NOISE POLLUTION MONITORING USING IOT

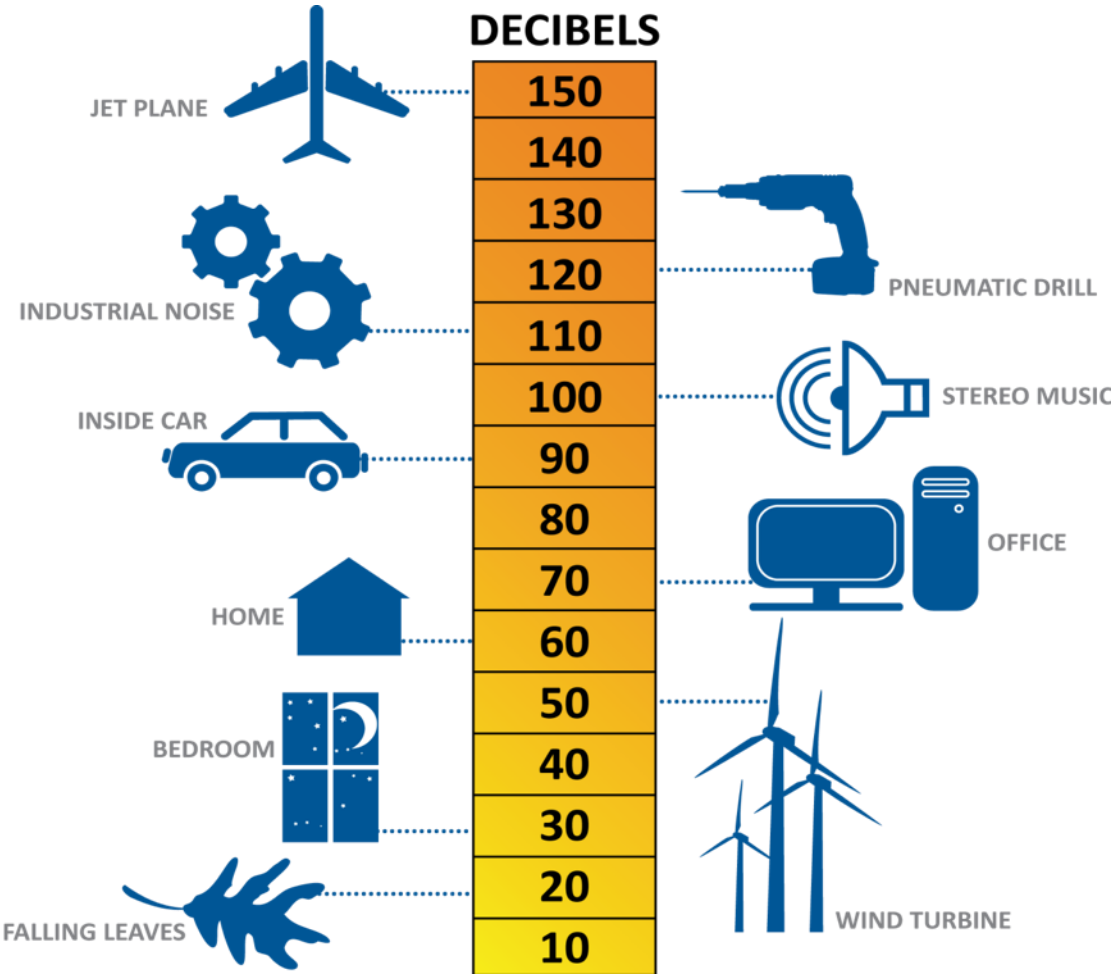
PHASE-3

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.



