Noise Pollution Monitoring using IoT in Urban and Rural Areas

***Abstract*:**
This large amount of increasing pollution has made human life prone to large number of diseases. Therefore, it has now become necessary to control the pollution to ensure healthy livelihood and better future. Here i propose a sound pollution monitoring system that allows us to monitor and check live sound pollution in a particular areas through IOT. Also system keeps measuring sound level and reports it to the online server over IOT. The sensors interact with microcontroller which processes this data and transmits it over internet. This allows authorities to monitor noise pollution in different areas and take action against it. Also authorities can keep a watch on the noise pollution near schools, hospitals and no honking areas, and if system detects noise issues it alerts authorities so they can take measures to control the issue.
***Keywords*: IoT, Microcontroller, Monitoring System, Noise Pollution.**

1. Introduction

The sensors interact with raspberry pi which processes this data and transmits it over the application. This allows authorities to monitor air pollution in different areas and act against it. Also, authorities can keep a watch on the noise pollution near schools, hospitals and no honking areas, and if system detects air quality and noise issues it alerts authorities so they can take measures to control the issue. Some future consumer applications envisioned for IoT sound like science fiction, but some of the more practical and realistic sounding possibilities for the technology include: Receiving warnings on your phone or wearable device when IoT networks detect some physical danger is detected nearby. Self-parking automobiles. Automatic ordering of groceriesand other home. Automatic tracking of exercise habits and other day-to-day personal activity including goal tracking and regular progress reports. Network Devices and the Internet of Things All kinds of ordinary household gadgetscan be modified to workin an IoT system. Wi-Fi network adapters, motion sensors, cameras, microphones and other instrumentation can be embedded in these devices to enable them for work in the Internet of Things. Home automation systems already implement primitive versions of this concept for things like light bulbs, plus other devices like wireless scales and wireless blood pressure monitors that each represent early examples of IoT gadgets. The main objective of IOT Air & Sound Monitoring System is that the Air and sound pollution is a growing issue these days.

It is necessary to monitor air quality and keep it under control for a better future and healthy living for all. Due to flexibility and low cost Internet of things (IoT) is getting popular day by day. With the urbanization and with the increase in the vehicles on road the atmospheric conditions have considerably affected. 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. Monitoring gives measurements of air pollutant and sound pollution concentrations, which can then be analyzed

interpreted and presented. This information can then be applicable in many ways. Analysis of monitoring data allows us to assess how bad air pollution and sound pollution is from day to day.

A. Internet of things

The IoT abbreviated as Internet of Things is the physical objects network. Mainly includes devices, buildings, vehicles and other items which are embedded with the electronics, sensors, software's and network connectivity which enables the objects to collect and exchange the data. The objects in the Internet of Things allows to sense and control remotely across existing network infrastructure. Opportunity creation for physical world directs the integration into computer based systems. An improved efficiency will be resulted. The accuracy and economic benefit also achieved. Augmentation of Internet of Things with the sensors and the actuators, technology becomes the more general instance of classes of cyber and physical systems, that encompasses the technologies like smart cities, smart homes, smart grids and the intelligent transportation systems. Each of the things is uniquely identifiable with the embedded computing systems but also it is able to interoperate within the existing infrastructure of Internet. The experts used to estimate that IoT consists of almost 50 billion objects by the year 2020.

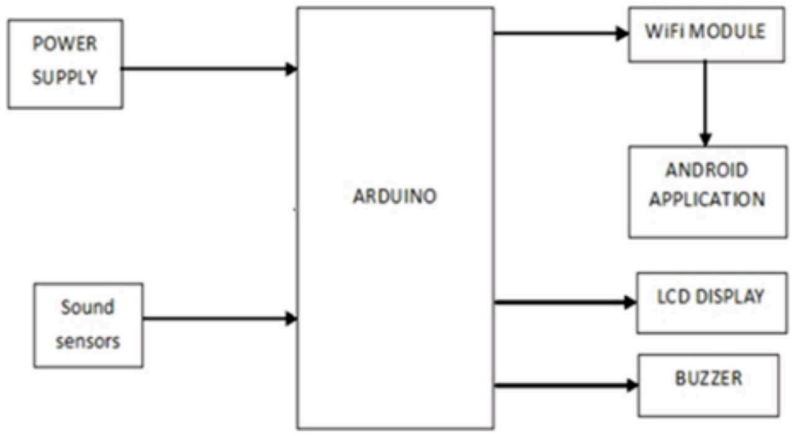
B. Sensors application in data collection

The physical objects that are being connected will possess one or more sensors. Each sensor will monitor a specific condition such as location, vibration, motion and temperature. In IoT, these sensors will connect to each other and to systems that can understand or present information from the sensor's data feeds. These sensors will provide new information to a company's systems and to people.

C. IP network communication

In the past, people communicated with people and with machines. Imagine if all of your equipment had the ability to communicate. What would it tell you? IoT-enabled objects will share information about their condition and the surrounding environment with people, software systems and other machines. This information can be shared in realtime or collected and shared at defined intervals. Going forward, everything will have a digital identity and connectivity, which means you, can identify, track and communicate with objects.

