Phase 4: Development Part_2

Noise Pollution Monitoring

COLLEGE CODE: 5113
COLLEGE NAME: Kingston Engineering College
DOMAIN: Internet of Things
PROJECT TITLE: Noise Pollution Monitoring System

PROJECT MEMBERS:	NAN MUDHALVAN Id
D Navin(Leader)	511321104062
D Namdev	511321104060
B Purushoth	511321104076
H Sai Jaswanth	511321104083
T Vamsi Krishna	511321104108

Components Required :

- Node MCU Board
- Microphone sensor
- 16*2 LCD Module
- Breadboard
- Connecting wires

How does the 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 the onboard 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.

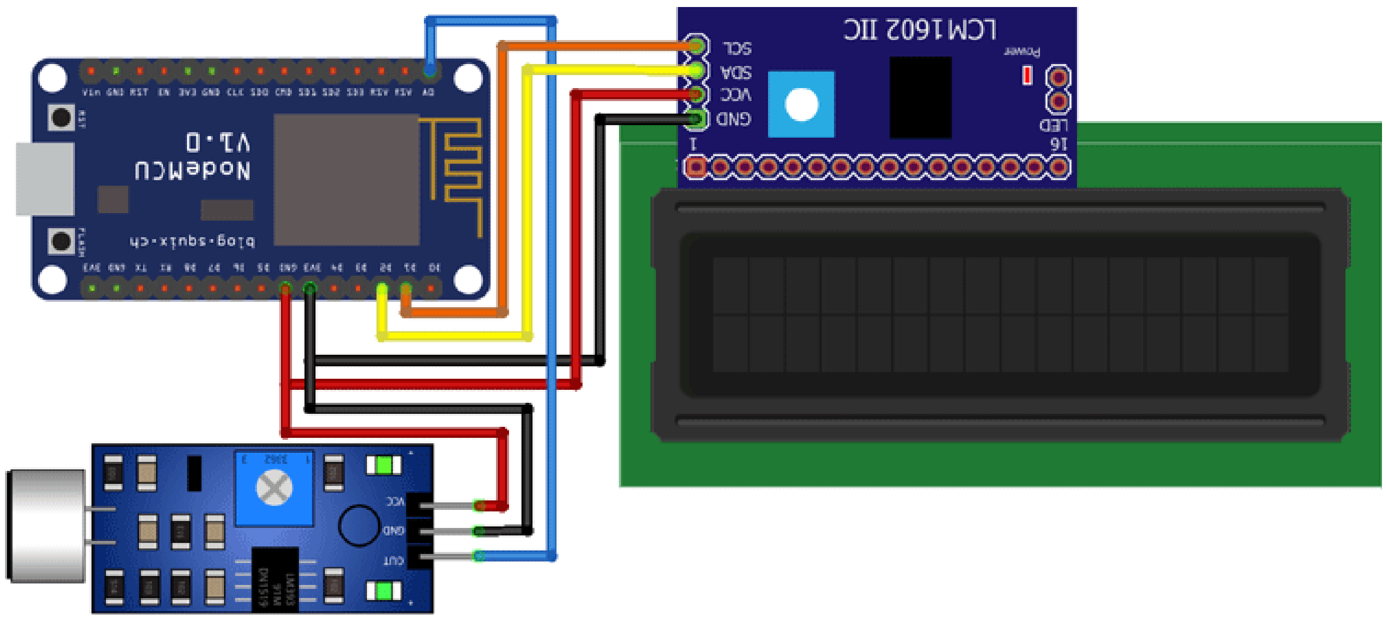
The sound sensor only has three pins: VCC, GND & OUT. VCC pin supplies power for the sensor & works on 3.3V to 5V. OUT pin outputs HIGH when conditions are quiet and goes LOW when sound is detected.

Working on 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 pins 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. 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 its value to decibels and after comparing the loudness, it should print it to the 16*2 LCD display and send it to the Blynk server.

