Phase 3: Development Part_1

Noise Pollution Monitoring

Components Required :

- Node MCU 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.

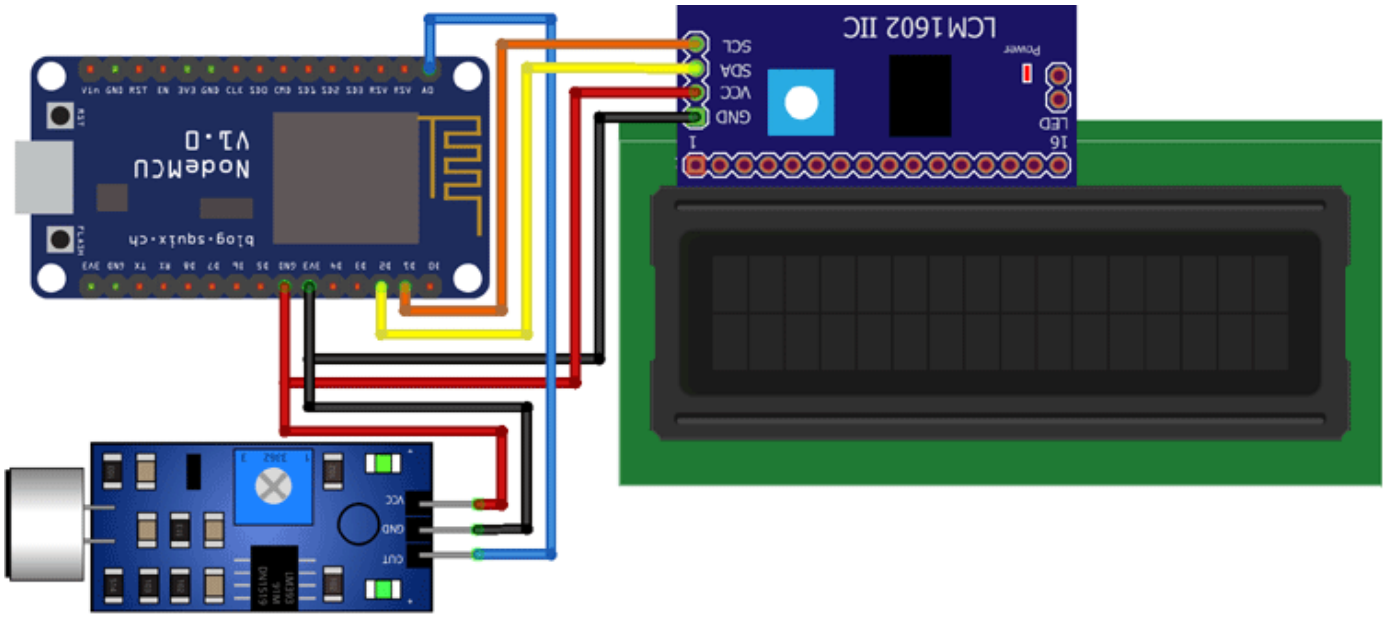
The sound sensor only has three pins: VCC, GND & OUT. VCC pin supplies power for the sensor & works on 3.3V to 5V. OUT pin outputs HIGH when conditions are quiet and goes LOW when sound is detected.

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.

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.

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.

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.

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. 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

Python 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.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();
}
```