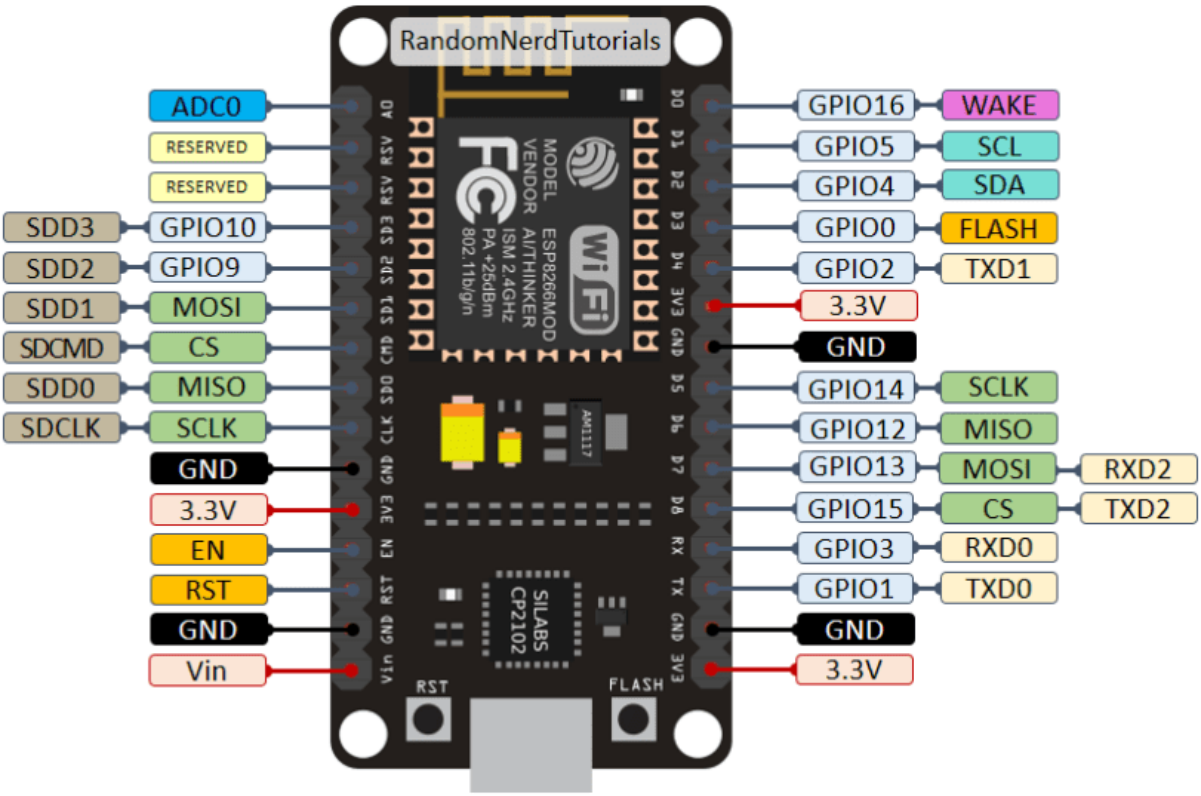
COMPONENTS REQUIRED

- 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



BLOCK DIAGRAM

A block diagram of a noise pollution monitoring and controlling system using IoT is shown below:

The main components of the system are:

- **Noise sensor**: A device that detects the sound waves in the environment and converts them into electrical signals. The noise sensor can be a microphone or a sound detector module.
- **Microcontroller**: A device that processes the electrical signals from the noise sensor and converts them into digital data. The microcontroller can be an Arduino or a NodeMCU board.
- **WiFi module**: A device that connects the microcontroller to the internet and enables wireless communication with other devices. The WiFi module can be integrated with the microcontroller (such as NodeMCU) or separate (such as ESP8266).
- **Cloud platform**: A service that stores and analyzes the digital data from the microcontroller and provides a user interface for accessing and visualizing the data. The cloud platform can be a web server or an IoT platform (such as Blynk or ThingSpeak).
- **Monitor**: A device that displays the noise levels in different formats, such as numbers, graphs, or maps. The monitor can be a computer screen or a mobile application.
- **Noise source**: A device that generates noise in the environment and can be controlled by the microcontroller. The noise source can be a machine, a speaker, a vehicle, etc.
- **Actuator**: A device that receives commands or signals from the microcontroller and performs actions on the noise source. The actuator can be a relay, a motor, a buzzer, etc.

