**FINAL PROJECT DOCUMENTATION**

**TEAM NAME:** proj\_228488\_Team\_1

**TEAM MEMBERS**

Aravindan p( 210621106006)

Ajai M(210621106003)

Avinash k(210621106008)

Bathmanaban v(210621106009)

**Noise pollution monitoring**

**PROBLEM DEFINITION AND DESIGN THINKING**

Noise pollution monitoring is essential for assessing and addressing the impact of excessive noise on the environment and human health Here are some key components of noise pollution monitoring:

Design Thinking Approach:

1. Empathize:
   * Understand the needs and concerns of the community affected by noise pollution.
   * Gather feedback from stakeholders such as residents, local authorities, and environmental experts.
2. Define:
   * Clearly define the objectives of the monitoring system: What data do you need to collect, and what are the specific goals of noise reduction?
   * Identify the key performance indicators (KPIs) for success, such as noise level thresholds and response times.
3. Ideate:
   * Brainstorm potential solutions for noise monitoring, including sensor types (e.g., microphones, sound level meters), data collection methods, and data visualization techniques.
   * Explore innovative technologies like machine learning for noise source identification.
4. Prototype:
   * Develop a prototype of the monitoring system to test its feasibility and functionality.
   * Consider the hardware and software components required, as well as the user interface for data access.
5. Test:
   * Conduct field tests to assess the accuracy and reliability of the monitoring system.
   * Gather feedback from users and make necessary improvements based on their input.
6. Implement:
   * Deploy the noise pollution monitoring system in target areas, ensuring proper installation and calibration of sensors.
   * Establish data management procedures and protocols for reporting noise levels.
7. Monitor and Iterate:
   * Continuously monitor the system’s performance and gather real-time data.
   * Use the collected data to analyze noise trends, identify problem areas, and adjust noise mitigation strategies accordingly.
8. Collaborate:
   * Collaborate with local authorities, environmental agencies, and community groups to share data and coordinate noise reduction efforts.
9. Educate and Engage:
   * Raise awareness about noise pollution and its impact on health and quality of life.
   * Engage the community in noise reduction initiatives and encourage responsible noise behavior.
10. Evaluate:
    * Periodically evaluate the effectiveness of noise reduction measures based on the data collected.
    * Adjust strategies and technologies as needed to achieve the desired noise reduction goals.

Remember that noise pollution monitoring should be conducted ethically, respecting privacy and the rights of individuals. Collaborating with experts in acoustics and environmental science can enhance the quality and credibility of your monitoring efforts.

**Innovation**

Innovations in noise pollution monitoring have the potential to improve the accuracy and efficiency of data collection and analysis. Here are some innovative approaches and technologies in this field:

\*\*1. Smart Noise Sensors:\*\*

- Develop compact, AI-powered noise sensors with built-in noise recognition capabilities.

- These sensors should be self-calibrating and capable of identifying specific noise sources.

\*\*2. IoT Integration:\*\*

- Implement an IoT (Internet of Things) architecture for seamless connectivity and data transfer.

- Sensors should communicate with a central server in real-time through low-power, long-range networks like LoRaWAN or NB-IoT.

\*\*3. Artificial Intelligence and Machine Learning:\*\*

- Utilize AI and ML algorithms to process data at the edge and in the cloud.

- Train models to distinguish between different types of noise sources (e.g., traffic, construction, wildlife).

\*\*4. Predictive Analytics:\*\*

- Develop predictive models to forecast noise pollution trends and potential hotspots.

- Use historical data and real-time inputs for accurate predictions.

\*\*5. Mobile App Interface:\*\*

- Create a user-friendly mobile app for real-time access to noise data.

- Include features like noise maps, personal noise exposure tracking, and the ability to report noise disturbances.

\*\*6. Noise Signature Analysis:\*\*

- Use AI to analyze noise signatures and match them with known noise sources.

- Provide detailed information about identified sources.

