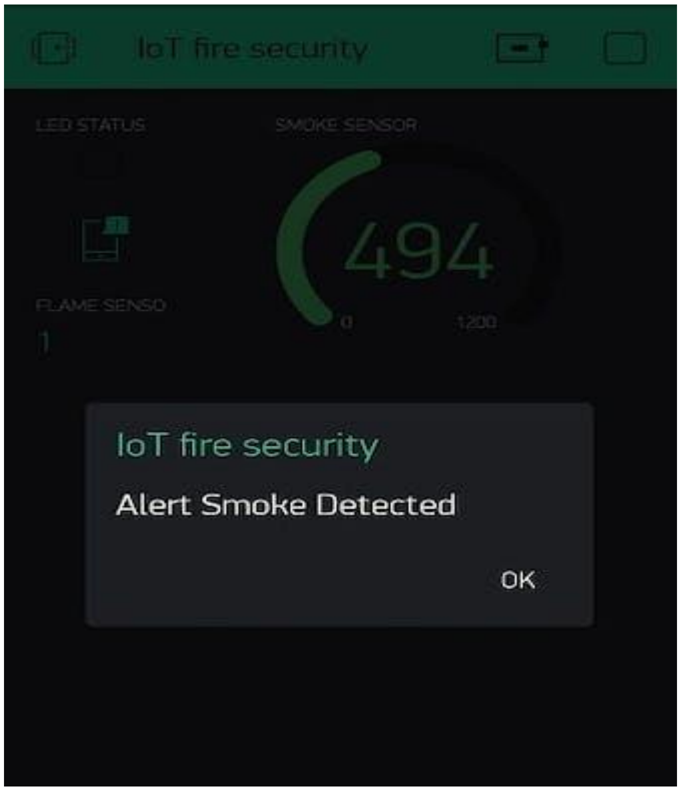
0 900

FLAME SENSOR

1

1. Open the blynk app and tap on create a new project. Fill the details as shown below
2. You will receive the Auth token on your registered email id.
3. Tap on + sign to add below highlighted widgets
4. Add a notification widget as well.
5. Configure the gauge with below details.
6. Also configure the LED
7. Set the Value Display widget as per below settings
8. And finally your Blynk project setup is complete. You can rearrange the widgets and place them as you wish.

- **Step 8: Building and Testing**



IoT fire security

Alert Smoke Detected

We have connected the components as per the circuit diagram and now its time to test the project.
For both the alert LED will glow and buzzer will start beeping.

- You will receive below alert notification when flame sensor detects flame or fire.
- Once the smoke sensor detects smoke and the value in the gauge goes beyond 500 then smoke alert will be detected. You can change the value in the code as per your need.

- **Coding of the circuit**

Here we use the Arduino platform to program our device and also we link it with Blynk app that we setup earlier.We add the library of node mcu and blynk app in it .

```
//Nextpcb
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
```

```
#include <BlynkSimpleEsp8266.h>

int led = 13; // Connected to D7 pin of NodeMCU
int flame_sensor = 12; //Connected to D6 pin of NodeMCU
int buzzer = 4; //Connected to D2 pin of NodeMCU
int relay = 5; //Connected to D1 pin of NodeMCU
int smoke_sensor = A0; //Connected to A0 pin of NodeMCU
int temp_smoke_sensor;
int flame_sensor_read;

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "Auth Token"; // Enter Auth Token

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "Your WiFi SSID"; //Enter the wifi name
char pass[] = "Your WiFi Password";// Enter the wifi password

void setup()
{
  pinMode(led, OUTPUT);
  pinMode(flame_sensor, INPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(relay, OUTPUT);
  pinMode(smoke_sensor, INPUT);
  digitalWrite(led, LOW);
  digitalWrite(buzzer, LOW);
  digitalWrite(relay, LOW);
  // Debug console
  Serial.begin(9600);
  Blynk.begin(auth, ssid, pass);
}
void loop()
{
  flame_sensor_read = digitalRead(flame_sensor); //reading the sensor data on A0
pin
  Blynk.virtualWrite(V0, flame_sensor_read); //sending data to Blynk
  Serial.print("Flame Status:");
  Serial.println(flame_sensor_read);
  int led_status = digitalRead(led);
  if (led_status == HIGH)
  {
   Blynk.virtualWrite(V1, 255);
  }
  else
  {
   Blynk.virtualWrite(V1, 0);
  }
  temp_smoke_sensor = analogRead(smoke_sensor);
  Blynk.virtualWrite(V2, temp_smoke_sensor);
  Serial.print("Current Gas Level:");
  Serial.println(temp_smoke_sensor);
  if (temp_smoke_sensor > 500)
  {
    digitalWrite(led, HIGH);
    digitalWrite(buzzer, HIGH);
    digitalWrite(relay, HIGH);
    Blynk.notify("Alert Smoke Detected");
    }
  else
    {
     digitalWrite(led, LOW);
     digitalWrite(buzzer, LOW);
     digitalWrite(relay, LOW);
    }
 if (flame_sensor_read == 0)
    {
    //Blynk.virtualWrite(V1, 255);
    digitalWrite(led, HIGH);
    digitalWrite(buzzer, HIGH);
    digitalWrite(relay, HIGH);
    Blynk.notify("Alert Fire Detected");
    }
  else
      {
```

```
    digitalWrite(led, LOW);
    digitalWrite(buzzer, LOW);
    digitalWrite(relay, LOW);
    }
    delay(500);
    Blynk.run();
}
```

