NOISE POLLUTION MONITORING

OVERVIEW:

Noise monitoring refers to the systematic process of measuring, recording, and assessing sound levels in various environments to understand the extent of noise pollution and its potential impact on human health and the surrounding ecosystem.

Project Objectives:

I)Measurement of Ambient Noise Levels

2) Real-time Monitoring

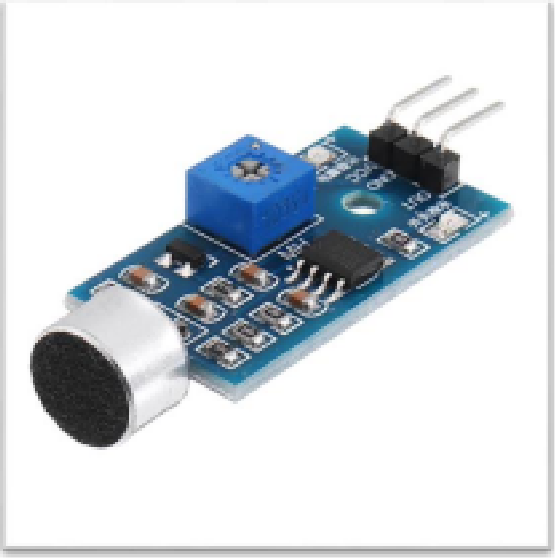
3) Data Collection and Analysis:

IOT Device Setup:

The IoT device setup in wokwi involves the following components and connections required for noise pollution monitoring.

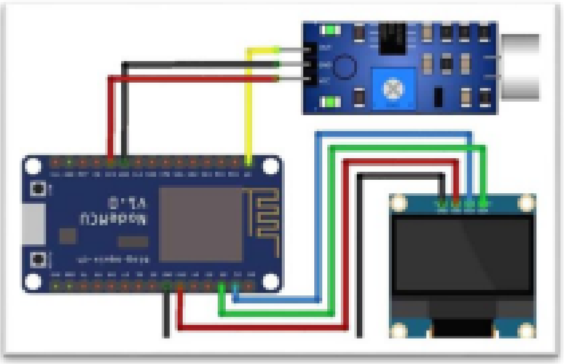
1)Microphone/Sound Sensor:

In Wokwi, you can simulate a microphone or sound sensor using analog input components. For example, you can use a potentiometer to simulate the analog output of a sound sensor. Connect the potentiometer as a voltage divider circuit to mimic changes in sound level.



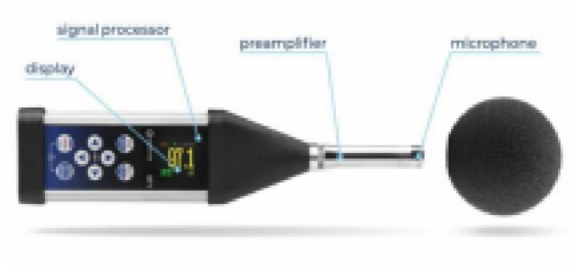
2)IoT Device:

You can select a microcontroller or development board (e.g., Arduino, ESP8266, ESP32) available in the Wokwi component library as your IoT device.



3)Preamplifier (Optional):

In a virtual environment like Wokwi, you can skip the preamplifier since you can directly simulate the analog input from the sound sensor.



4)Data Acquisition System (ADC):

Most microcontrollers have built-in analog-to-digital converters (ADCs). You can use these internal ADCs to convert the analog signal from the simulated sound sensor into digital data.

5)Power Supply:

Wokwi provides a virtual power supply for your IoT device, and you can configure its voltage and current as needed.