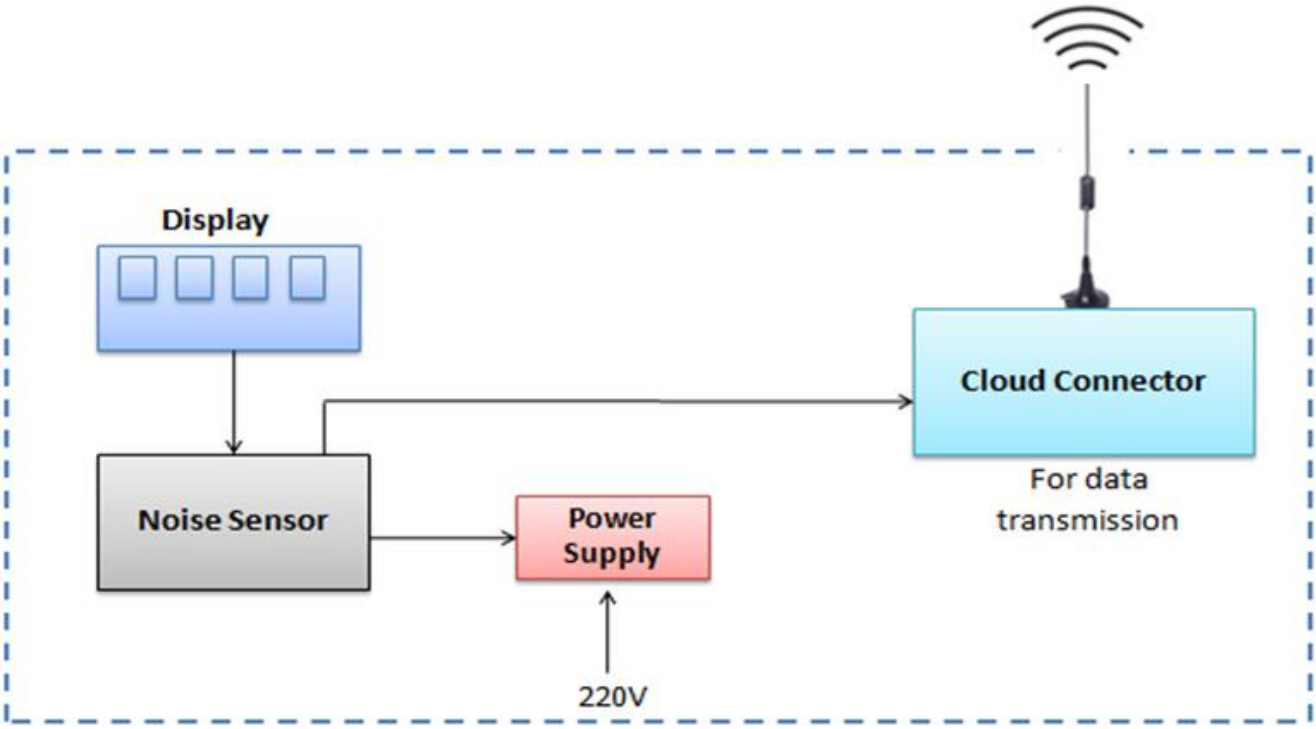
The working principle of the system is:

- The noise sensor captures the sound waves in the environment and converts them into electrical signals.
- The microcontroller reads the electrical signals from the noise sensor and converts them into digital data. The microcontroller also calculates the noise level in decibels (dB) using a formula or a library.
- The WiFi module sends the digital data and the noise level to the cloud platform via the internet.
- The cloud platform stores and analyzes the digital data and the noise level and provides a user interface for accessing and visualizing them.
- The monitor displays the noise level in different formats on a screen or an application.
- The user can view the noise level in real-time and compare it with a threshold or a standard value. The user can also set alerts or notifications for high or low noise levels.
- The user can also control the noise source by sending commands or signals to the microcontroller via the cloud platform.
- The microcontroller receives the commands or signals from the user and sends them to the actuator via wires or wireless communication.

