

# NOISE POLLUTION MONITORING USING IOT

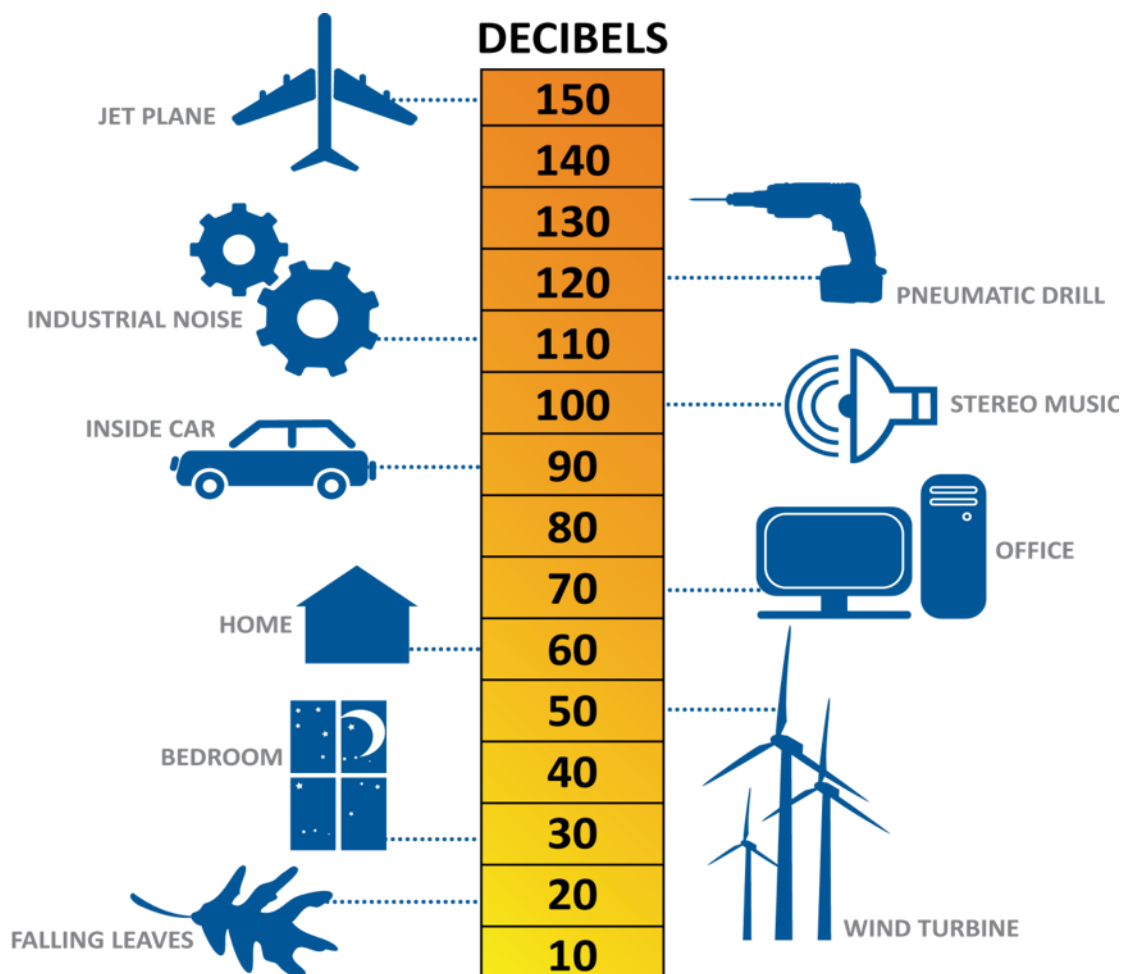
## PHASE-4

### INTRODUCTION

Noise pollution is one of the major environmental problems that affects the health and well-being of humans and animals. It can cause hearing loss, stress, hypertension, sleep disturbance, and reduced productivity. Therefore, it is important to monitor and control the noise level in different environments, such as urban areas, industrial zones, schools, hospitals, and airports.

One of the possible solutions to monitor noise pollution is to use the Internet of Things (IoT) technology. IoT is a network of interconnected devices that can collect, process, and transmit data over the internet. By using IoT devices, such as microphones, microcontrollers, and Wi-Fi modules, we can measure the noise level in real time and send the data to a cloud server for analysis and visualization. We can also generate alerts and notifications when the noise level exceeds a certain threshold or violates the regulations.