2. Block diagram of the proposed system

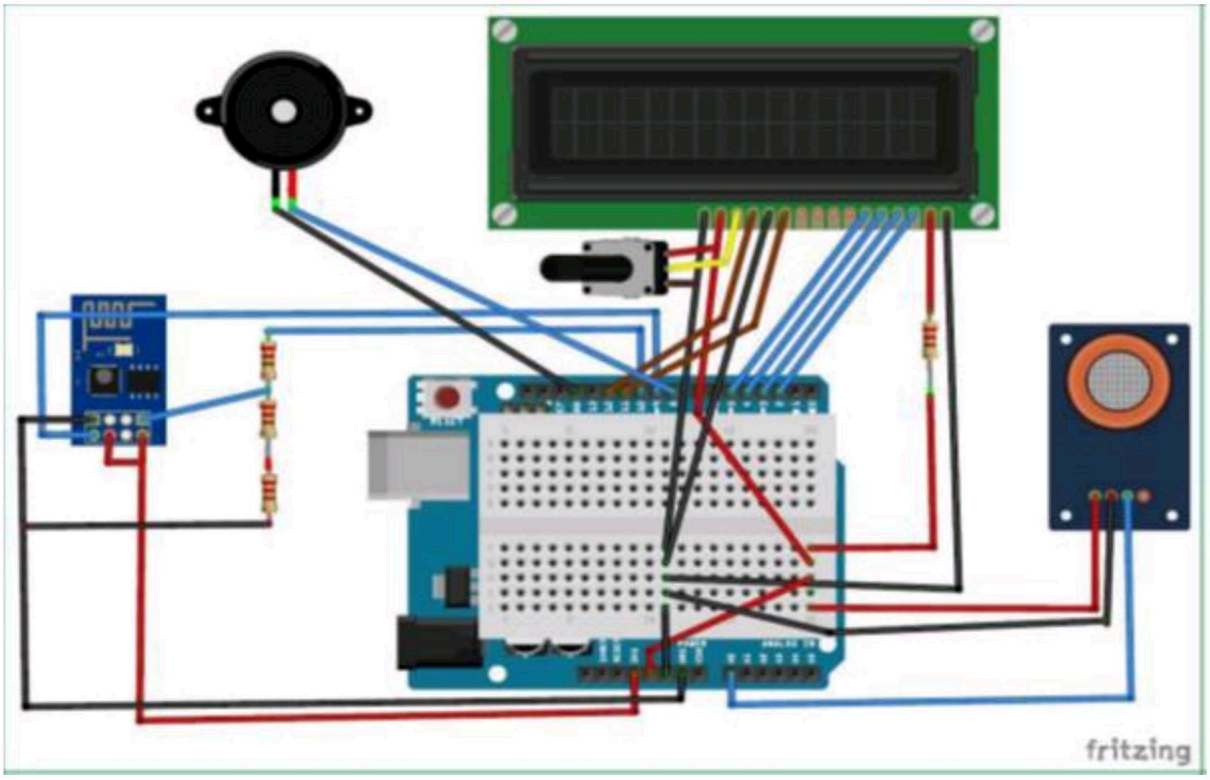


This system is made to fulfill the purpose and need of the society to monitor and check the live air quality and sound pollution in an area through IOT. The system uses air sensors to check the presence of harmful and hazardous gases/ compounds [such as Methane, propane, Butane, alcohol, noxious gases, carbon monoxide etc. in the air and also uses the sound sensor to keep measuring sound level in the surroundings. MQ2 is the air sensors which are used to collect air pollutants and a sound sensor module mic is used to capture sound. These sensors interact with arduino which processes this data and then transmit it over the mobile application. To send the data over remote location WIFI modem is also installed. and whenever the air pollution is detected, a buzzer immediately beeps and when there is a noise pollution an LED starts blinking continuously. With this system, not only the authorities but also the localized people can check the transmitted data through their mobile phone and that too without spending single penny and the people can act against it on their level and try to bring the pollution level under control. This system would contribute as a part in the building of a healthy society.

3. Circuit diagram of noise pollution monitoring system

First of all we will connect the ESP8266 with the Arduino. ESP8266 runs on 3.3V and if you will give it 5V from the Arduino then it won't work properly and it may get damage. Connect the VCC and the CH_PD to the 3.3V pin of Arduino. The RX pin of ESP8266 works on 3.3V and it will not communicate with the Arduino when we will connect it directly to the Arduino. So, we will have to make a voltage divider for it which will convert the 5V into 3.3V. This can be done by connecting three resistors in series like we did in the circuit. Connect the TX pin of the ESP8266 to the pin 10 of the Arduino and the RX pin of the esp8266 to the pin 9 of Arduino through the resistors. ESP8266 Wi-Fi module gives your projects access to Wi-Fi or internet. It is a very cheap device and make your projects very powerful. It can communicate with any microcontroller and it is the most leading devices in the IOT platform. Learn more about using ESP8266 with Arduino here. Then we will connect the MQ135 sensor with the Arduino. Connect the VCC and the ground pin of the sensor to the 5V and ground of the Arduino and the Analog pin of sensor to the A0 of the Arduino. Connect a buzzer to the pin 8 of the Arduino which will start to beep when the condition becomes true. In last, we will connect LCD with the Arduino. The connections of the LCD are as follows

- Connect pin 1 (VEE) to the ground.
- Connect pin 2 (VDD or VCC) to the 5V.
- Connect pin 3 (V0) to the middle pin of the 10K potentiometer and connect the other two ends of the potentiometer to the VCC and the GND. The potentiometer is used to control the screen contrast of the LCD. Potentiometer of values other than 10K will work too.
- Connect pin 4 (RS) to the pin 12 of the Arduino.
- Connect pin 5 (Read/Write) to the ground of Arduino. This pin is not often used so we will connect it to the ground.
- Connect pin 6 (E) to the pin 11 of the Arduino. The RS and E pin are the control pins which are used to send data and characters.
- The following four pins are data pins which are used to communicate with the Arduino.
- Connect pin 11 (D4) to pin 5 of Arduino. Connect pin 12 (D5) to pin 4 of Arduino. Connect pin 13 (D6) to pin 3 of Arduino. Connect pin 14 (D7) to pin 2 of Arduino.
- Connect pin 15 to the VCC through the 220 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.



4. Flow Chart

The MQ135 sensor can sense NH3, NOx, alcohol, Benzene, smoke and some other gases, so it is perfect gas sensor for our noise Quality Monitoring. When we will connect it to Arduino then it will sense the gases, and we will get the Pollution level in PPM (parts per million). MQ135 gas sensor gives the output in form of voltage levels and we need to convert it into PPM. So for converting the output in PPM, here we have used a library for MQ135 sensor, it is explained in detail in "Code Explanation" section below. Sensor was giving us value of 90 when there was no gas near it and the safe level of air quality is 350 PPM and it should not exceed 1000 PPM. When it exceeds the limit of 1000 PPM, then it starts cause Headaches, sleepiness and stagnant, stale, stuffy air and if exceeds beyond 2000 PPM then it can cause increased heart rate and many other diseases. When the value will be less than 1000 PPM, then the LCD and webpage will display "Fresh Air". Whenever the value will increase 1000 PPM, then the buzzer will start beeping and the LCD and webpage will display "Poor Air, Open Windows". If it will increase 2000 then the buzzer will keep beeping and the LCD and webpage will display "Danger! Move to fresh Air".