Step 9: WORKING & TESTING

➢ Here we programmed our device and now we test the working of the device with different parameters.

<u>QUESTION 3.</u>

<u>QUES 3.</u>To interface OLED with Arduino/Raspberry Pi and write a program to print temperature and humidity readings on it.

Arduino – OLED-

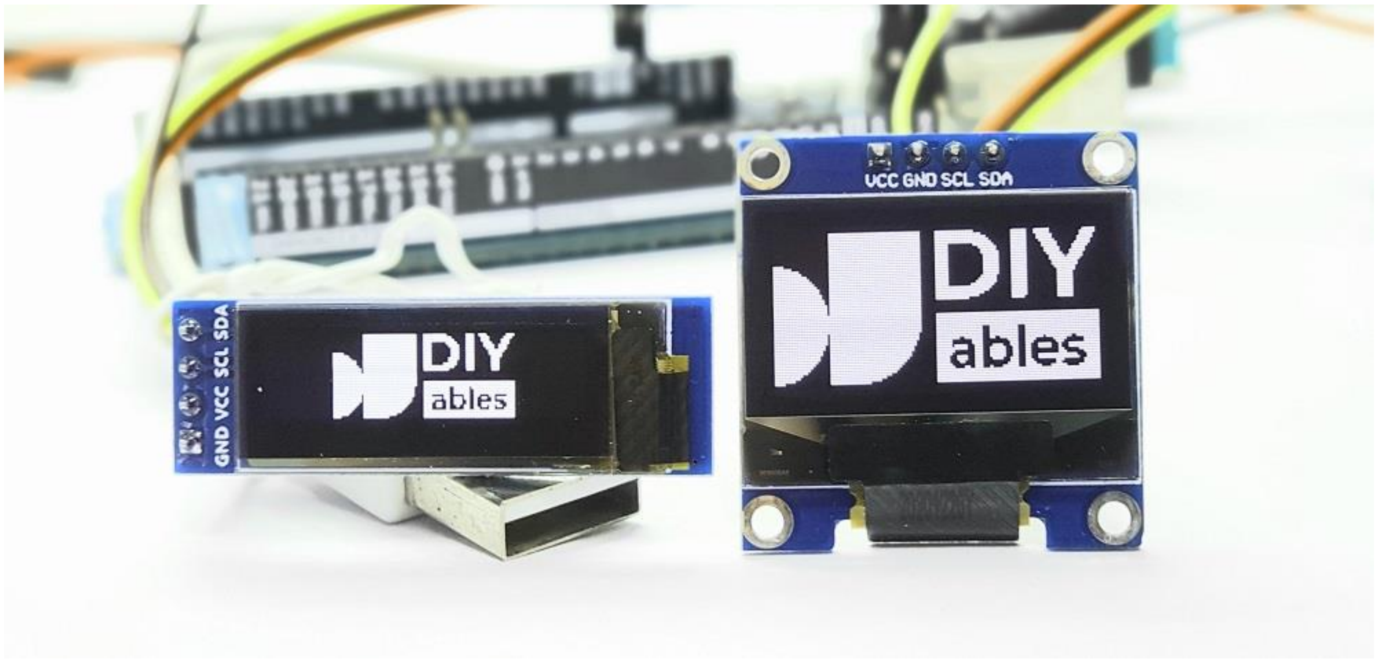
The OLED (Organic Light-Emitting Diode) display is an alternative for LCD display. The OLED is super-light, almost paper-thin, flexible, and produce a brighter and crisper picture.