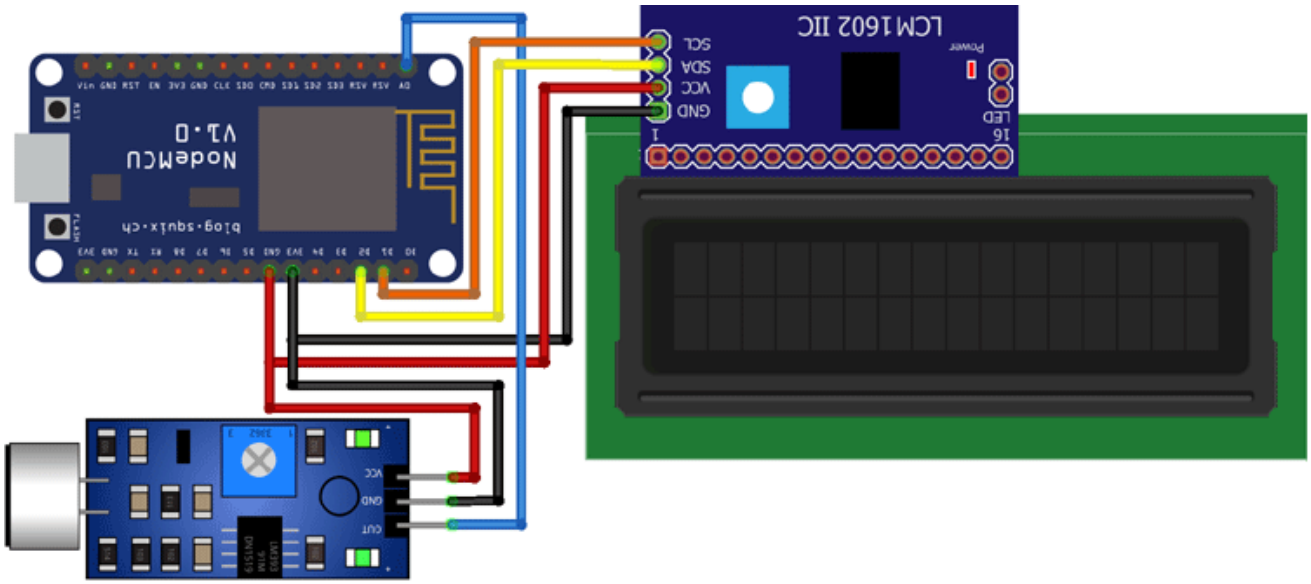
- The actuator performs actions on the noise source according to the commands or signals from the user. For example, it can turn on or off the noise source, adjust its volume or speed, etc.

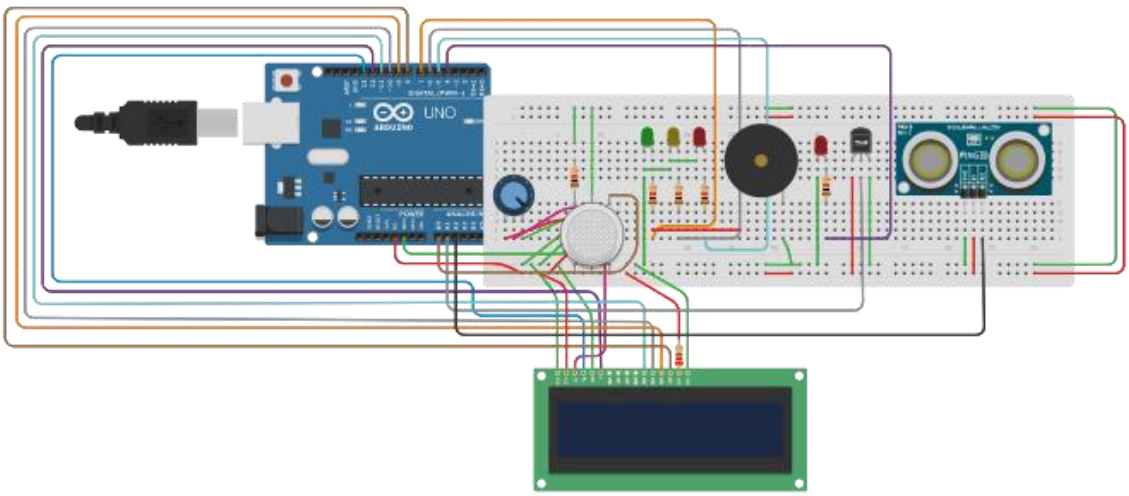
The advantages of using IoT for noise pollution monitoring and controlling are:

- It provides real-time and accurate measurement of noise levels in different places.
- It allows remote access and control of noise sources from anywhere using an internet connection.
- It enables data storage and analysis for further research and improvement of noise pollution management.
- It reduces human intervention and error in noise pollution monitoring and controlling.