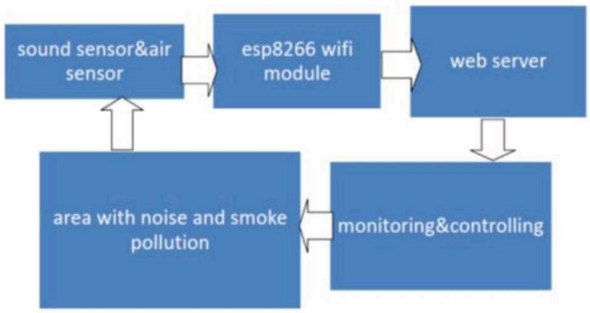
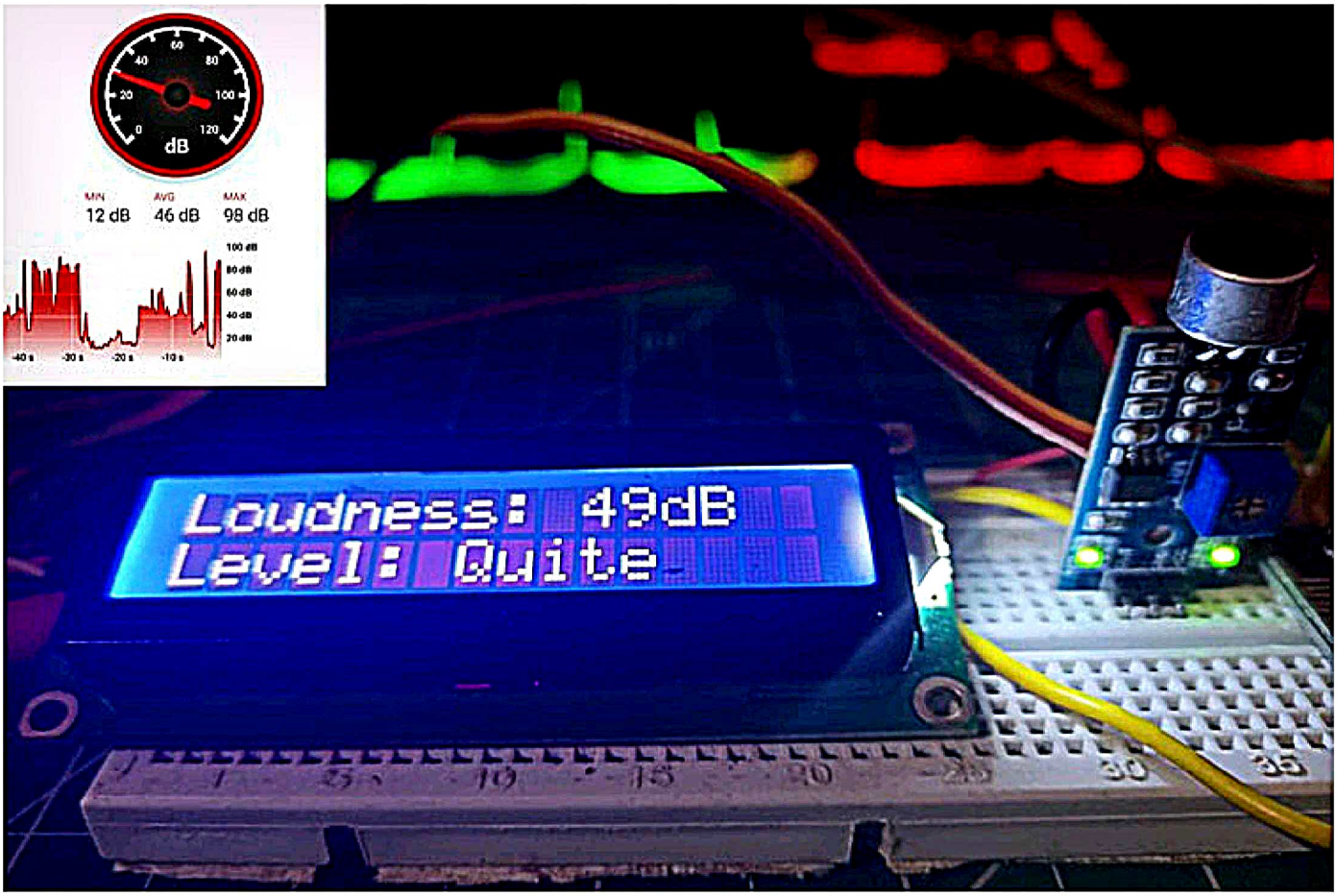


Fig. 3. Flow chart of proposed system

5. Conclusion

In this paper it is concluded that, the system is designed using structured modeling and is able to provide the desired results. It can be successfully implemented as a Real Time system with certain modifications. Science is discovering or creating major breakthrough in various fields, and hence technology keeps changing from time to time. Going further, most of the units can be fabricated on a single along with microcontroller thus making the system compact thereby making the existing system more effective. To make the system applicable for real time purposes components with greater range needs to be implemented.

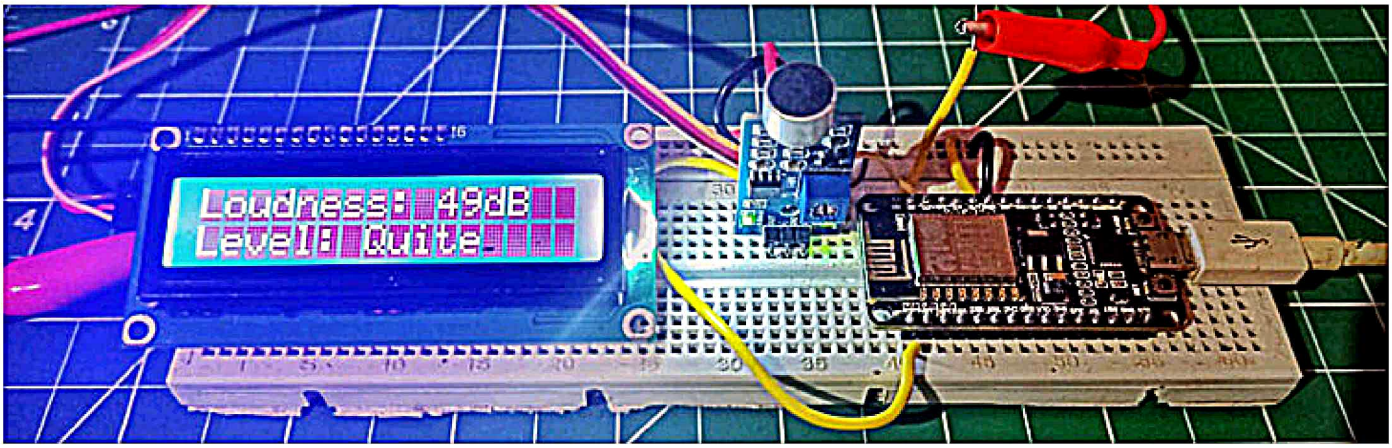
IoT Based Sound Pollution Monitoring System – Measure and Track Decibels (dB) using NodeMCU



We surely can't imagine a world without sound. Sound is one of an integral part of our day to day life, everything just becomes monotonous without the presence of audio. But too much of anything is dangerous, with the advent of automobiles, loudspeakers, etc. sound pollution has become a threat in recent days. So, in this project, we will build an **IoT decibel meter** to measure sound in a particular place and record the value in a graph using IoT. A device like this will be useful in places like hospitals and schools to track and monitor the sound levels and take action accordingly. Previously we have also built an Air pollution meter (https://circuitdigest.com/interview/arjun-natarajan-ceo-of-wiitronics-solutions-on-how-they-develop-iot-based-parking-solutions-to-combat-urban-battle-for-parking-space) to monitor air quality using IoT.

A **sound level meter** is employed for acoustic (sound that travels through the air) measurements. The simplest sort of microphone for sound level meters is the capacitor microphone, which mixes precision with stability and reliability. The diaphragm of the microphone responds to changes in air pressure caused by sound waves. That's why the instrument is usually mentioned as a **sound pressure level (SPL) Meter**.

Sound level meters are commonly utilized in sound pollution studies for the quantification of various sorts of noise, especially for industrial, environmental, mining, and aircraft noise. The reading from a sound level meter doesn't correlate well to human-perceived loudness, which is best measured by a loudness meter. Specific loudness may be a compressive nonlinearity and varies at certain levels and certain frequencies. These metrics also can be calculated in several other ways.



Here we are going to make an **IoT based decibel meter** that will **measure the sound in decibels(dB)** using a sound sensor and display it to the LCD display along with that, it will also be pushing the readings to the **Blynk IoT platform** making it accessible from across the world.