About OLED Display-

There are many types of OLED display. They differ from each other in communication interface, sizes and colors-

- Communication interface: I2C, SPI
- Size: 128x64, 128×32...
- Color: white, blue, dual color...

SPI is generally faster than I2C but requires more Arduino pins. While I2C requires only two pins and can be shared with other I2C peripherals. It's a trade-off between pins and communication speed. The choice is up to you. For OLED with I2C interface, there are several types of driver such as SSD1306, SH1106 driver . This tutorial uses SSD1306 I2C OLED Display 128x64 and 128x32.



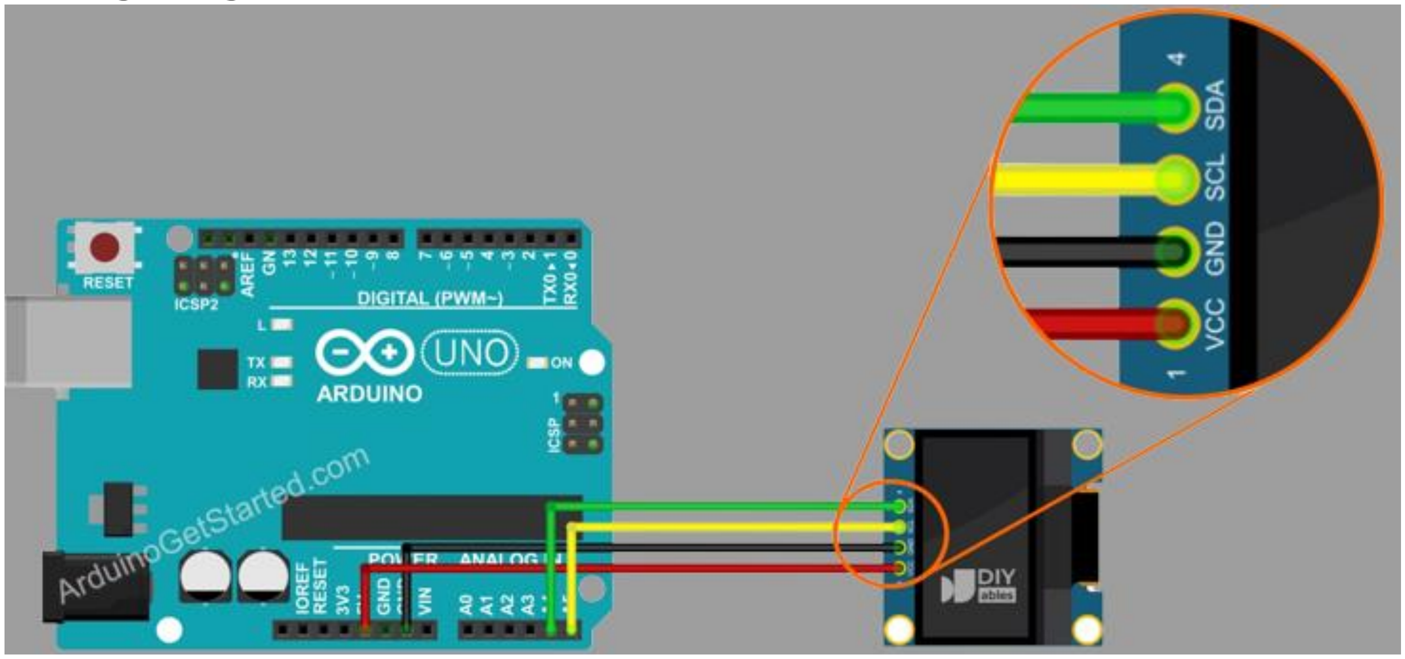
I2C OLED Display Pinout-

GND pin: should be connected to the ground of Arduino.
VCC pin: is the power supply for the display which we connect the 5 volts pin on the Arduino.
SCL pin: is a serial clock pin for I2C interface.
SDA pin: is a serial data pin for I2C interface.