The complete code for this project can be found at the bottom of this page. You can directly copypaste 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 first part of the code, we 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

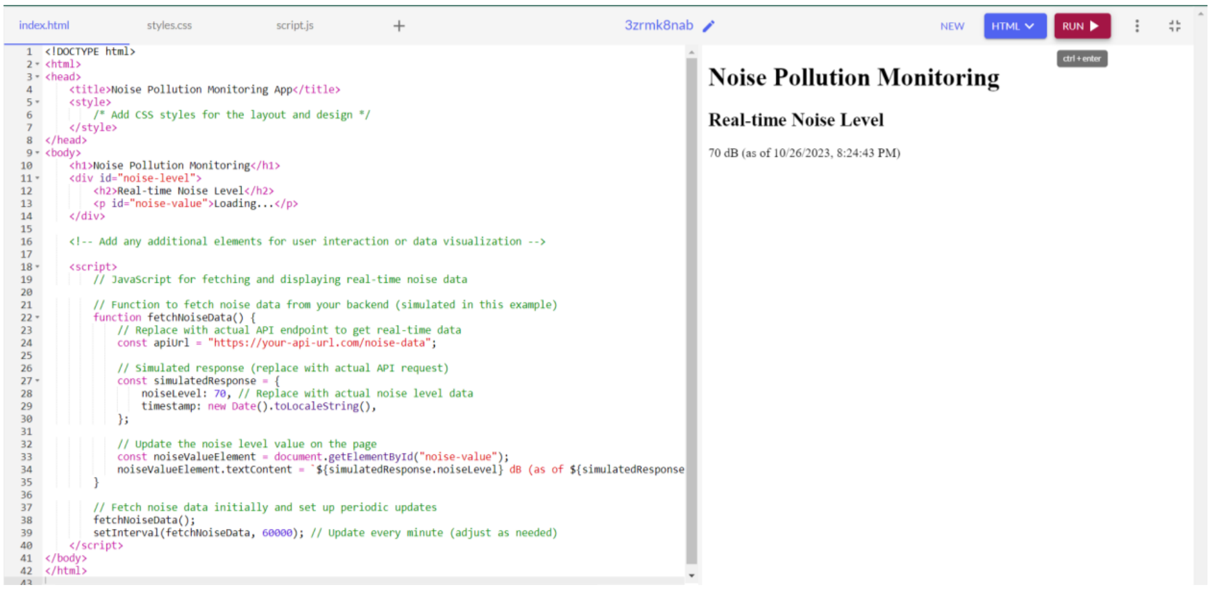
Html Code:

```
<!DOCTYPE html>
<html>
<head>
  <title>Noise Pollution Monitoring App</title>
  <style>
    /* Add CSS styles for the layout and design */
  </style>
</head>
<body>
  <h1>Noise Pollution Monitoring</h1>
  <div id="noise-level">
    <h2>Real-time Noise Level</h2>
    <p id="noise-value">Loading...</p>
  </div>

  <!-- Add any additional elements for user interaction or data visualization -->

  <script>
    // JavaScript for fetching and displaying real-time noise data
```

```
        // Function to fetch noise data from your backend (simulated in this example)
        function fetchNoiseData() {
            // Replace with actual API endpoint to get real-time data
            const apiUrl = "https://your-api-url.com/noise-data";

            // Simulated response (replace with actual API request)
            const simulatedResponse = {
                noiseLevel: 70, // Replace with actual noise level data
                timestamp: new Date().toLocaleString(),
            };

            // Update the noise level value on the page
            const noiseValueElement = document.getElementById("noise-value");
            noiseValueElement.textContent = `${simulatedResponse.noiseLevel} dB (as of 
${simulatedResponse.timestamp})`;
        }

        // Fetch noise data initially and set up periodic updates
        fetchNoiseData();
        setInterval(fetchNoiseData, 60000); // Update every minute (adjust as needed)
    </script>
</body>
</html>
```