\*\*7. Public Engagement:\*\*

- Foster community involvement by allowing users to contribute data and observations.

- Gamify noise reduction efforts to encourage participation.

\*\*8. Remote Calibration:\*\*

* Implement remote calibration for sensors, reducing maintenance costs and downtime.

\*\*9. Noise Pollution Index:\*\*

- Create a Noise Pollution Index (NPI) that quantifies noise pollution in a standardized manner.

- This index can be used for comparisons and compliance assessment.

\*\*10. Noise Reduction Suggestions:\*\*

* Provide users with noise reduction recommendations based on their specific noise exposure patterns.

\*\*11. Integration with Urban Planning:\*\*

* Collaborate with urban planners to incorporate noise data into city development and zoning decisions.

\*\*12. Data Monetization:\*\*

* Explore monetization options by providing aggregated, anonymized data to researchers, urban planners, or businesses interested in noise-related insights.

\*\*13. Security and Privacy:\*\*

* Prioritize data security and privacy, ensuring user data is protected.

\*\*14. Sustainable Power Sources:\*\*

* Implement energy-efficient sensors and explore sustainable power sources like solar panels and energy harvesting.

\*\*15. Regulatory Compliance:\*\*

* Ensure that the system complies with local noise pollution regulations and standards.

These innovations can lead to more comprehensive and effective noise pollution monitoring, which in turn can help in developing better policies, strategies, and technologies for noise reduction and mitigation

**DEVELOPMENT PART 1**

Creating a full simulation for noise pollution monitoring is a complex task that typically requires specialized software development.

1. \*\*Hardware Setup:\*\*

- Choose suitable IoT noise sensors (e.g., microphones) and microcontrollers (e.g., Raspberry Pi or Arduino).

- Assemble the hardware components and connect the sensors to the microcontroller.

2. \*\*Software Development:\*\*

- Develop a Python script for the microcontroller to capture noise level data from the sensors.

- Implement data preprocessing and filtering to ensure accurate readings.

- Use libraries like `RPi.GPIO` for Raspberry Pi or `Adafruit\_CircuitPython` for Adafruit boards to interface with sensors.

3. \*\*Data Transmission:\*\*

- Set up communication protocols (e.g., MQTT, HTTP, or WebSocket) to send data from the sensors to the central platform.

4. \*\*Central Platform:\*\*

- Create a cloud-based or local server to receive and store the incoming noise data.

- Develop a web application or API for data visualization and analysis.

- You can use frameworks like Flask or Django for web development and databases like PostgreSQL or MongoDB to store data.

5. \*\*Real-time Data Processing:\*\*

- Implement real-time data processing for immediate insights.

- You may want to use tools like Apache Kafka or RabbitMQ for message queuing and processing.

6. \*\*User Interface:\*\*

- Create a user-friendly dashboard to visualize noise pollution data.

- Use HTML, CSS, and JavaScript for the frontend, and libraries like D3.js for data visualization.

7. \*\*Alerts and Notifications:\*\*

- Add alerting mechanisms to notify relevant authorities or the public when noise levels exceed predefined thresholds.

8. \*\*Security:\*\*

- Ensure data security and authentication measures to protect the system from unauthorized access.

9. \*\*Power Management:\*\*

- Optimize power usage for IoT devices, especially if they run on batteries.

10. \*\*Testing and Calibration:\*\*

- Thoroughly test the system, calibrate sensors, and validate data accuracy.

11. \*\*Deployment:\*\*

- Deploy IoT sensors in public areas, ensuring they are well-protected from environmental conditions.

- Monitor and maintain the system to ensure its continued functionality.

Certainly, here’s a basic example of a Python script for IoT sensors to send real-time noise level data to a server using MQTT (Message Queuing Telemetry Transport) as a communication protocol. This example uses the `paho-mqtt` library for MQTT communication. Ensure you have the library installed on your IoT device.