- **NOTE THAT:**

The order of the OLED module's pins can vary between manufacturers and module types. ALWAYS use the labels printed on the OLED module. Look closely!.

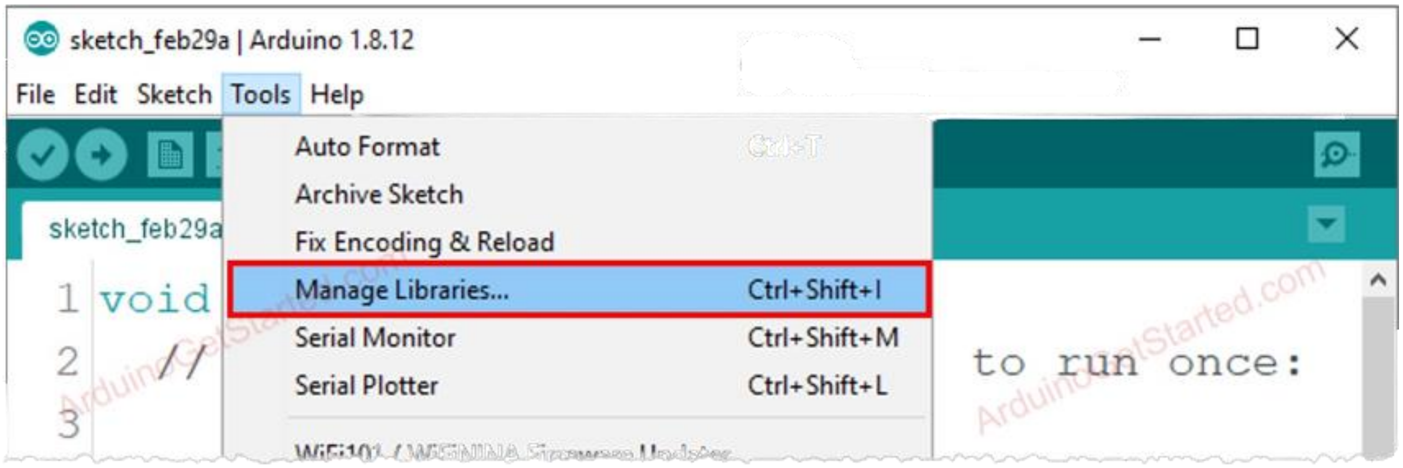
Wiring Diagram-



The real wiring diagram between Arduino and OLED 128x64

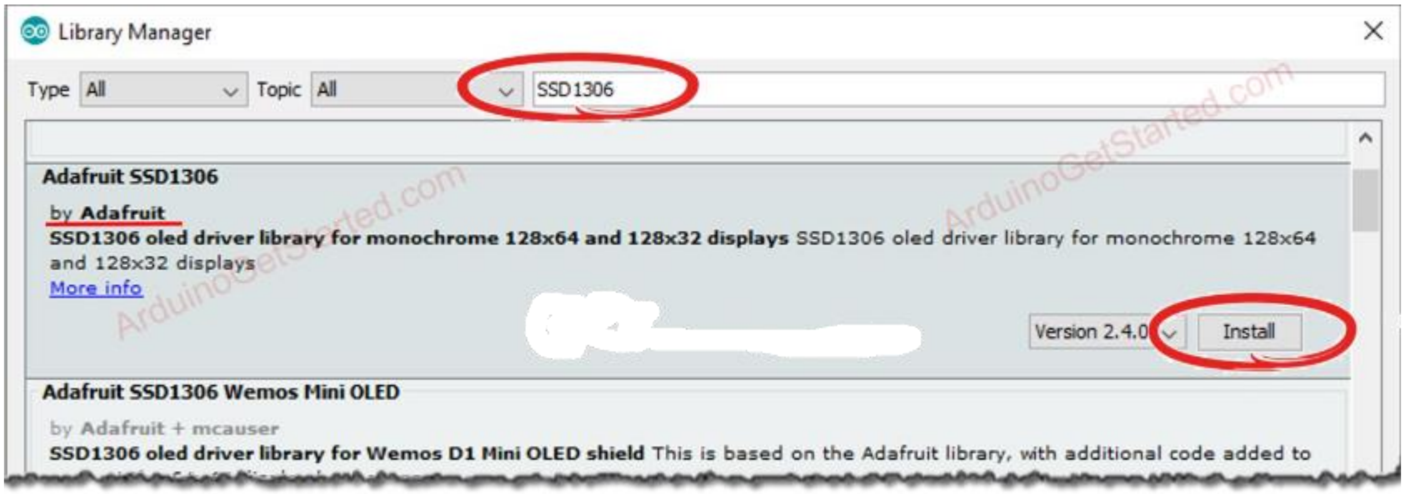
How To Use OLED with Arduino:-

Step 1 => On Arduino IDE, go to Tools >> Manage Libraries


sketch_feb29a | Arduino 1.8.12
File Edit Sketch Tools Help