Platform Development:

Creating a web-based platform to display real-time sound quality data is a valuable project that can benefit various stakeholders. This platform can be used for noise pollution monitoring and management. Here are the steps and considerations to develop such a platform:

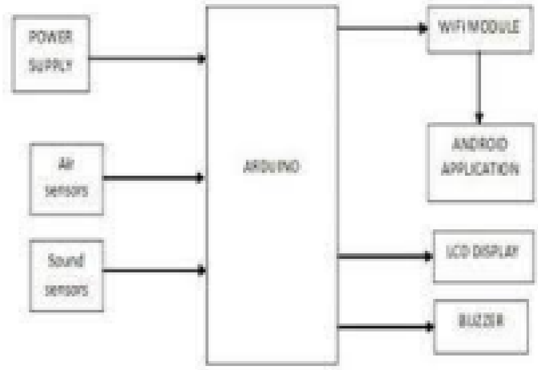
1) Web development framework and programming languages (e.g., HTML, CSS, JavaScript, Python, PHP) to create the platform.

2) Develop algorithms for real-time data processing, including sound quality analysis, noise source identification, and trend recognition.

3) Create real-time data visualization components, such as charts, graphs, and maps, to display sound quality data.

4) Create a database for storing real-time and historical sound quality data.

Block diagram:

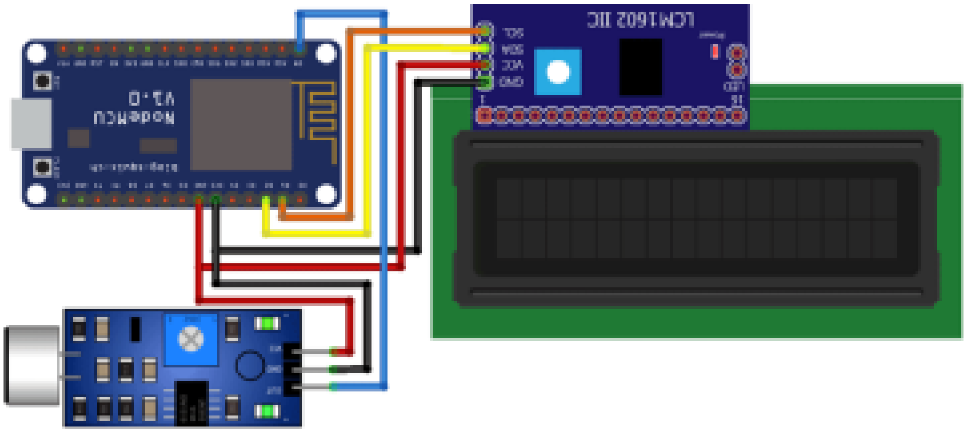


The Program Execution Code :

```
<!DOCTYPE html>

<html>

<head>

 <meta charset="UTF-8">

 <title>Noise Pollution Awareness</title>

<style>

 body {

 font-family: Arial, sans-serif;

margin: 20px;

 }


 h1 {

 color: #0066cc;

 }
 p {

 margin-bottom: 20px;
 }

 .image-container {
 text-align: center;
 margin: 20px 0;
 }

 img {
 max-width: 100%;
 }

 .button {
 display: inline-block;
 padding: 10px 20px;
 background-color: #0066cc;
color: #fff;
 text-decoration: none;
 border: none;
 border-radius: 5px;
 }
 </style>
</head>
<body>
 <h1>Understanding Noise Pollution</h1>
 <p>Noise pollution is the presence of loud, disruptive, or harmful sounds in the
environment that can have adverse effects on human health, well-being, and the ecosystem.
It is essential to raise awareness about the causes and consequences of noise pollution.</p>
```

```
<p>Key points about noise pollution:</p>

<ul>

<li>Noise sources include traffic, industrial operations, construction, and recreational
activities.</li>

<li>Prolonged exposure to noise pollution can lead to stress, hearing
```

Connected components:



Simulation:

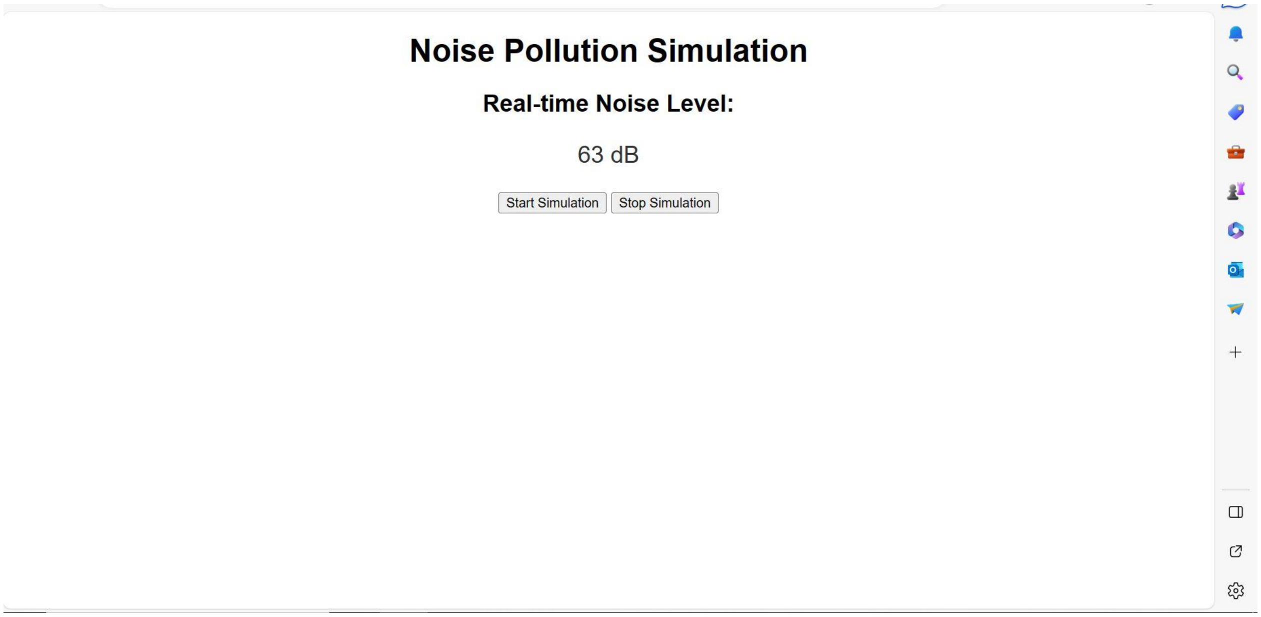
To create a web-based simulation for noise pollution monitoring, you'll need a combination of HTML, CSS, and JavaScript to build an interactive interface that simulates the process. Below is a basic example of an HTML page with JavaScript for simulating noise pollution data:

```
<!DOCTYPE html>

<html>

<head>

<title>Noise Pollution Simulation</title>

<style>

/* Add your CSS styles for the simulation interface here */

body {

font-family: Arial, sans-serif;
text-align: center;

}

#simulation-container {
```

```
width: 80%;
margin: 0 auto;
}
#noise-level {
font-size: 24px;
color: #333;
}
</style>
</head>
<body>
<h1>Noise Pollution Simulation</h1>

<div id="simulation-container">
<h2>Real-time Noise Level:</h2>
<p id="noise-level">0 dB</p>

<button onclick="startSimulation()">Start Simulation</button>
<button onclick="stopSimulation()">Stop Simulation</button>
</div>

<script>
let simulationInterval;

function startSimulation() {
simulationInterval = setInterval(updateNoiseLevel, 1000); // Simulate noise update every second  }
function stopSimulation() {
clearInterval(simulationInterval);
```

```
document.getElementById("noise-level").textContent = "Simulation stopped";  }


function updateNoiseLevel() {
// Simulate noise data (replace with your logic)
const noiseLevel = Math.floor(Math.random() * 100); // Random value for demonstration
document.getElementById("noise-level").textContent = `${noiseLevel} dB`;
}
</script>
</body>
</html>
```

Output:



Noise Pollution Simulation

Real-time Noise Level:

63 dB

In this simulation, we have a simple web page with a "Start Simulation" button, a "Stop Simulation" button, and an element that displays the simulated noise level. When you click the "Start Simulation" button, it initiates a JavaScript function to simulate real-time noise level updates (in this case, generating random values for demonstration purposes).