Python

Import machine

Import time

Import urequests

Import ujson

Import network

Import math

# Define your Wi-Fi credentials

Wifi\_ssid = ‘Wokwi-GUEST’

Wifi\_password = ‘’ # Replace with the actual Wi-Fi password

# Connect to Wi-Fi

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

Wifi.active(True)

Wifi.connect(wifi\_ssid, wifi\_password)

# Wait for Wi-Fi connection

While not wifi.isconnected():

Pass

# Define ultrasonic sensor pins (Trig and Echo pins)

Ultrasonic\_trig = machine.Pin(15, machine.Pin.OUT)

Ultrasonic\_echo = machine.Pin(4, machine.Pin.IN)

# Define microphone pin

Microphone = machine.ADC(2)

Calibration\_constant = 2.0

Noise\_threshold = 60 # Set your desired noise threshold in dB

# Firebase Realtime Database URL and secret

Firebase\_url = ‘https://noise-pollution-bd0ab-default-rtdb.asia-southeast1.firebasedatabase.app/’ # Replace with your Firebase URL

Firebase\_secret = ‘nBsgyQFTqHUe4qExlaZX6VL3mpf5gn6BlpnMiuR0’ # Replace with your Firebase secret

Def measure\_distance():

# Trigger the ultrasonic sensor

Ultrasonic\_trig.value(1)

Time.sleep\_us(10)

Ultrasonic\_trig.value(0)

# Measure the pulse width of the echo signal

Pulse\_time = machine.time\_pulse\_us(ultrasonic\_echo, 1, 30000)

# Calculate distance in centimeters

Distance\_cm = (pulse\_time / 2) / 29.1

Return distance\_cm

Def measure\_noise\_level():

# Read analog value from the microphone

Noise\_level = microphone.read()

Noise\_level\_db = 20 \* math.log10(noise\_level / calibration\_constant)

Return noise\_level, noise\_level\_db

# Function to send data to Firebase

Def send\_data\_to\_firebase(distance, noise\_level\_db):

Data = {

“Distance”: distance,

“NoiseLevelDB”: noise\_level\_db

}

url = f’{firebase\_url}/sensor\_data.json?auth={firebase\_secret}’

try:

response = urequests.patch(url, json=data) # Use ‘patch’ instead of ‘put’

if response.status\_code == 200:

print(“Data sent to Firebase”)

else:

print(f”Failed to send data to Firebase. Status code: {response.status\_code}”)

except Exception as e:

print(f”Error sending data to Firebase: {str€}”)

try:

while True:

distance = measure\_distance()

noise\_level, noise\_level\_db = measure\_noise\_level()

print(“Distance: {} cm, Noise Level: {:.2f} dB”.format(distance, noise\_level\_db))

if noise\_level\_db > noise\_threshold:

print(“Warning: Noise pollution exceeds threshold!”)

# Send data to Firebase

Send\_data\_to\_firebase(distance, noise\_level\_db)