Auto Format
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
sketch_feb29a

```
void
  //                          to run once:
```

Step 2 => Search "SSD1306", then find the SSD1306 library by Adafruit.
Step 3 => Click Install Button to install the Library.



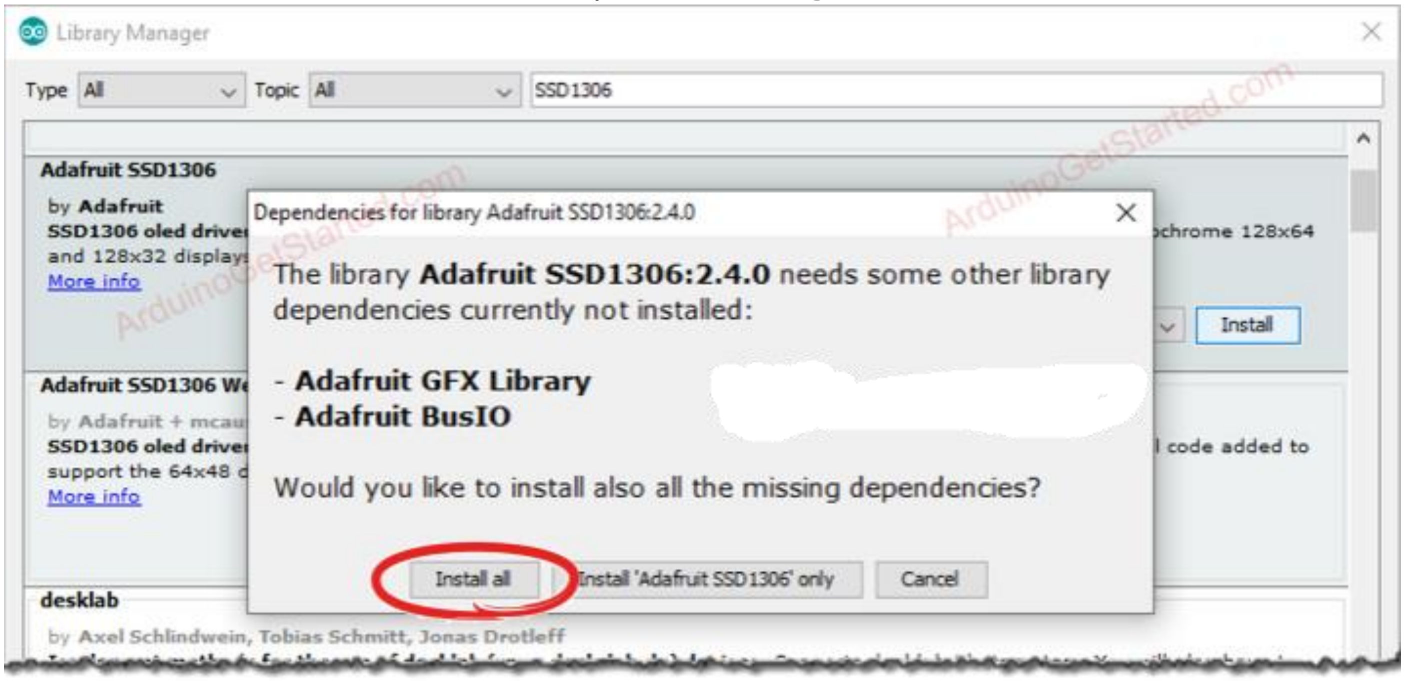
Adafruit SSD1306
by **Adafruit**
SSD1306 oled driver library for monochrome 128x64 and 128x32 displays SSD1306 oled driver library for monochrome 128x64 and 128x32 displays