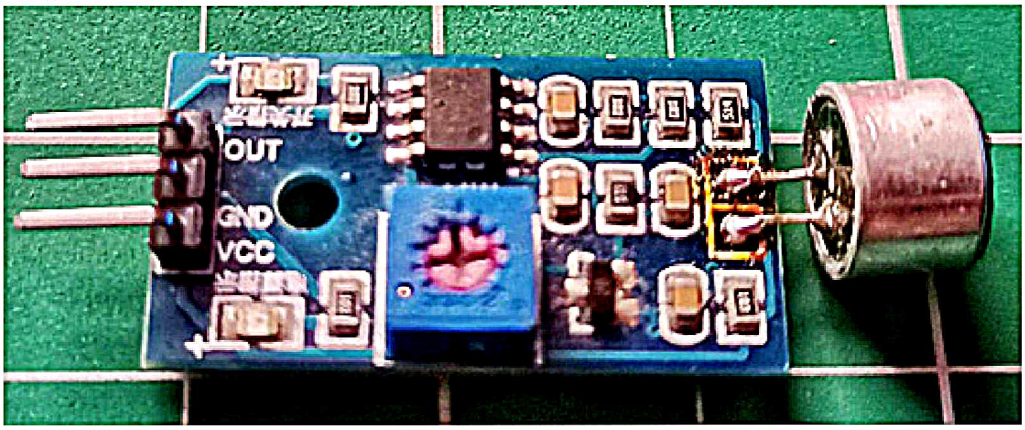
Components Required

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

How does Microphone Module Work?

The microphone based sound sensor is used to detect sound. It gives a measurement of how loud a sound is. The sound sensor module is a small board that mixes a microphone (50Hz-10kHz) and a few processing circuitry to convert sound waves into electrical signals. This electrical signal is fed to on-board **LM393 High Precision Comparator** to digitize it and is made available at the OUT pin.

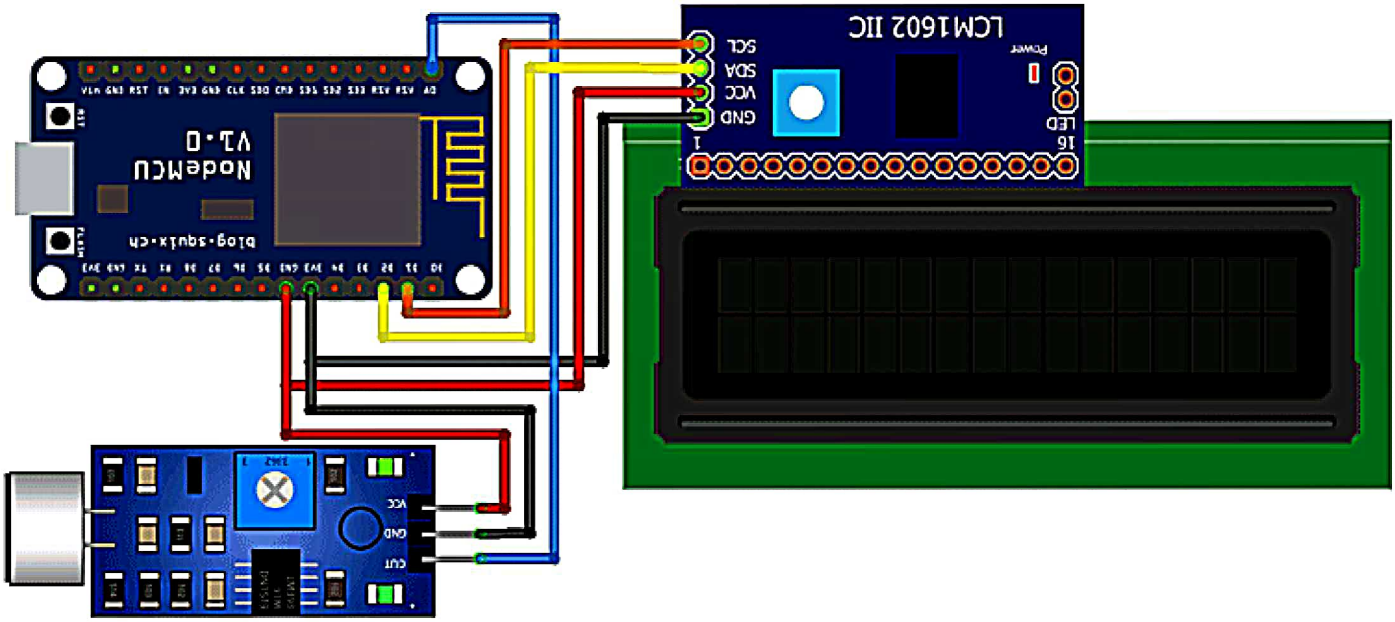
The module features a built-in potentiometer for sensitivity adjustment of the OUT signal. We will set a threshold by employing a potentiometer. So that when the amplitude of the sound exceeds the edge value, the module will output LOW, otherwise, HIGH. Apart from this, the module has two LEDs. The facility LED will illuminate when the module is powered. The Status LED will illuminate when the digital output goes LOW.