Output:

```
Loudness: 49dB
Level: Quite
```

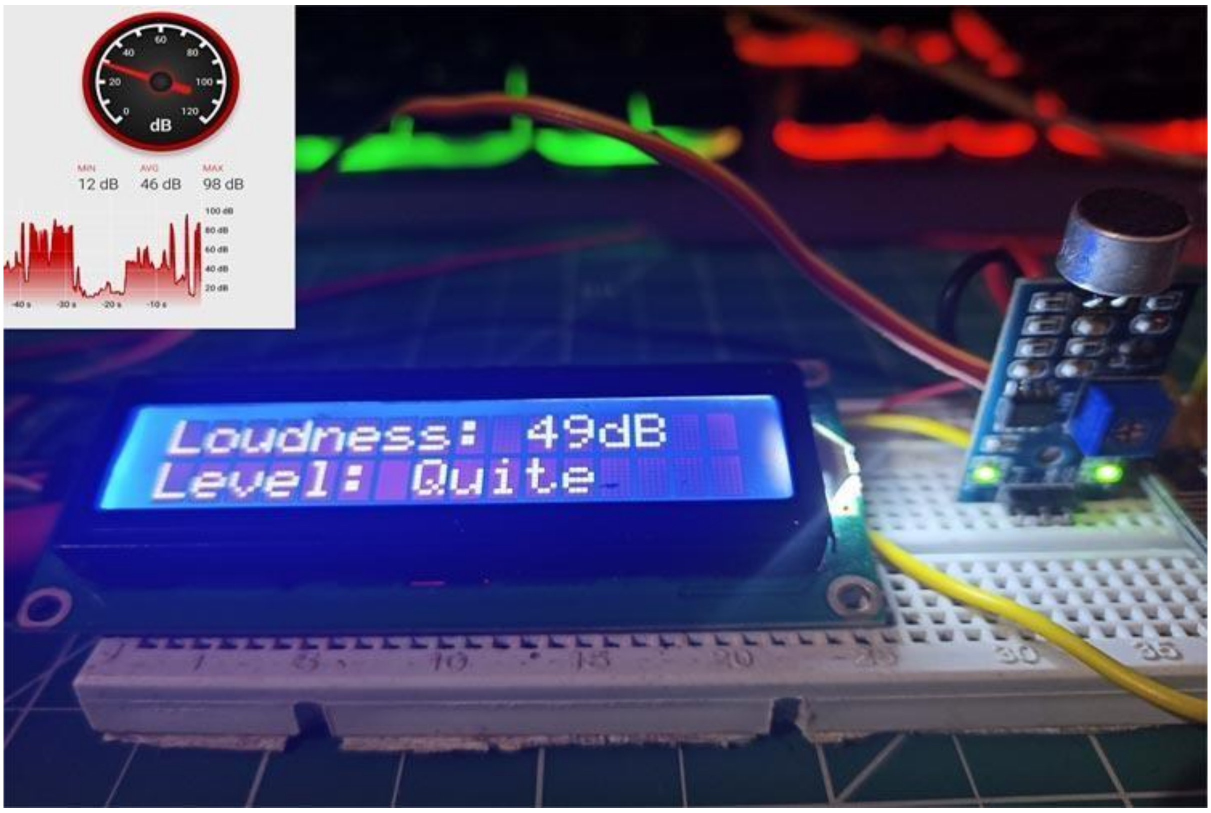


```html
<!DOCTYPE html>
<html>
<head>
    <title>Noise Pollution Monitoring App</title>
    <style>
        /* Add CSS styles for the layout and design */
    </style>
</head>
<body>
    <h1>Noise Pollution Monitoring</h1>
    <div id="noise-level">
        <h2>Real-time Noise Level</h2>
        <p id="noise-value">Loading...</p>
    </div>

    <!-- Add any additional elements for user interaction or data visualization -->

    <script>
        // JavaScript for fetching and displaying real-time noise data

        // Function to fetch noise data from your backend (simulated in this example)
        function fetchNoiseData() {
            // Replace with actual API endpoint to get real-time data
            const apiUrl = "https://your-api-url.com/noise-data";

            // Simulated response (replace with actual API request)
            const simulatedResponse = {
                noiseLevel: 70, // Replace with actual noise level data
                timestamp: new Date().toLocaleString(),
            };

            // Update the noise level value on the page
            const noiseValueElement = document.getElementById("noise-value");
            noiseValueElement.textContent = `${simulatedResponse.noiseLevel} dB (as of ${simulatedResponse
        }

        // Fetch noise data initially and set up periodic updates
        fetchNoiseData();
        setInterval(fetchNoiseData, 60000); // Update every minute (adjust as needed)
    </script>
</body>
</html>
```