Time.sleep(1) # Adjust the sleep duration as needed

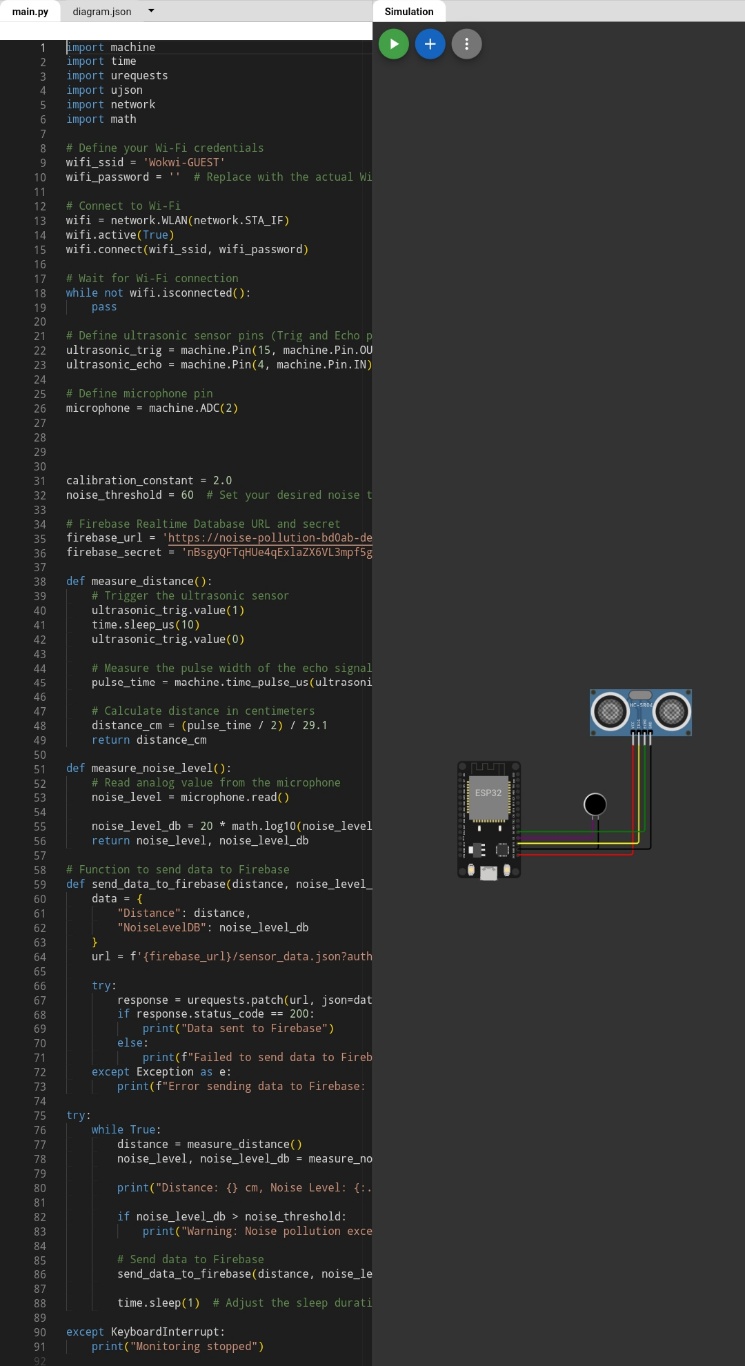
Except KeyboardInterrupt:

Print(“Monitoring stopped”)

**Simulation link:**

[**https://wokwi.com/projects/378892802870919169**](https://wokwi.com/projects/378892802870919169)

**SIMULATION OUTPUT**



CONCLUSION

In conclusion, developing a simulation for noise pollution monitoring is a comprehensive and technologically advanced endeavor. This simulation combines elements of 3D modeling, data generation, real-time analysis, and user interaction to provide a virtual environment for studying and visualizing noise pollution. Such a simulation can serve multiple purposes, including research, education, and policy development.By simulating various noise sources, sensor networks, and environmental conditions, the simulation allows users to gain insights into the spatial and temporal dynamics of noise pollution. It can be a valuable tool for understanding how different factors contribute to noise levels and exploring potential solutions for noise mitigation.

**DEVELOPMENT PART 2**

**NOISE POLLUTION MONITORING APP**

Creating a noise pollution monitoring app would be a valuable project. We would need to incorporate features like real-time noise level measurements, location tagging, and data visualization. We can Consider using a smartphone's built-in microphone and GPS capabilities for data collection. Additionally, you could allow users to report noise disturbances and provide educational resources on noise pollution.