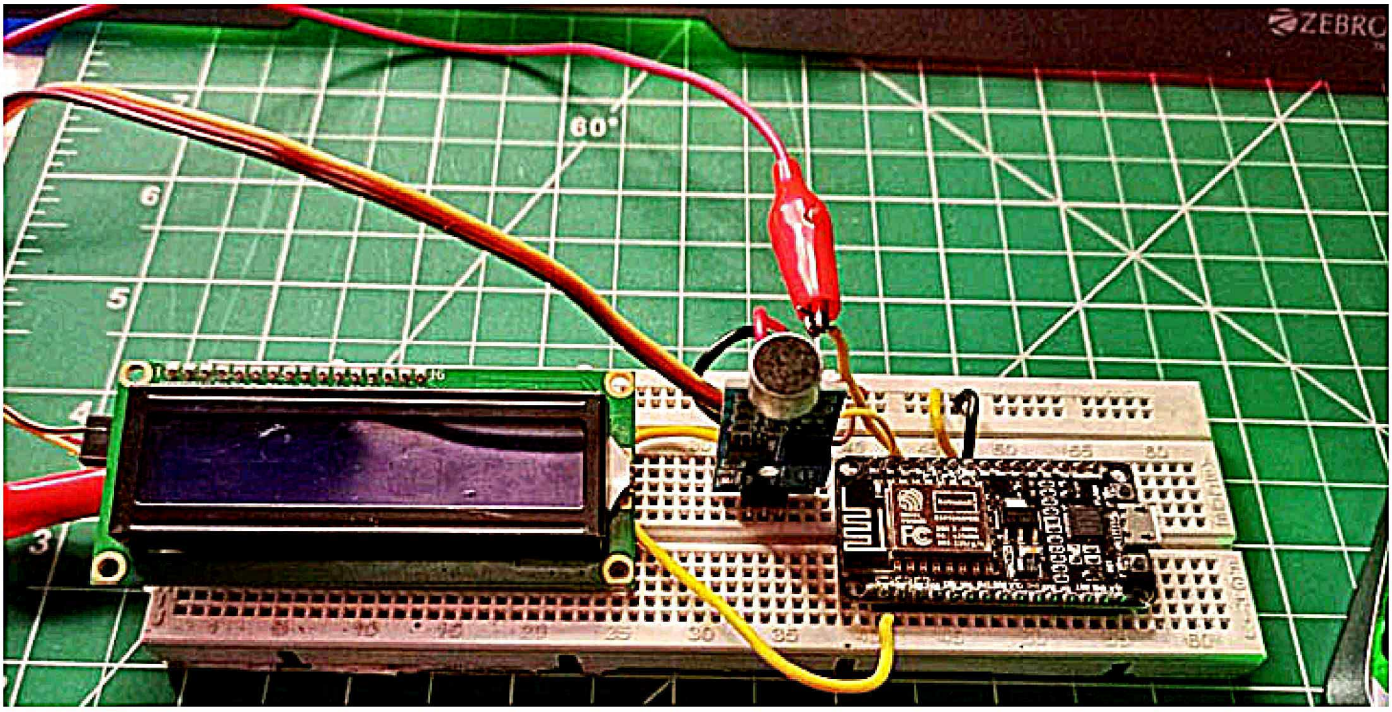


The sound sensor only has three pins: VCC, GND & OUT. VCC pin supplies power for the sensor & works on 3.3V to 5V. OUT pin outputs HIGH when conditions are quiet and goes LOW when sound is detected.

Circuit Diagram for IoT Sound Meter

The connections are pretty simple, we just have to connect the sound sensor to one of the Analog pin and the LCD to the I2C pins.





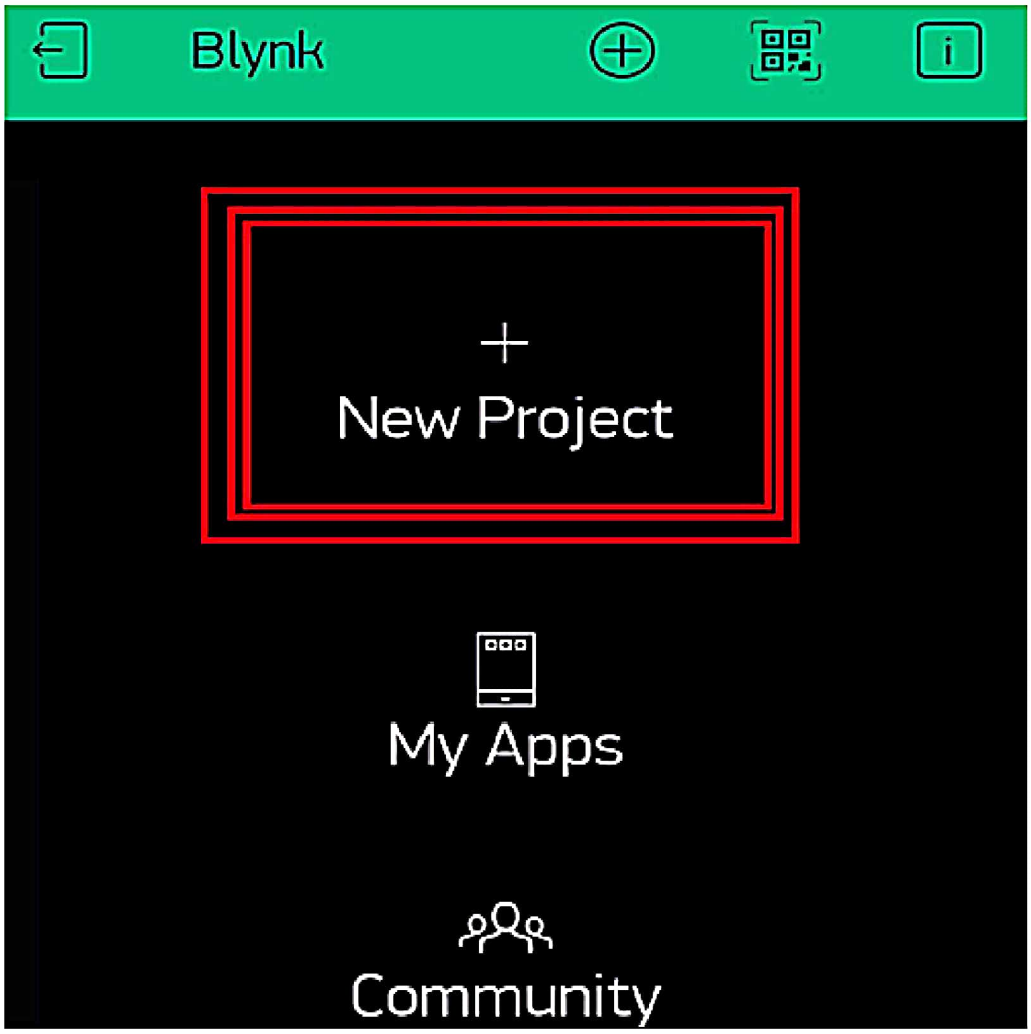
In the above diagram, we have connected the power pins of the sound sensor and LCD display to 3v3 and GND pin of NodeMCU. Along with that, we have also connected the SCL and SDA pins of the module to D1 and D2 respectively, and the OUT pin of the sound sensor to A0 pin.

Setting up Blynk for Remote Monitoring

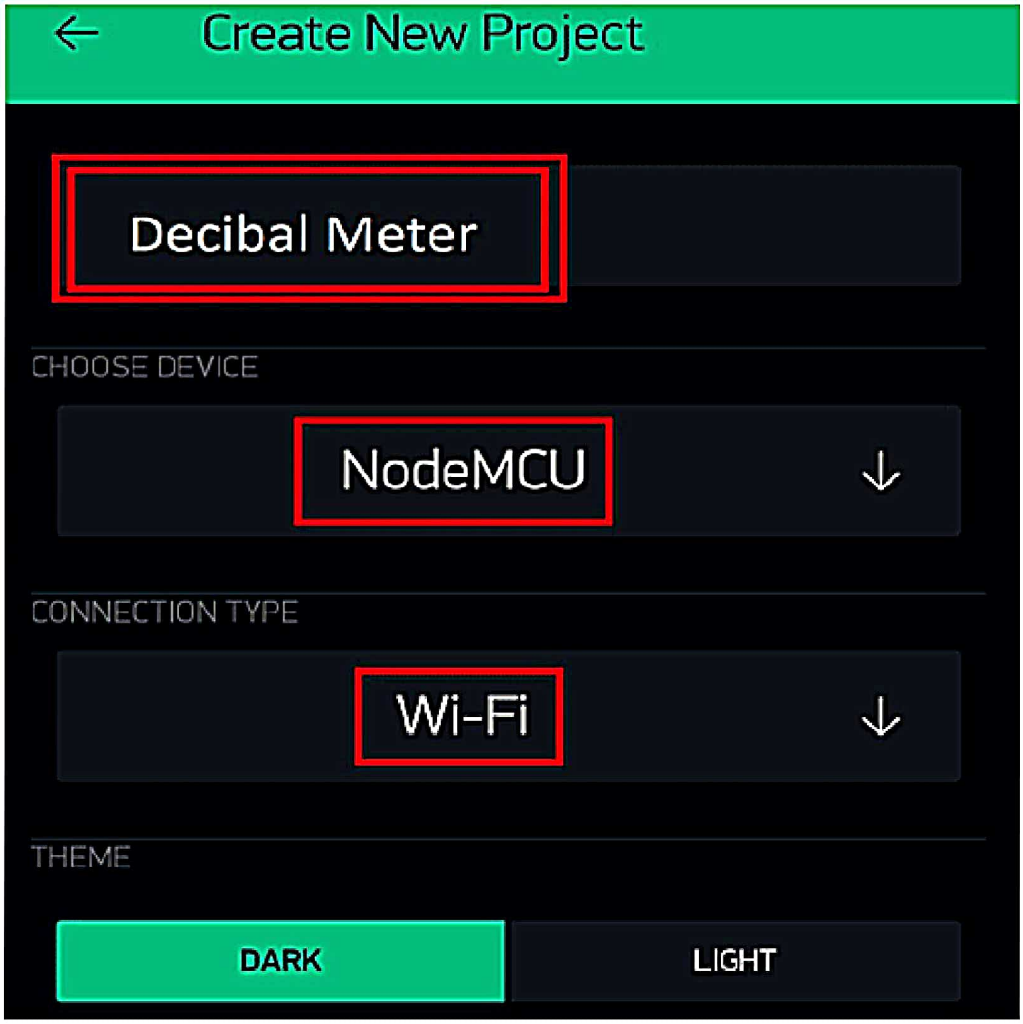
For the IoT part, we will be using the **Blynk IoT platform**. We have previously used Blynk with Nodemcu (https://iotdesignpro.com/projects/iot-controlled-led-using-blynk-and-esp8266-node-mcu) to build many different projects. You can also check out other Blynk projects (https://iotdesignpro.com/projects/iot-controlled-led-using-blynk-and-esp8266-node-mcu) that we have built earlier. In this application, we will be adding a gauge to display the intensity of sound in decibels. Let us set up the app quickly.