In this paper, we present a noise pollution monitoring system using IoT that can detect and report the noise level in different locations.



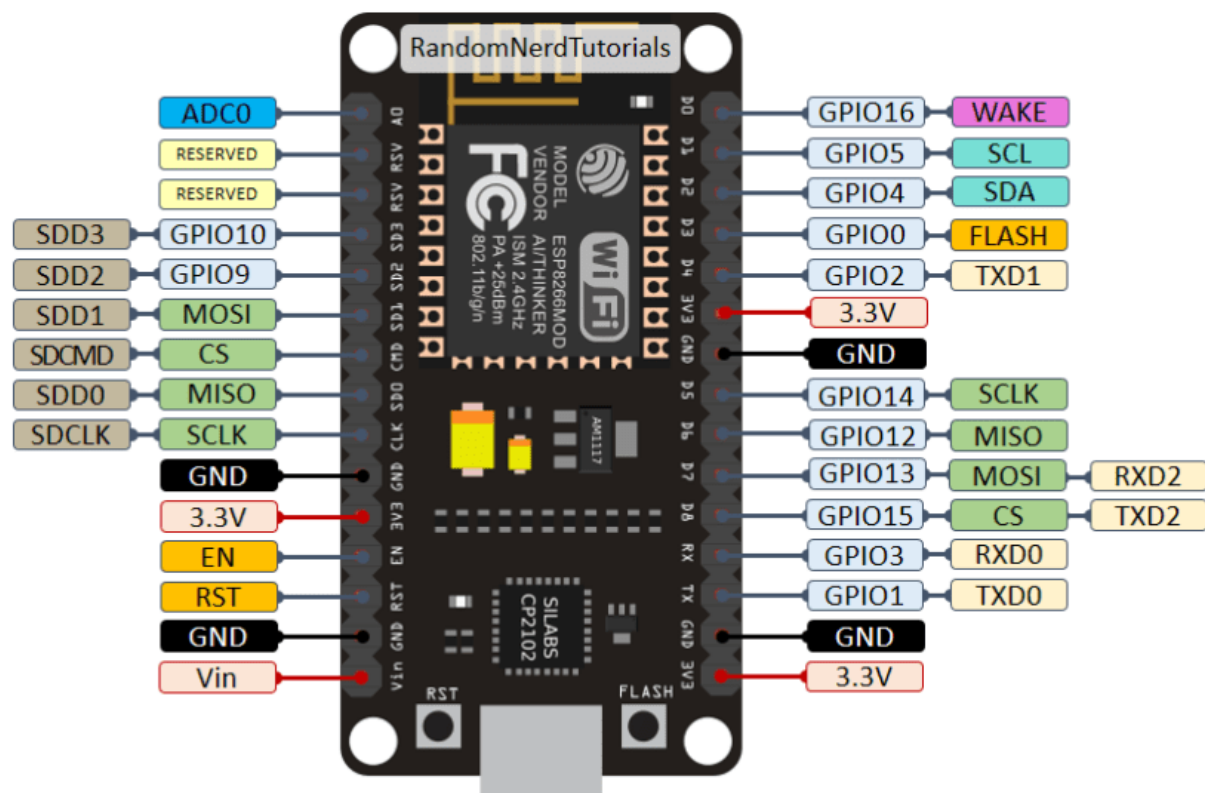
## COMPONENTS REQUIRED

- ESP8266 NodeMCU Board
- Microphone sensor
- 16\*2 LCD Module
- Breadboard
- Connecting wires