**Features:**

**Real-time Noise Level Monitoring**

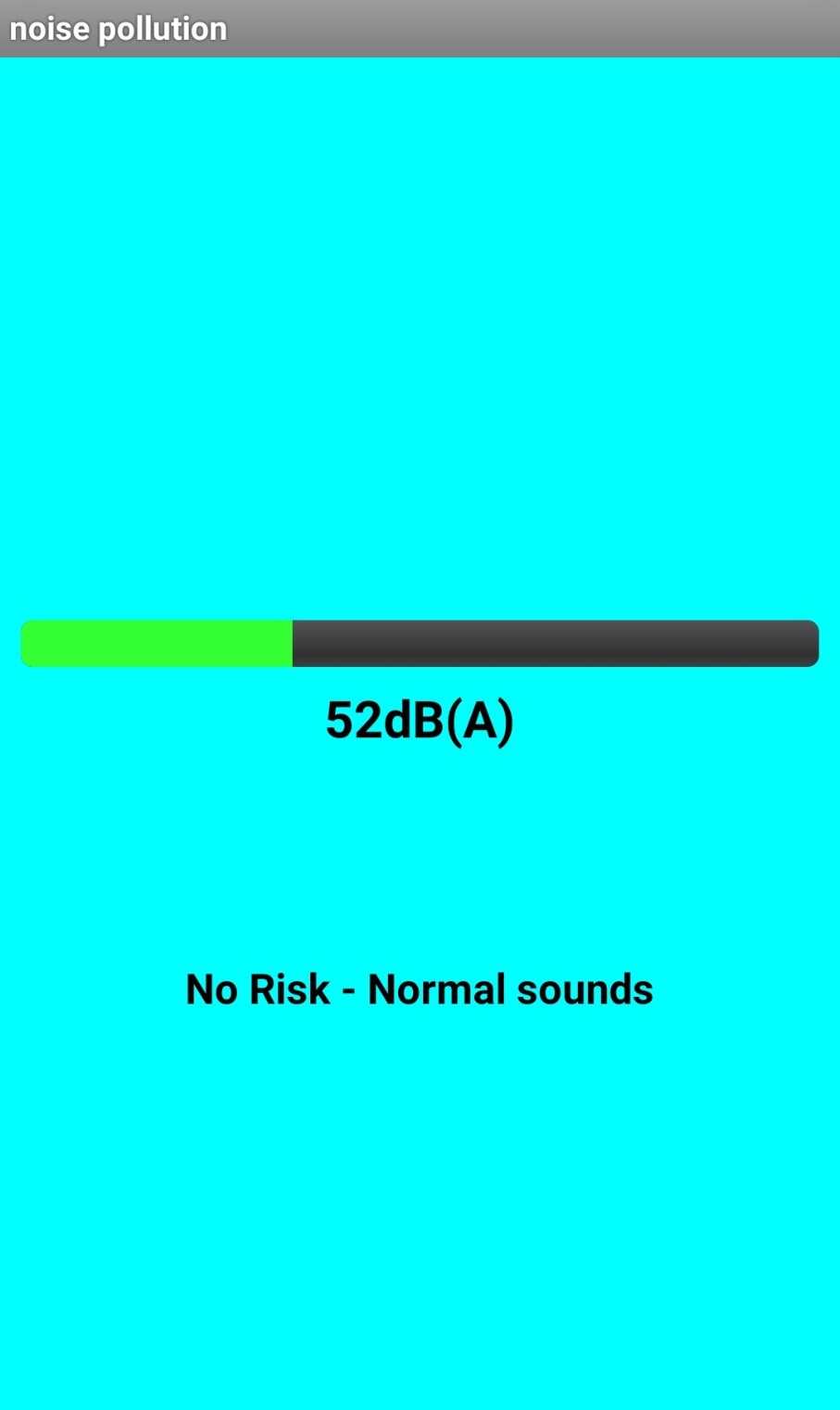
Utilize the smartphone's microphone to measure and display current noise levels in decibels (dB).Provide a continuous real-time graph to show noise level fluctuations.

**Noise Disturbance Reporting:**

Enable users to report noise disturbances by recording audio snippets and providing descriptions.