- First of all, install the Blynk app from PlayStore and create an account.

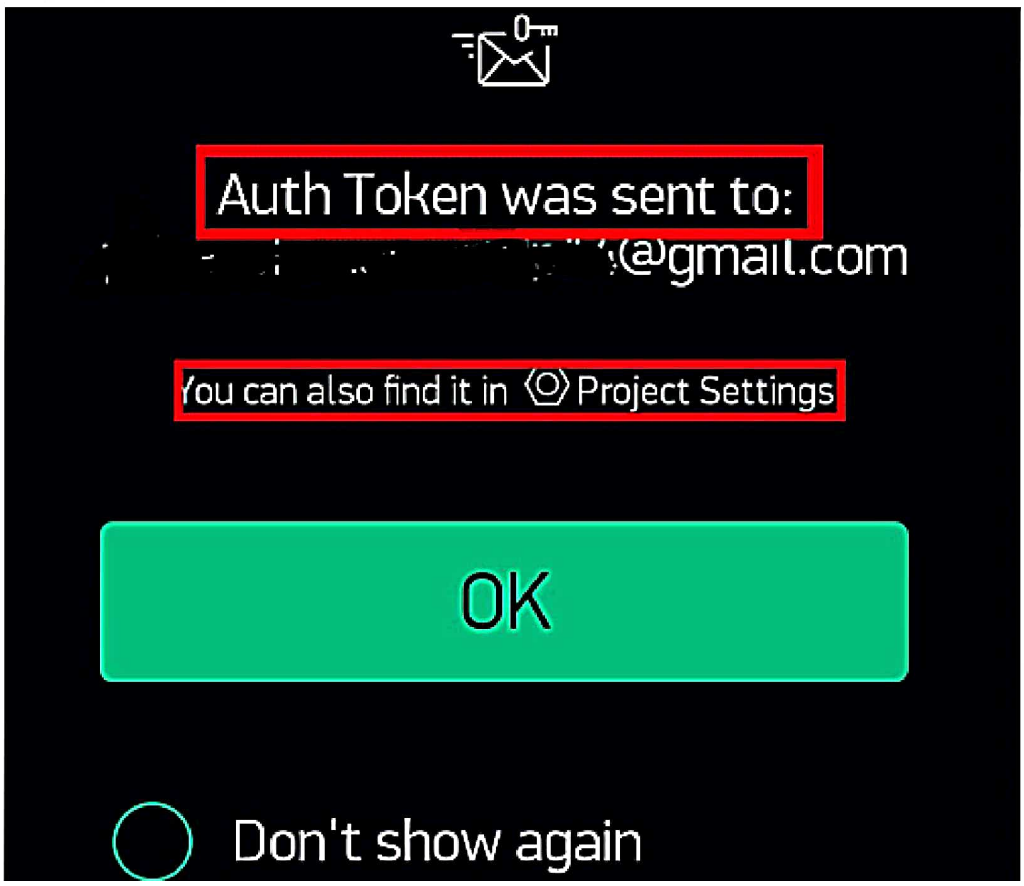
- Click on the create button and create your new project.