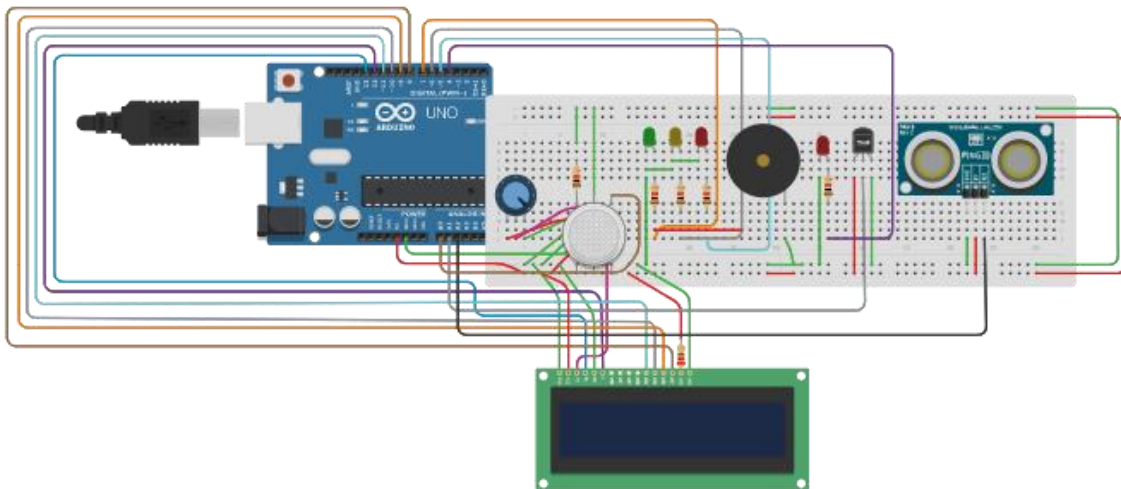
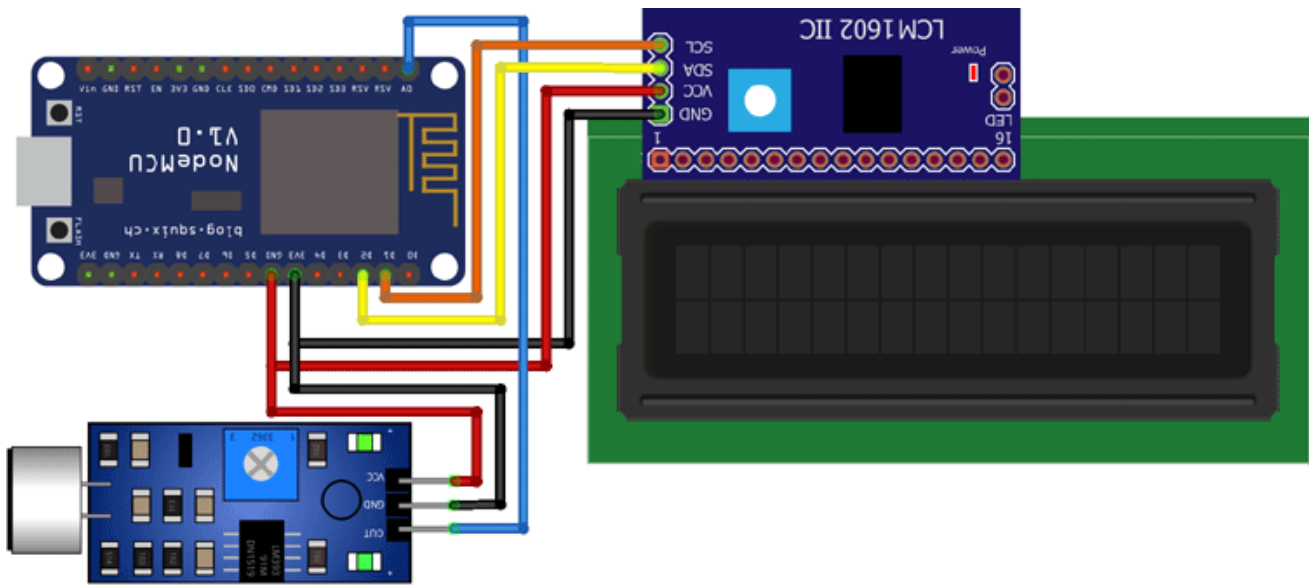
# ESP8266

The **ESP8266** is a low-cost Wi-Fi microchip with built-in TCP/IP networking software and microcontroller capability. It is designed and manufactured by Espressif Systems in Shanghai, China. The chip contains the crucial elements of a computer, including CPU, RAM, networking (WiFi), and even a modern operating system and SDK.

