Phase 4: Development Part\_2

Noise Pollution Monitoring

**Abstract:**

The project aims to address the critical issue of noise pollution by developing a comprehensive noise pollution information platform and mobile applications for iOS and Android.Using web development technologies such as HTML, CSS, and JavaScript, the platform will display real-time noise level data collected from IoT sensors. Simultaneously, mobile apps will be designed to provide users with instant access to real-time noise level updates. This project serves the dual purpose of raising awareness about noise pollution and empowering informed decision-making among the public.

**Modules for Real-time noise level data:**

1. \*\*Platform Development:\*\*

**- Use HTML, CSS, and JavaScript for the frontend to create the user interface of the platform.**

**- Choose a backend technology (e.g., Python with a framework like Django or Flask) to handle data processing and storage.**

**- Implement a database to store real-time noise data.**

**- Develop APIs for communication between the frontend and backend.**

**- Integrate real-time data from IoT sensors and display it on the platform.**

**- Ensure a user-friendly interface with graphs or charts to visualize noise levels.**

**2. \*\*Mobile App Development:\*\***

**- For iOS, you'll use Swift and for Android, Java or Kotlin are common choices. Alternatively, consider cross-platform frameworks like React Native or Flutter for building both apps simultaneously.**

**- Develop the mobile app's UI to provide access to real-time noise level updates.**

**- Implement features for user registration, login, and user settings.**

**- Use APIs to fetch data from the platform.**

**- Design notifications or alerts for noise level thresholds.**

**- Ensure the app's compatibility with various device screen sizes and resolutions.**

**3. \*\*Testing and Quality Assurance:\*\***

**- Thoroughly test the platform and mobile apps for functionality and user experience.**

**- Conduct real-world testing with the IoT sensors to ensure data accuracy.**

**- Address any bugs or issues that arise during testing.**

**4. \*\*Deployment:\*\***

**- Deploy the platform on a web server.**

**- Publish the mobile apps on the Apple App Store and Google Play Store.**

**5. \*\*Maintenance and Updates:\*\***

**- Regularly maintain the platform and apps to ensure they continue to function correctly.**

**- Provide updates to add new features, improve performance, and fix bugs.**

**6. \*\*User Support and Feedback:\*\***

**- Offer customer support for users who encounter issues or have questions.**

**- Encourage user feedback to make improvements based on their needs.**

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.

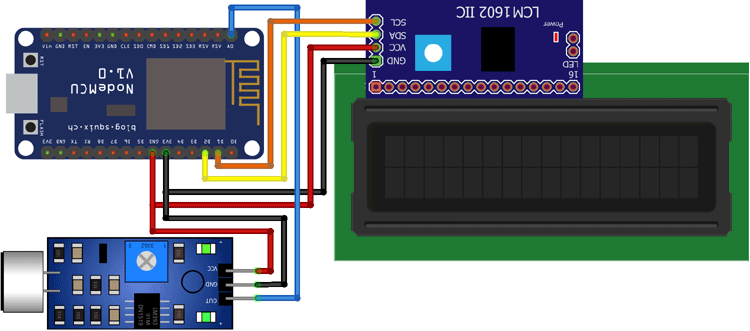
-The sound sensor only has three pins: VCC, GND & OUT. VCC pin supplies power for the sensor & works on 3.3V to 5V. OUT pin outputs HIGH when conditions are quiet and goes LOW when sound is detected.

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 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:

Import unrequests

Import machine

Import time

# Wi-Fi and server settings

WIFI\_SSID = “Samsung Galaxy A70”

WIFI\_PASSWORD = “Agalya123@ssid”

SERVER\_URL =http://agalya.com/api/noise-data

# Define the pin connected to the noise sensor

NOISE\_SENSOR\_PIN = 34

# Function to read noise level from the sensor

Def read\_noise\_level():

Return machine.ADC(NOISE\_SENSOR\_PIN).read()

# Function to send data to the server

Def send\_noise\_data(data):

Headers = {‘Content-Type’: ‘application/json’}

Payload = ‘{“noise\_level”: ‘ + str(data) + ‘}’

Response = urequests.post(SERVER\_URL, data=payload, headers=headers)

Response.close()

# Connect to WiFi

Import network

Wifi = network.WLAN(network.STA\_IF)

If not wifi.isconnected():

Print(“Connecting to WiFi...”)

Wifi.active(True)

Wifi.connect(WIFI\_SSID, WIFI\_PASSWORD)

While not wifi.isconnected():

Pass

Print(“Connected to WiFi”)

# Main loop

While True:

Noise\_level = read\_noise\_level()

Print(“Loudness:”, noise\_level)

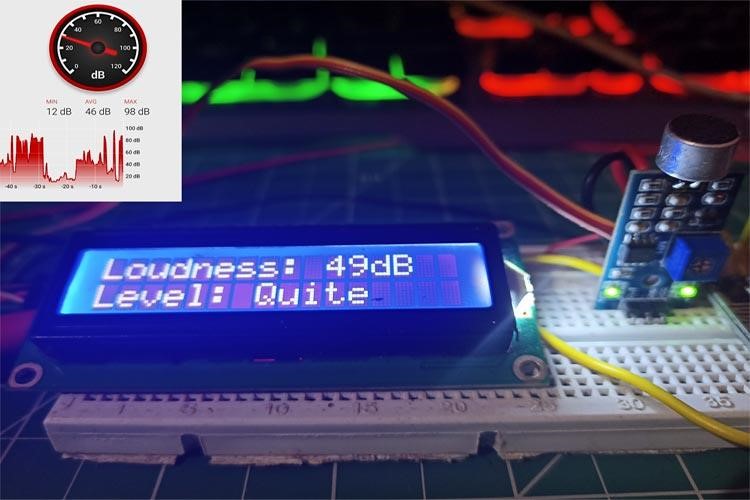
Send\_noise\_data(noise\_level)

Time.sleep(60) # Send data every minute

**Output:**

Loudness: 49dB

Level: Quite



Loudness: 93dB

Level: High



**Conclusion:**

The development of a noise pollution information platform and mobile app is a valuable project that can have a positive impact on the health and well-being of individuals and communities. By using web development technologies and mobile app development technologies, we can create a platform and mobile apps that are accessible, convenient, and informative.