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

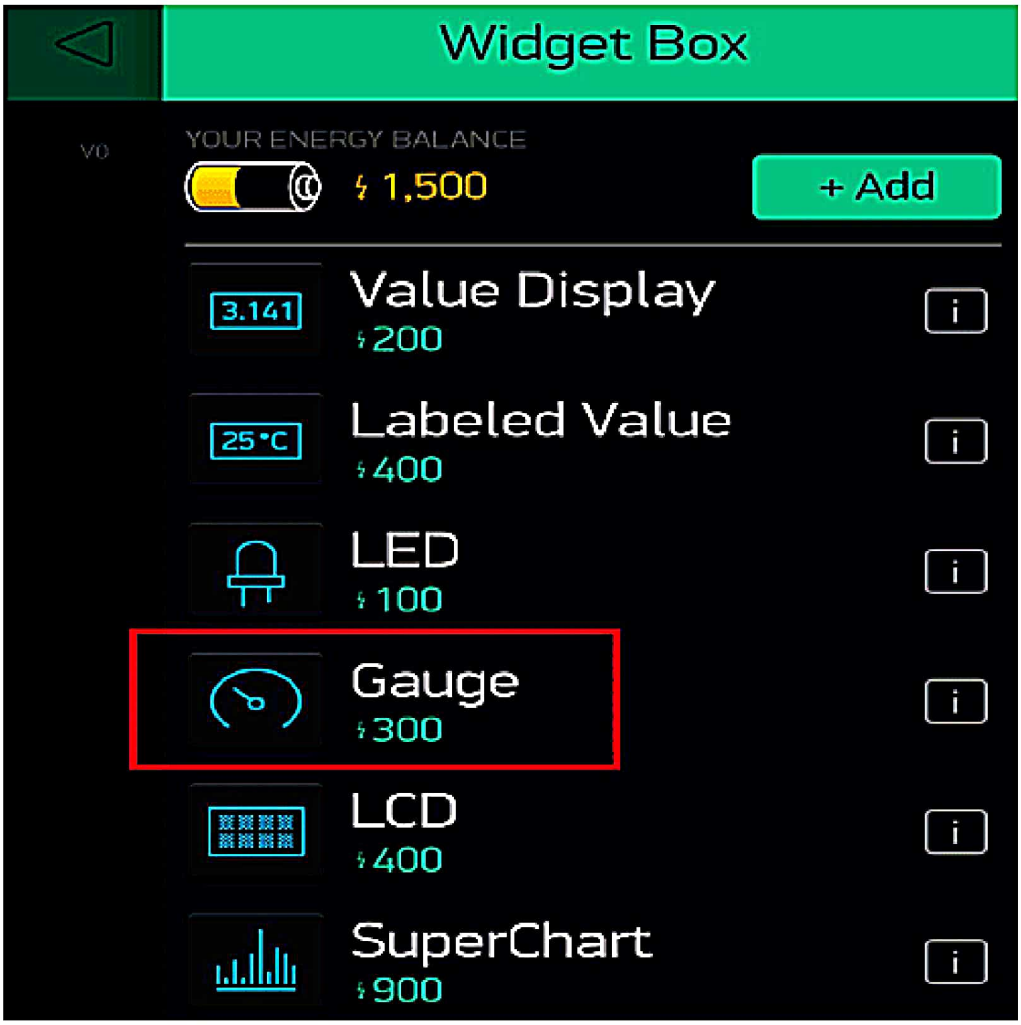


- An auth token will be sent to your registered email id. Keep it safe as it will be used later on while programming.

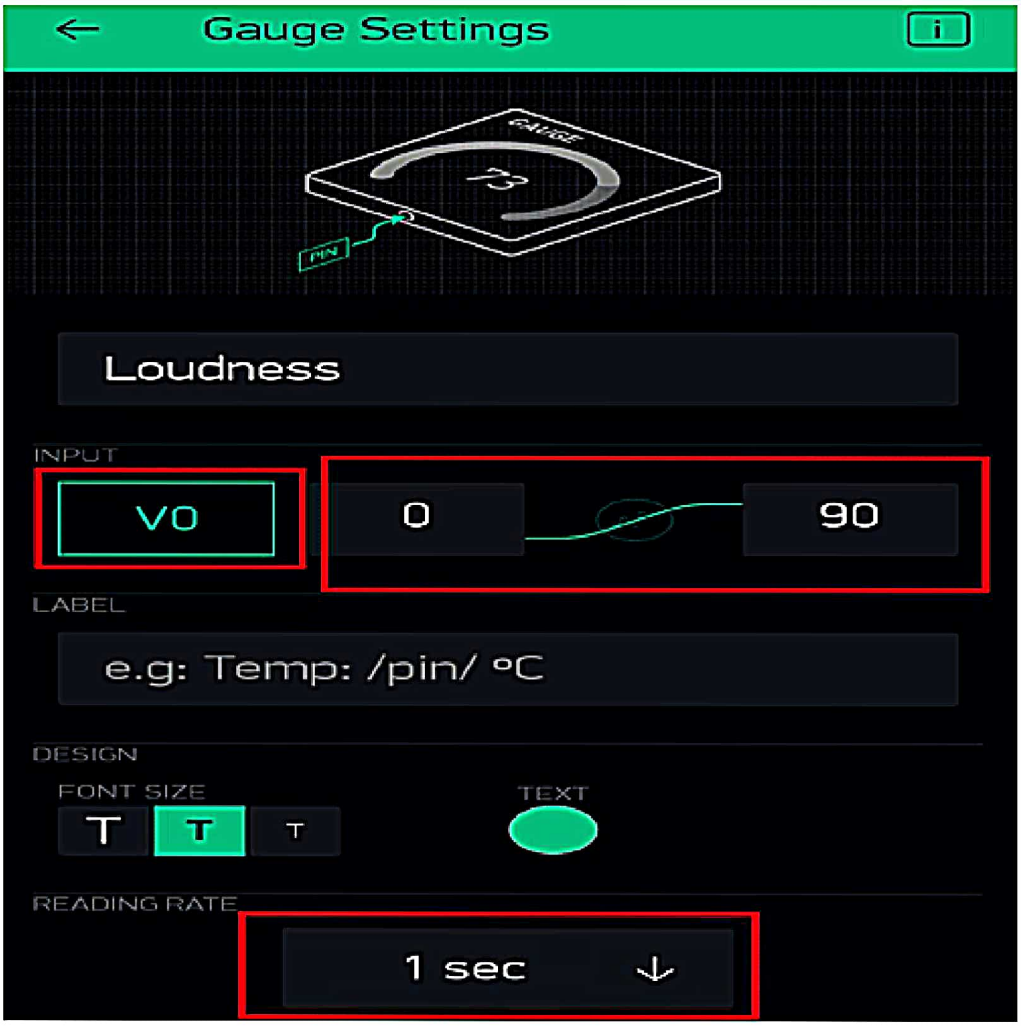


Auth Token is a unique alphanumeric string to identify your project in the Blynk's server and assigning the correct data through it.

- 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.



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

Program for IoT Decibel Meter

Here, we have to develop a code that takes input from the sound sensor and maps it value to decibels and after comparing the loudness, it should not only print it to the 16*2 LCD display but should also send it to the Blynk server.

The complete code for this project can be found at the bottom of this page. You can directly copy-paste it in your IDE and change only three parameters i.e. SSID, pass, and auth token. The explanation of the code is as follows.

In the very first part of the code, we have included all the necessary libraries and definitions. Also, we have defined the necessary variables and objects for further programming.

```
#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[] = "YourSSID";
char pass[] = "YourPass";
```

Further ahead, we have created a Blynk function to handle the virtual pin that our gauge is connected to. We are simply sending the values stored in the dB variable to the V0 pin.

```
BLYNK_READ(V0)
{
  Blynk.virtualWrite(V0, db);
}
```

In the setup part of the code, we are defining the pin mode as input and beginning the LCD display as well as the Blynk function.