Noise Pollution Monitoring

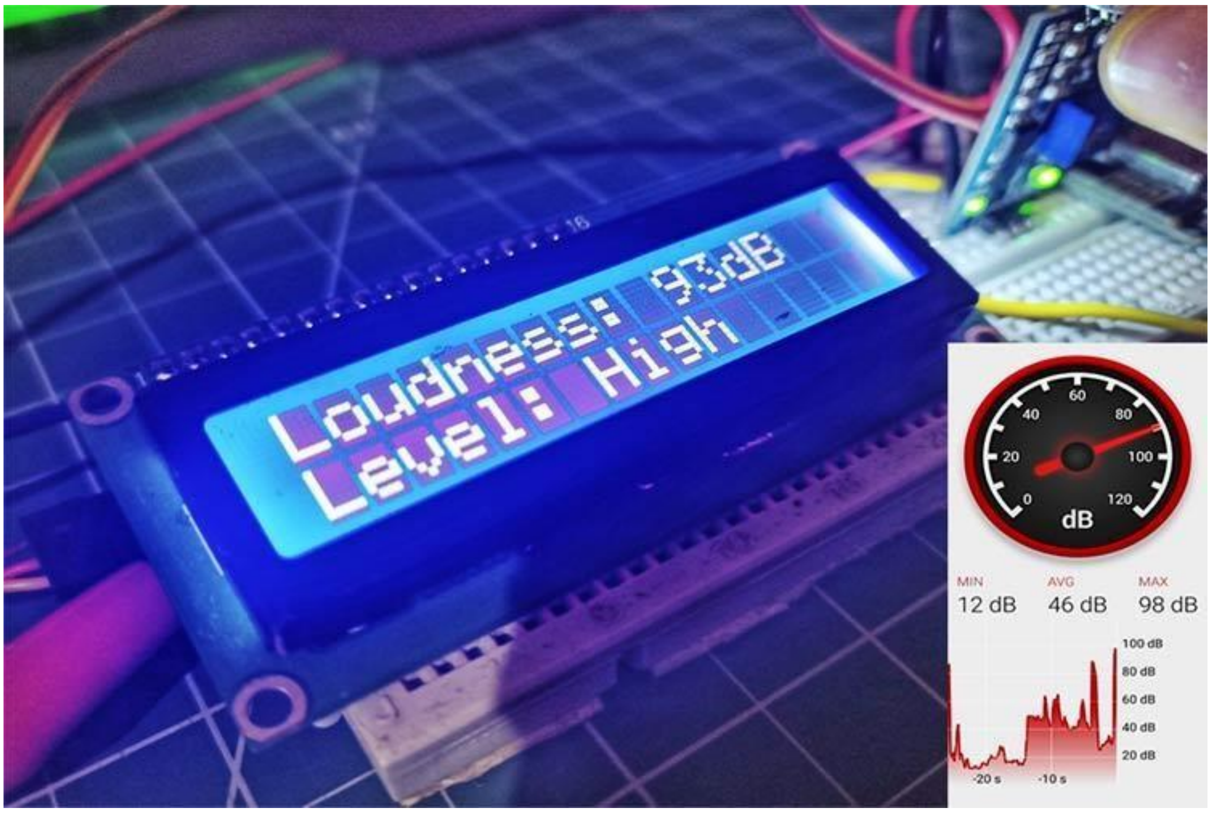
Real-time Noise Level

70 dB (as of 10/26/2023, 8:24:43 PM)



Loudness: 93dB

Level: High


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