**Development Considerations:**

Choose a suitable platform for app development (iOS, Android, or both).Utilize sound processing algorithms to accurately measure noise levels.Implement data storage and management, considering scalability.Secure user data and ensure privacy compliance.Test the app extensively to ensure accurate noise measurements and user-friendly functionality.



**WEB CODE**

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Real-Time Noise Level Data</title>

<script>

// Function to update the noise level

function updateNoiseLevel() {

// Simulate real-time data (replace with actual data retrieval)

const noiseLevel = Math.floor(Math.random() \* 101); // Random noise level between 0 and 100 dB

// Update the displayed noise level

document.getElementById('noise-level').textContent = `Noise Level: ${noiseLevel} dB`;

}

// Update the noise level every 5 seconds (adjust as needed)

setInterval(updateNoiseLevel, 5000);

// Initial update

updateNoiseLevel();

</script>

</head>

<body>

<h1>Real-Time Noise Level Data</h1>

<div id="noise-data">

<p id="noise-level">Loading...</p>

</div>

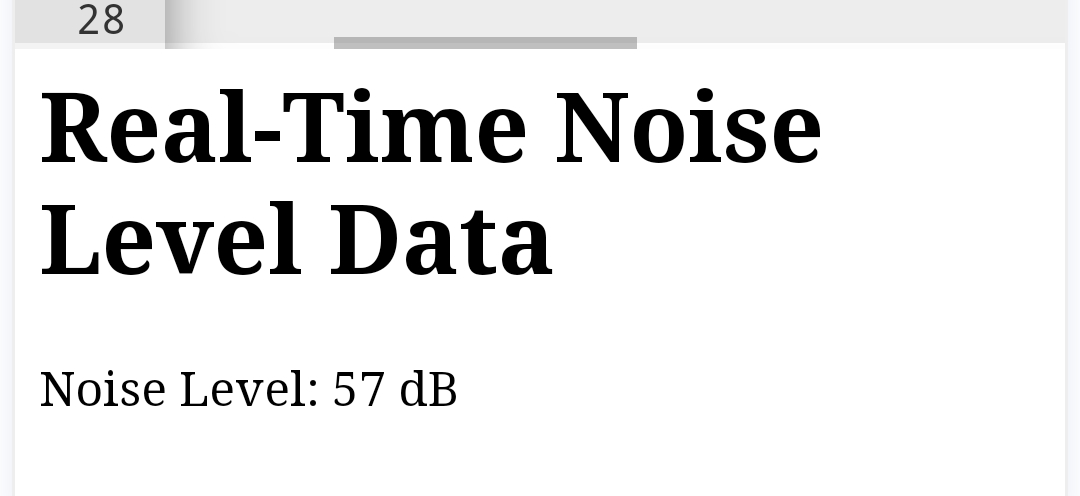
</body>

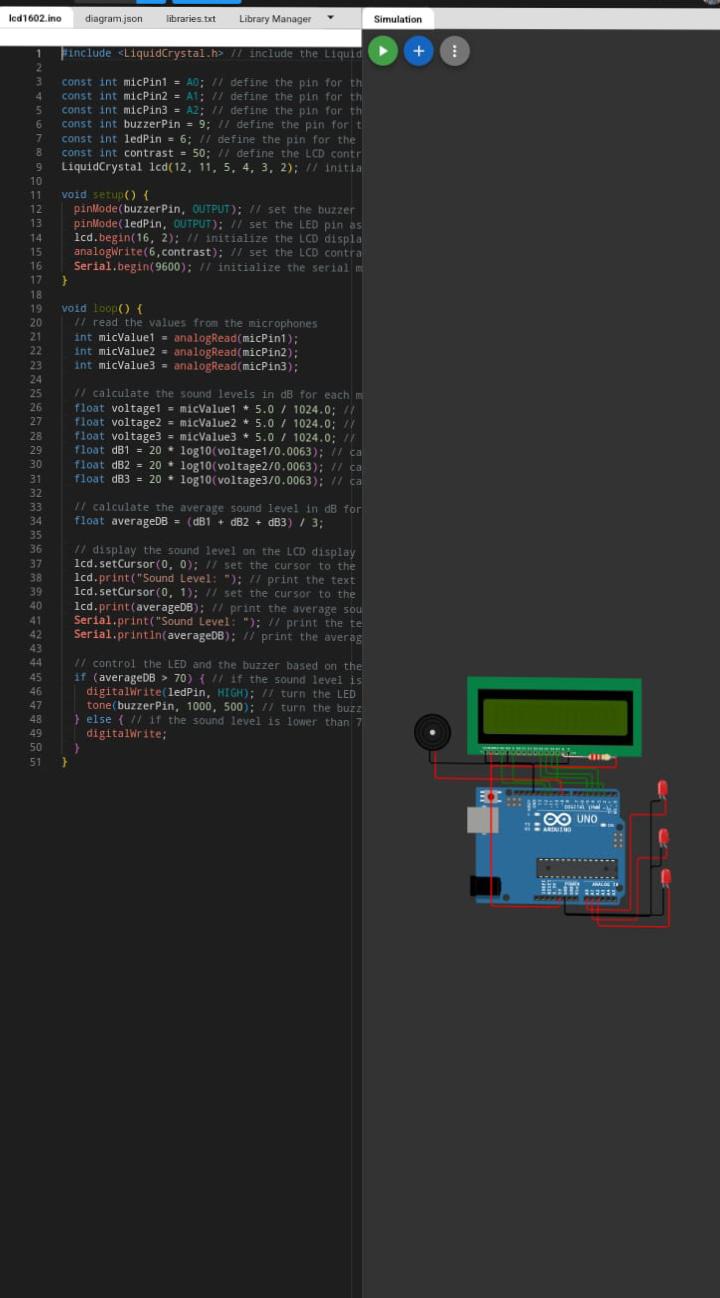
</html>

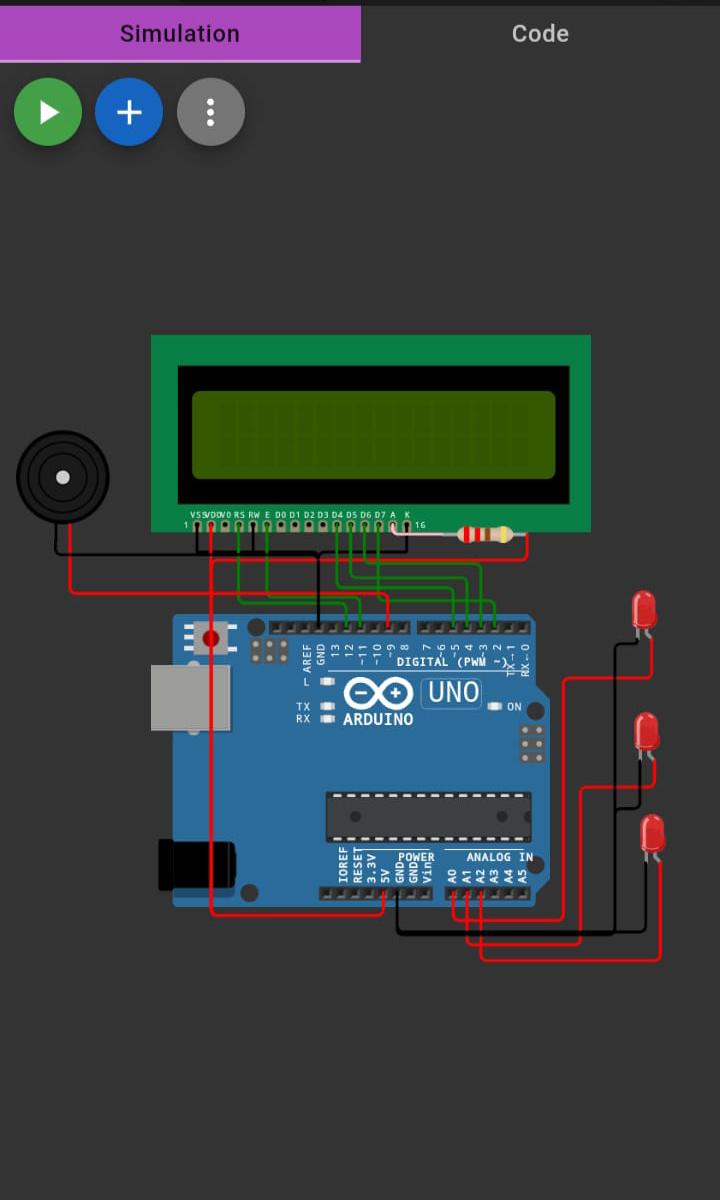
The above code will display the real time noise level data.