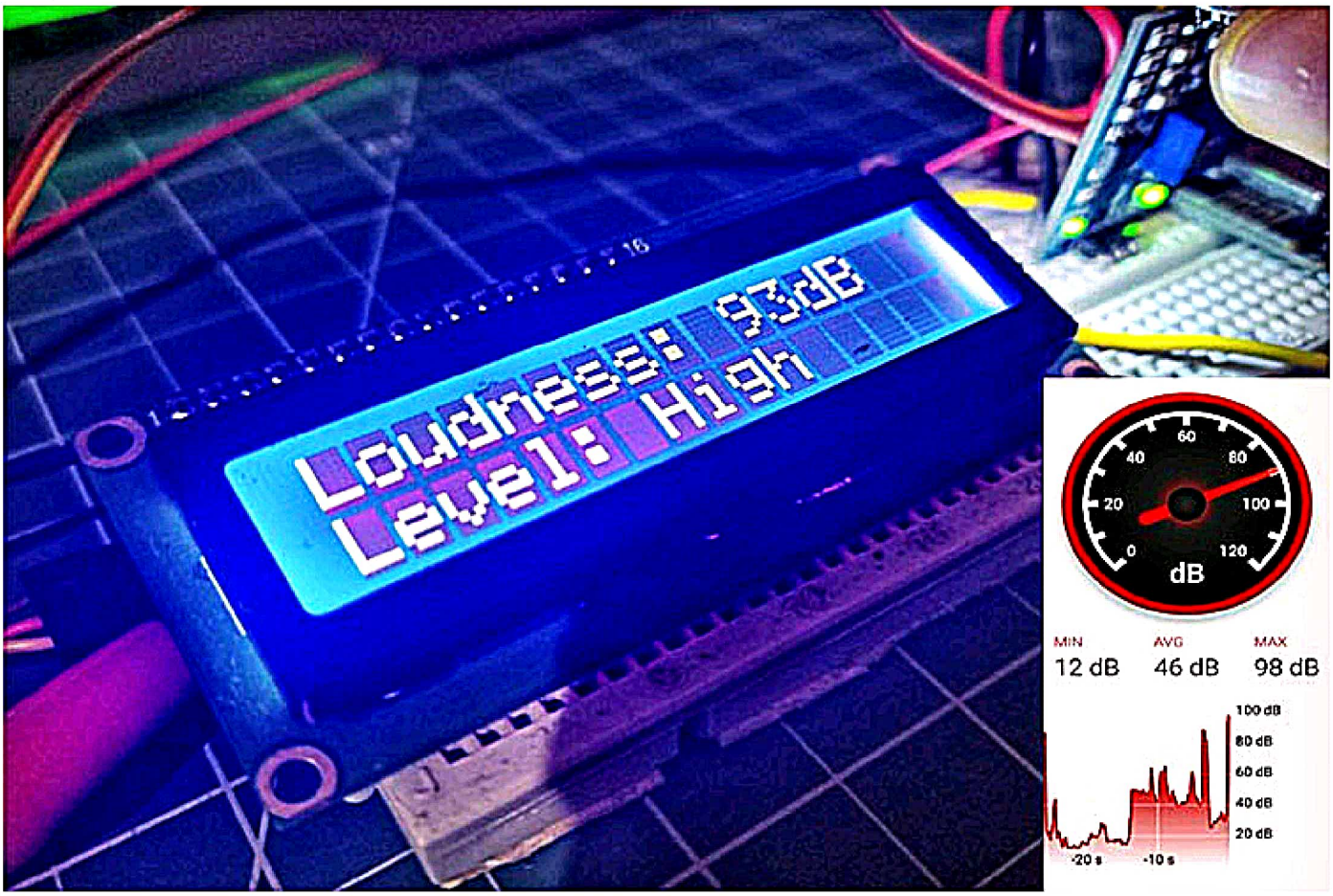
```
void setup() {
  pinMode (SENSOR_PIN, INPUT);
  lcd.begin(16, 2);
  lcd.backlight();
  lcd.clear();
  Blynk.begin(auth, ssid, pass);
}
```

In the loop part, we are doing all the processing tasks like comparison and value assignment along with running the Blynk function.

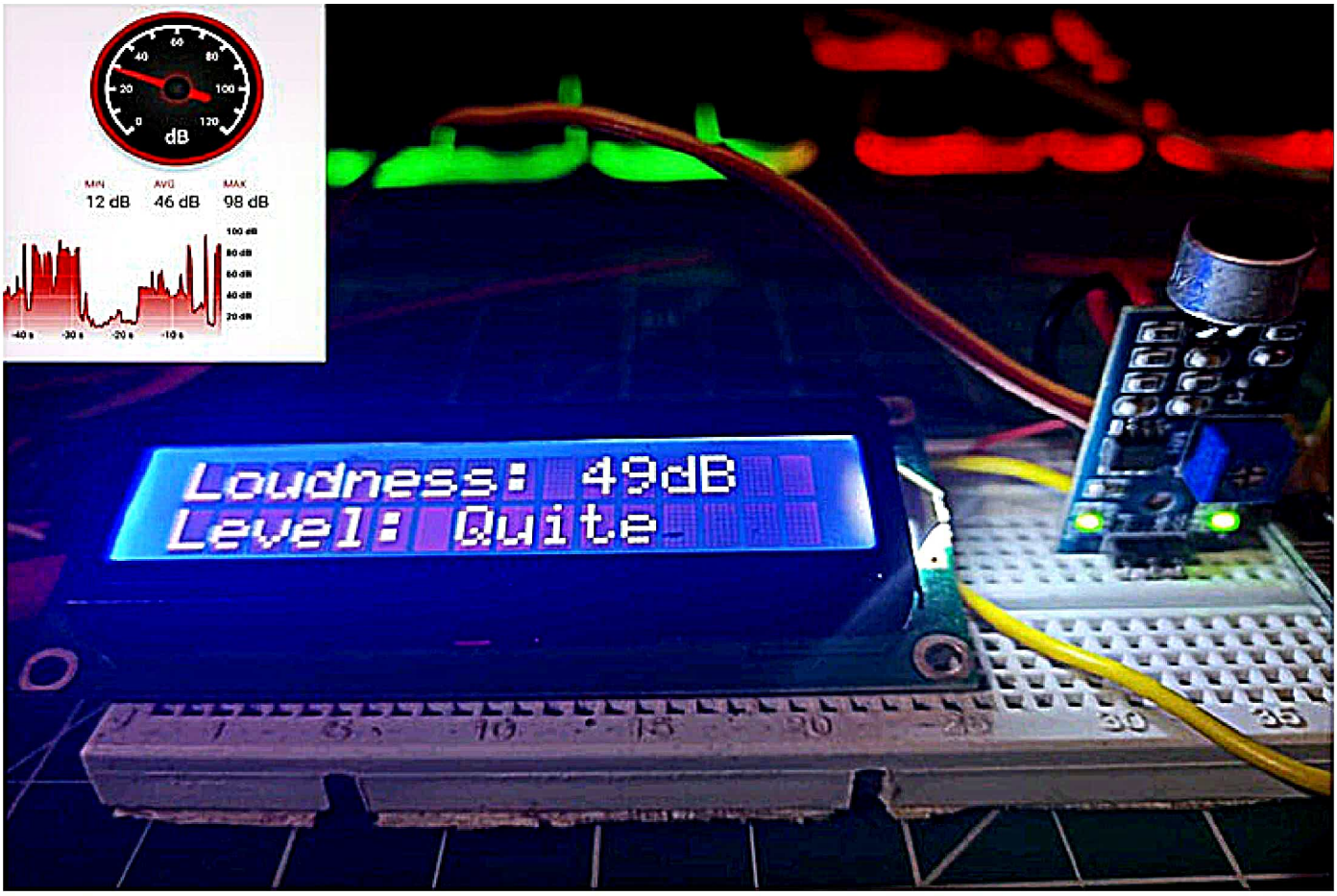
```
void loop() {
  Blynk.run();
  unsigned long startMillis = millis(); // Start of sample window
  float peakToPeak = 0; //peak-to-peak level
  unsigned int signalMax = 0; //minimum value
  unsigned int signalMin = 1024; //maximum value
  // collect data for 50 mS
  while (millis() - startMillis < 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();
```

Working of the Project

Now that you have understood the code, you can simply upload it to your NodeMCU board and the project should start working.



To make sure the values are correct, I compared them to an android application on my phone that could measure sound. As you can see from the pictures, the results were quite close.



The complete working of this project is also demonstrated in the video linked below. Hope you enjoyed the project and learned something useful if you have any questions, leave them in the comment section below.

Code

```
#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.clea ();
```

```
  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();
}
```