## PIN DIAGRAM



## CIRCUIT DIAGRAM



## PROGRAM

```
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
```

```

#include <LiquidCrystal_I2C.h>
#define SENSOR_PIN A0
LiquidCrystal_I2C lcd(0x3F, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
const int sampleWindow = 50;
unsigned int sample;
int db;
char auth[] = "IEu1xT825VDt6hNfrCFgdJ6InJ1QUfsA";
char ssid[] = "realme 6";
char pass[] = "evil@zeb";
BLYNK_READ(V0)
{
  Blynk.virtualWrite(V0, db);
}
void setup() {
  pinMode (SENSOR_PIN, INPUT);
  lcd.begin(16, 2);
  lcd.backlight();
  lcd.clear();
  Blynk.begin(auth, ssid, pass);
}
void loop() {
  Blynk.run();
  unsigned long startMillis = millis(); // Start of sample window
  float peakToPeak = 0; // peak-to-peak level
  unsigned int signalMax = 0; //minimum value
  unsigned int signalMin = 1024; //maximum value
  // collect data for 50 mS
  while (millis() - startMillis < sampleWindow)
  {
    sample = analogRead(SENSOR_PIN); //get reading from microphone
    if (sample < 1024) // toss out spurious readings
    {
      if (sample > signalMax)
      {
        signalMax = sample; // save just the max levels
      }
      else if (sample < signalMin)
      {
        signalMin = sample; // save just the min levels
      }
    }
  }
  peakToPeak = signalMax - signalMin; // max - min = peak-peak amplitude
  Serial.println(peakToPeak);
  db = map(peakToPeak, 20, 900, 49.5, 90); //calibrate for deciBels
  lcd.setCursor(0, 0);
  lcd.print("Loudness: ");

```

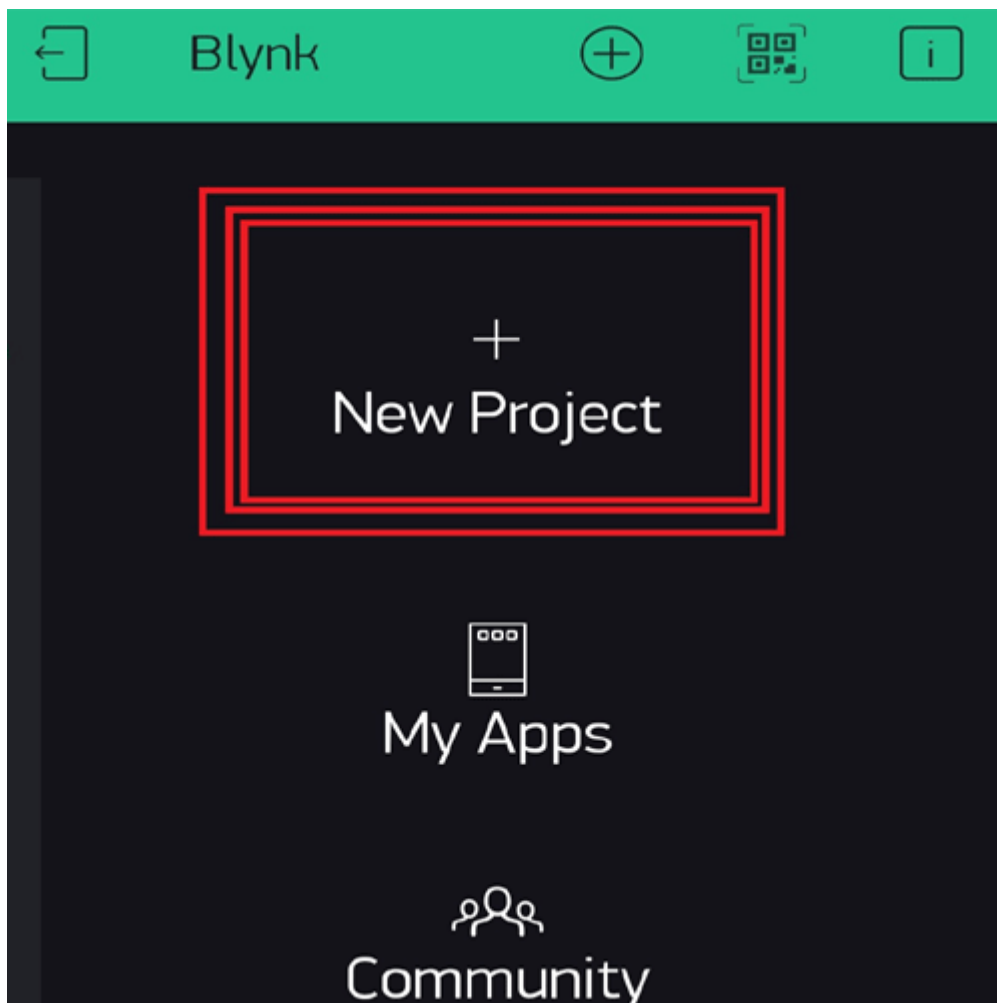
```
lcd.print(db);  
lcd.print("dB");  
if (db <= 50)  
{  
    lcd.setCursor(0, 1);  
    lcd.print("Level: Quite");  
}  
else if (db > 50 && db < 75)  
{  
    lcd.setCursor(0, 1);  
    lcd.print("Level: Moderate");  
}  
else if (db >= 75)  
{  
    lcd.setCursor(0, 1);  
    lcd.print("Level: High");  
}  
delay(600);  
lcd.clear();  
}
```