Link : https://drive.google.com/drive/folders/1YjhPlc9ZAk867RiJxe7JytnUMnwMsZus

**OUTPUT**

****





To create a noise pollution monitoring system using Arduino, you will need a set of components and sensors. Here is a list of key components you may require:

1. \*\*Arduino Board\*\*: Choose an Arduino board that suits your needs. The Arduino Uno is a popular choice for many projects, but you can select boards like the Arduino Mega or Arduino Nano based on your requirements.

2. \*\*Sound Sensor\*\*: Use a sound sensor, like a microphone or a sound level sensor, to capture audio data. An Electret Microphone or a Sound Detection Module can be suitable options.

3. \*\*Amplifier (Optional)\*\*: Depending on the sensitivity of your sound sensor, you may need an amplifier to boost the signal for accurate readings.

4. \*\*Display\*\* (Optional): To provide real-time feedback, you can include an LCD screen or LED display to show noise levels.

5. \*\*Data Storage\*\*: You can use an SD card module or an external device to log and store the collected noise data.

6. \*\*Power Supply\*\*: Choose an appropriate power source, such as batteries, a power adapter, or even solar panels, to power your Arduino and components.

7. \*\*Enclosure\*\*: Protect your electronics from environmental factors with a weatherproof enclosure.

8. \*\*Cables and Connectors\*\*: Ensure you have the necessary cables, wires, and connectors to link all components together.

9. \*\*Calibrated Sound Source (for calibration)\*\*: To calibrate your system, you'll need a calibrated sound source or a reference sound level meter.

10. \*\*Resistors and Capacitors\*\*: Depending on your sensor and amplifier, you may need resistors and capacitors for signal conditioning.

11. \*\*Breadboard or PCB\*\*: Use a breadboard for prototyping, or design a custom PCB for a more permanent setup.

12. \*\*Alerting Components\*\* (Optional): If you want to include alarms or notifications, you may need LEDs, buzzers, or relays.

13. \*\*Wi-Fi/Cellular Module\*\* (Optional): For remote monitoring, consider adding a Wi-Fi or cellular module to transmit data to a remote server or your smartphone.

14. \*\*Sensors for Environmental Data\*\* (Optional): To enhance your system, you can incorporate other sensors like temperature, humidity, or GPS for location-based information.

15. \*\*Tools\*\*: Standard tools like soldering equipment, a screwdriver, and a multimeter for testing and assembly.

The specific components you choose may vary based on your project's requirements and budget. Make sure to read the datasheets and documentation for your selected components, as they will provide information on wiring and programming.

**CONCLUSION**

The above are the components of the noise pollution monitoring.

The app will record the sound from microphone and show the real time noise level data. This project will very useful to avoid noise pollution and to protect us from over noise which causes damage to our hearing. It wil also work for future generations and they will also be protected from unwanted noise.