CIRCUIT DIAGRAM



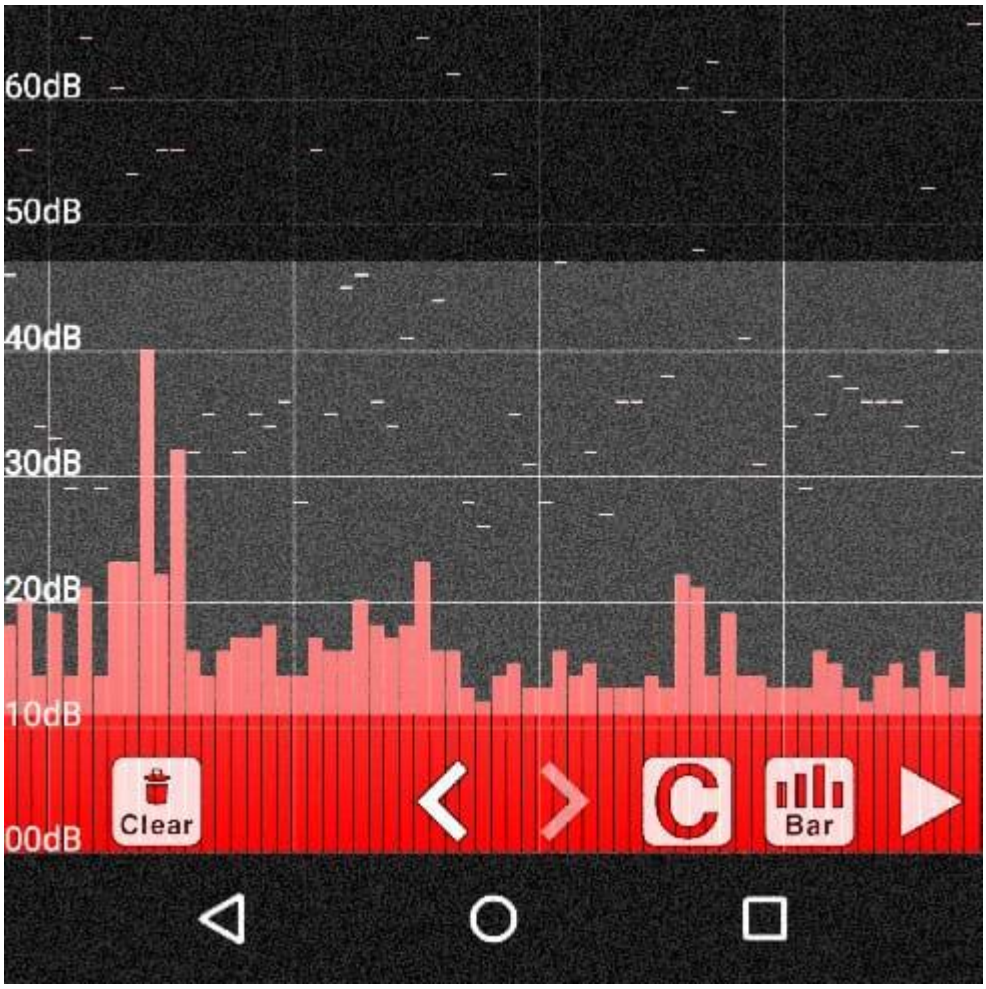


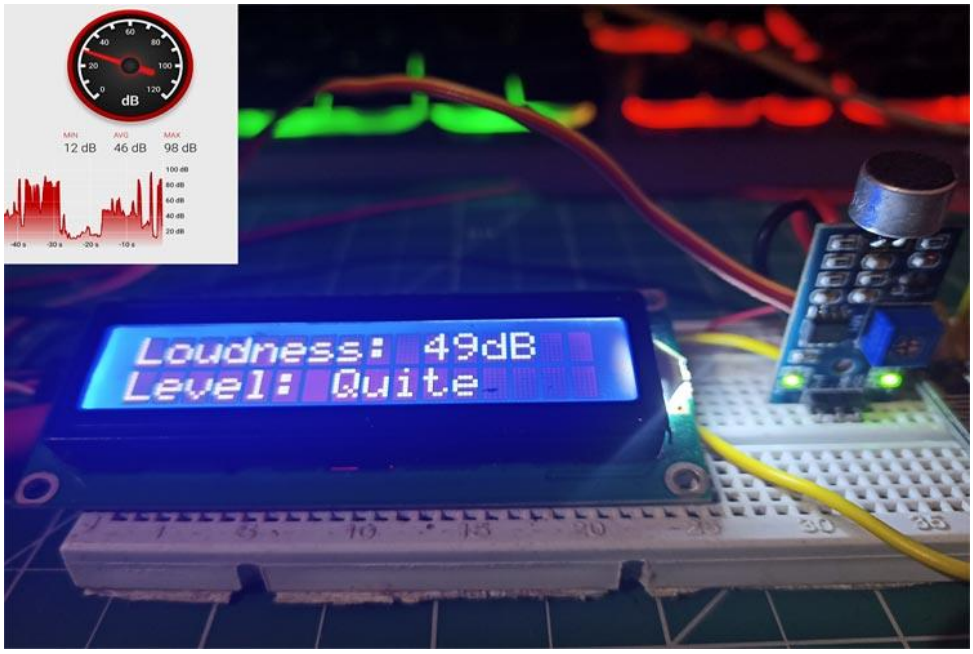
PROGRAM

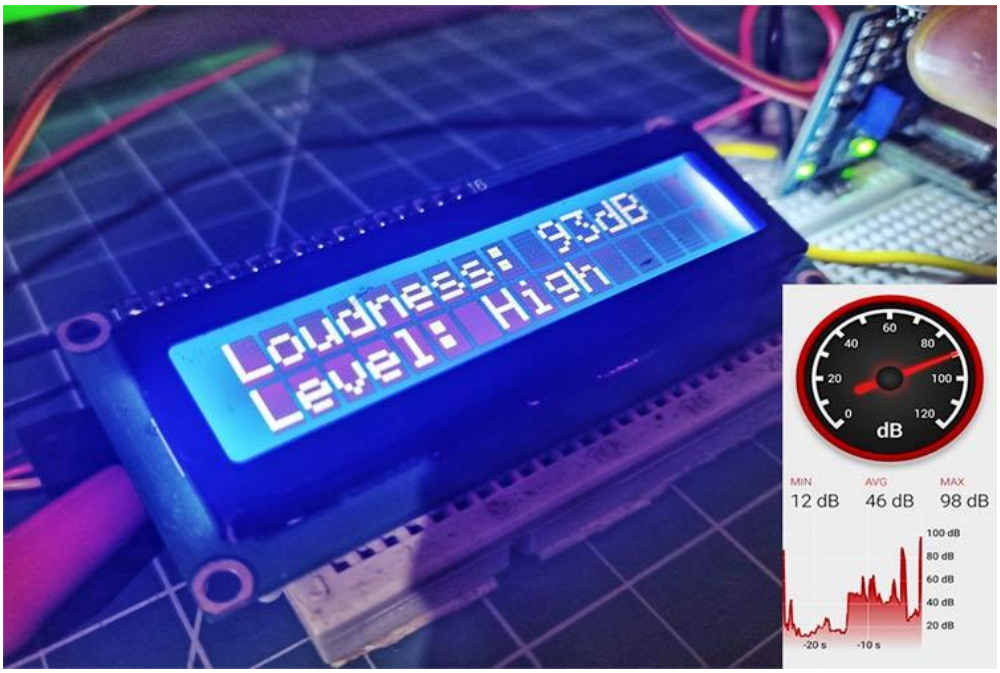
```
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <LiquidCrystal_I2C.h>
#define SENSOR_PIN A0
LiquidCrystal_I2C lcd(0x3F, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
const int sampleWindow = 50;
unsigned int sample;
int db;
char auth[] = "IEu1xT825VDt6hNfrcFgdJ6InJ1QUfsA";
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();
}
```

SAMPLE OUTPUT






Loudness: 93dB
Level: High
40
60
80
20
100
0
120
dB
MIN
AVG
MAX
12 dB
46 dB
98 dB
100 dB
80 dB
60 dB
40 dB
20 dB
-20 s
-10 s