## CONNECTING TO MOBILE APPLICATION (BLYNK APK)

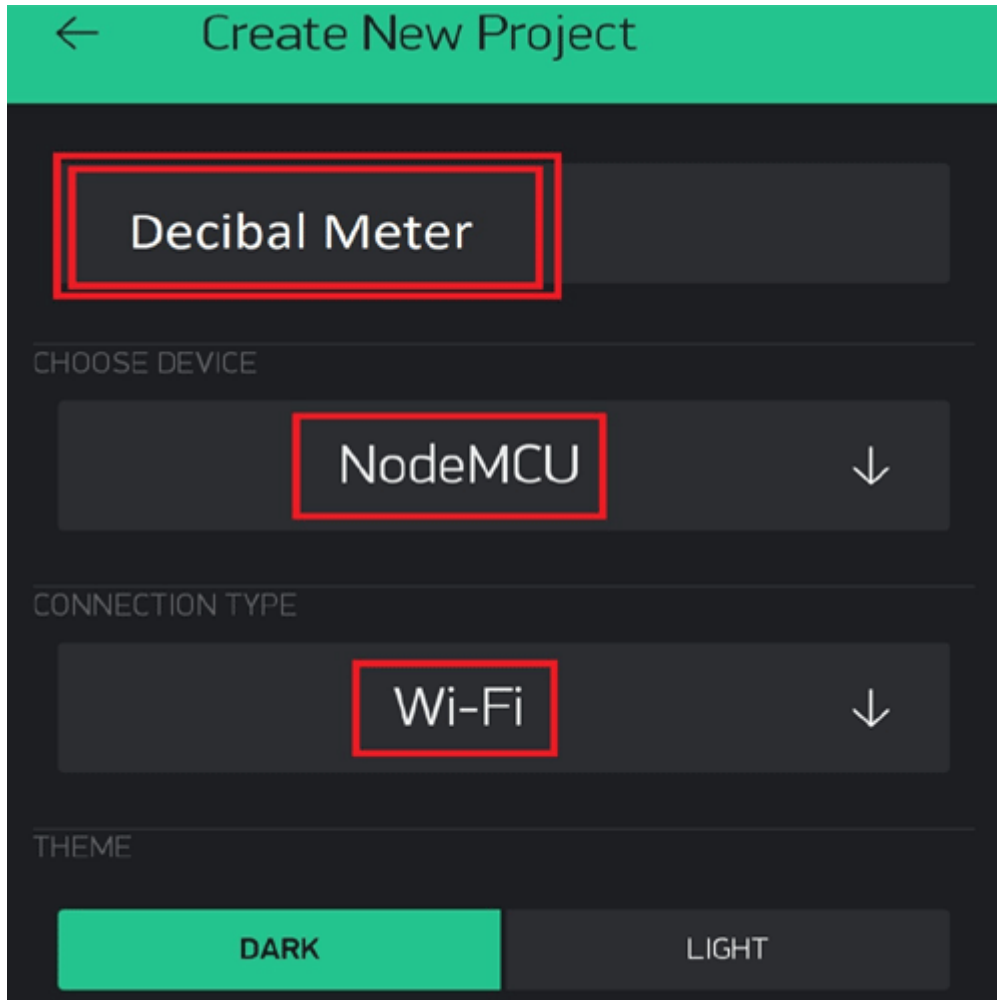
Blynk is an **IoT software platform** that provides infrastructure for the internet of Things. [It supports 400+ development boards, SBC's, and other modules.](#) It allows users to build and manage connected products with no code, and to remotely control, monitor, and automate them with mobile and web applications. [It also offers secure cloud, data analytics, user and access management, alerts, and Over-The-Air firmware updates<sup>1</sup>.](#)

To create a device using Blynk, you can manually create a device using Blynk.Console for initial prototyping (works for any hardware) or use Static Tokens for cellular, Ethernet, and other non-WiFi connection methods. Activating devices with manually generated AuthTokens is recommended for prototyping stages or when you build a device for yourself.