Adafruit SSD1306 Wemos Mini OLED
by **Adafruit + mcauser**
SSD1306 oled driver library for Wemos D1 Mini OLED shield This is based on the Adafruit library, with additional code added to

Step 4 => You will be asked for insatalling some other library dependencies.

Step 5 => Click Install all to Install all library dependencies.

And also install some library According to needs.



Dependencies for library Adafruit SSD1306:2.4.0

The library **Adafruit SSD1306:2.4.0** needs some other library dependencies currently not installed:

- **Adafruit GFX Library**
- **Adafruit BusIO**

Would you like to install also all the missing dependencies?

<u>Program to print temperature and Humidity reading on it.</u>

<u>Introduction</u>:-

The DHT22 is a basic, low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed). It's easy to use, but requires careful timing to grab data. The only real downside of this sensor is you can only get new data from it once every 2 seconds, so when using our library, sensor readings can be up to 2 seconds old.

Hardware connection.

- Maker Uno
- DHT22 Sensor Module Breakou
- OLED I2C 0.96Inch 128×64 Blue Display
- Breadboard 8.5×5.5cm (400 Holes)

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "DHT.h"

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
#define OLED_RESET     4 // Reset pin # (or -1 if sharing Arduino reset
pin)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET);

#define DHTPIN 2
#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321
DHT dht(DHTPIN, DHTTYPE);

#define PIEZO 8
```

```
#define ALARM  1000

int Sound[] = {ALARM, ALARM};
int SoundNoteDurations[] = {4, 4};

#define playSound() playMelody(Sound, SoundNoteDurations, 2)

char inChar;
String inString;

void setup() {

  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for (;;); // Don't proceed, loop forever
  }
  display.clearDisplay();
  dht.begin();

}

void loop() {

  // Wait a few seconds between measurements.
  delay(2000);

  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow
sensor)
  float h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();
  // Read temperature as Fahrenheit (isFahrenheit = true)
  float f = dht.readTemperature(true);

  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  // Compute heat index in Fahrenheit (the default)
  float hif = dht.computeHeatIndex(f, h);
  // Compute heat index in Celsius (isFahreheit = false)
  float hic = dht.computeHeatIndex(t, h, false);

  if (hic >= 41) {
    displayOled();
   playSound();
  }
  else {
    displayOled();
    noTone(PIEZO);
  }

}

void displayOled() {

  // Read humidity
  float h = dht.readHumidity();
```

```
// Read temperature as Celsius (the default)
float t = dht.readTemperature();
// Read temperature as Fahrenheit (isFahrenheit = true)
float f = dht.readTemperature(true);
// Compute heat index in Fahrenheit (the default)
float hif = dht.computeHeatIndex(f, h);
// Compute heat index in Celsius (isFahreheit = false)
float hic = dht.computeHeatIndex(t, h, false);

display.clearDisplay();
display.setTextColor(WHITE);
display.setTextSize(1);
display.setCursor(5, 15);
display.print("Humidity:");
display.setCursor(80, 15);
display.print(h);
display.print("%");
display.setCursor(5, 30);
display.print("Temperature:");
display.setCursor(80, 30);
display.print(t);
display.print((char)247); // degree symbol
display.print("C");
display.setCursor(5, 45);
display.print("Heat Index:");
display.setCursor(80, 45);
display.print(hic);
display.print((char)247); // degree symbol
display.print("C");
display.display();
```

```
}

void playMelody(int *melody, int *noteDurations, int notesLength)
{
  pinMode(PIEZO, OUTPUT);

  for (int thisNote = 0; thisNote < notesLength; thisNote++) {
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(PIEZO, melody[thisNote], noteDuration);
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    noTone(PIEZO);
  }
}
```