- First we need to install the Blynk app from PlayStore and create an account.
- Click on the create button and create your new project.



- Give your project a name and choose the board as NodeMCU and connection type as Wi-Fi.



← Create New Project

Decibal Meter

CHOOSE DEVICE

NodeMCU ↓

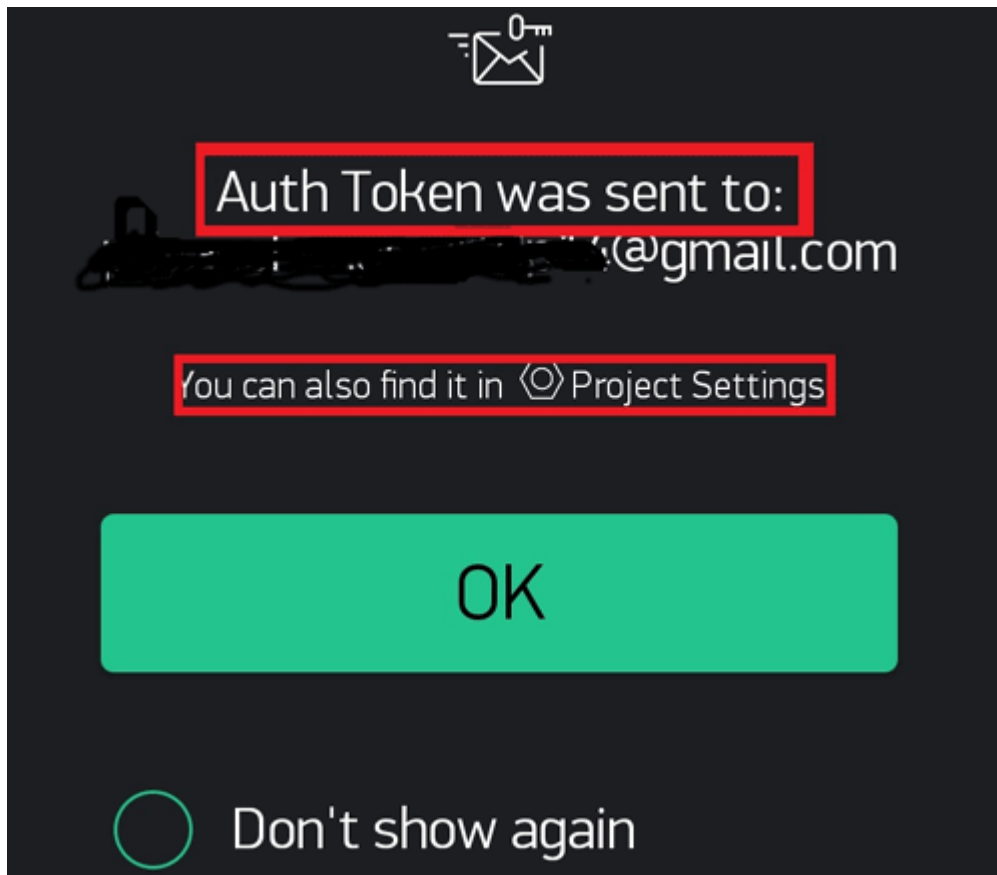
CONNECTION TYPE

Wi-Fi ↓

THEME

DARK LIGHT

- An auth token will be sent to your registered email id. Keep it safe as it will be used later on while programming.



*Auth Token is a unique alphanumeric string to identify your project in the Blynk's server and assigning the correct data through it.*

- Now drag and drop a gauge from the list and configure it.






- Connect the gauge to virtual pin V0 and set the values to 0 and 90 respectively, also set the reading rate to 1 sec.

Gauge Settings


i



Loudness

INPUT

V0

0  90

LABEL


e.g: Temp: /pin/ °C

DESIGN

FONT SIZE

T T T

TEXT

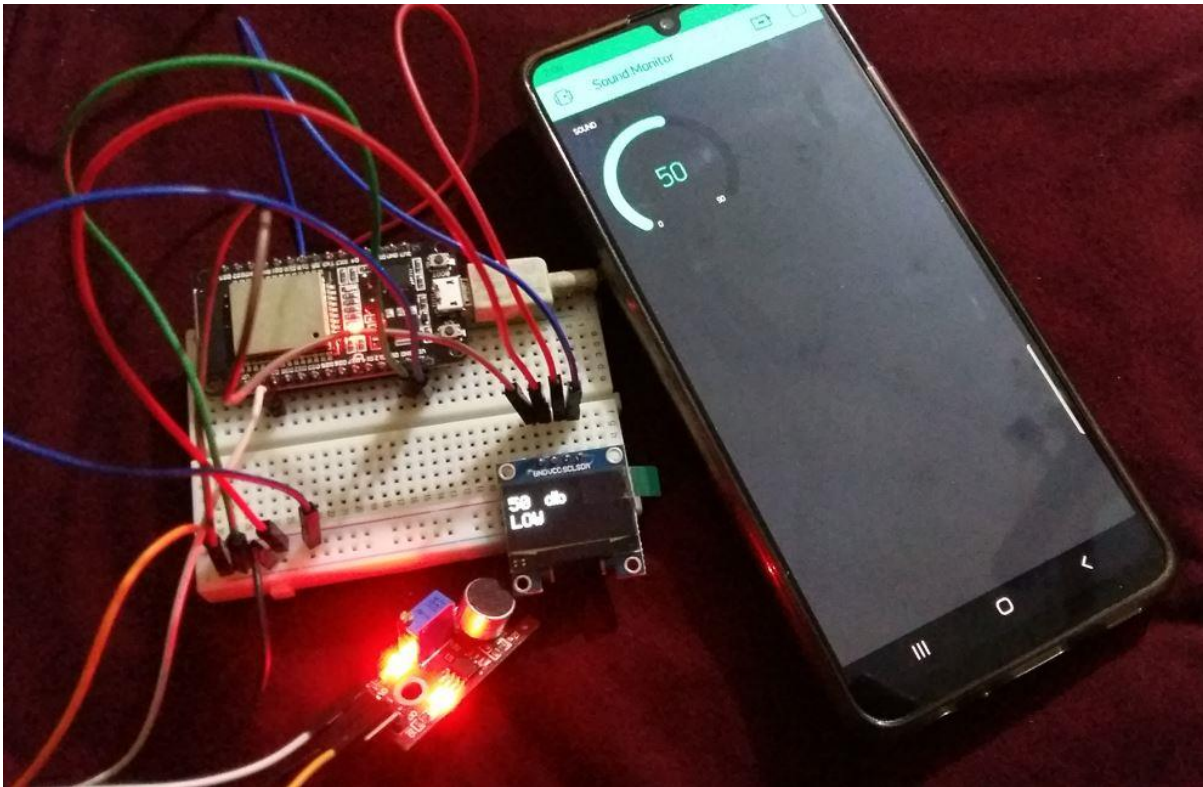


READING RATE

1 sec ↓

And now you're done with setting up Blynk. Let us jump to the coding part.

## MOBILE APP OUTPUT



## HTML PROGRAM

```
<!DOCTYPE html>

<html>

<head>

  <title>IoT Data Display</title>

</head>

<body>

  <h1>IoT Data Display</h1>

  <div id="iotData">

    <!-- IoT data will be displayed here -->

  </div>
```

```
<script>

// JavaScript code to retrieve and display IoT data

function fetchData() {

    const apiUrl = 'https://blynk.io';

    fetch(apiUrl)

        .then(response => response.json())

        .then(data => {

            // Assuming your IoT data is in a format like { "sensor1": value1, "sensor2":
value2 }

            const iotData = data;

            // Update the HTML element with the IoT data

            const iotDataDiv = document.getElementById('iotData');

            iotDataDiv.innerHTML = '<h2>IoT Data:</h2>' + JSON.stringify(iotData, null,
2);

        })

        .catch(error => {

            console.error('Failed to fetch IoT data:', error);

        });

}

// Call the fetchData function to start displaying IoT data

fetchData();

</script>

</body>

</html>
```

## SAMPLE OUTPUT

