**Noise pollution mointroing**

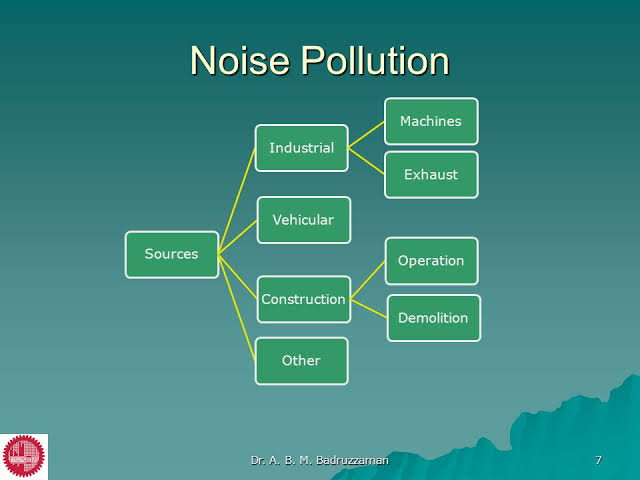
**Objectives:**

To regulate and control noise producing and generating sources. Maintaining the ambient air quality standards in respect of noise

.

Noise pollution,also known as environmental noise or sound pollution, is the propagation of noise with harmful impact on the activity of human or animal life.

**Diagram:**



**Sensor:**

The IoT sensor setup includes a noise sensor connected to a microcontroller

(e.g. Arduino or Raspberry Pi).

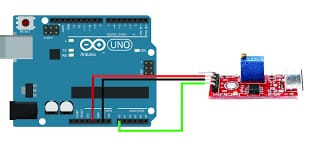
The sensor measures the sound decibel in a public place.

Normal sound level decibel is 60,noise level above 70dB is prolonged period of time may start to damage your hearing.

Loud noise above 120 dB can cause immediate harm to your ears.

The setup could also include additional sensors for temperature and humidity to gather environmental data.

Here's a simplified diagram of the IoT sensor setup:

****

**Arduino integration:**

1.Hardware: Arduino boards come in various models, each with its own specifications. They typically include digital and analog input/output pins, power connectors, USB interface, and more.

2.Software: Arduino uses a simplified version of the C/C++ programming language. You write code in the Arduino IDE (Integrated Development Environment), which is a user-friendly environment for programming and uploading code to the board.

3.Libraries: Arduino has a vast library of pre-written code and functions that make it easier to work with various sensors, displays, and modules.

4.Community: Arduino has a large and active community of users and developers who share projects, code, and provide support through forums and online resources .

5.Versatility: You can use Arduino for a wide range of applications, including home automation, robotics, wearable technology, and more.

**Code Implementation:**

The code for this project involves programming the microcontroller (Arduino or Raspberry Pi) to collect and transmit sensor data. It also includes the development of the mobile app and the code for the Ardunio to process and manage data.

Arduino/Raspberry Pi Code: Involves reading data from sensors, processing it, and transmitting it to the Arduino or a cloud service.

Mobile App Code: Develop a user-friendly app that communicates with the Arduino or cloud service to fetch and display real-time and historical data, set targets, and receive alerts.

**code:**

import serial

import requests

# Define the Arduino serial port (change this to the correct port)

arduino\_port = 'COM6'

# Define the API endpoint of the noise pollution information platform

api\_url = "https://your-api-endpoint.com/data"

try:

arduino = serial.Serial(arduino\_port, 9600)

except Exception as e:

print(f"Error opening Arduino serial port: {str(e)}")

exit()

while True:

try:

data = arduino.readline().decode('utf-8').strip()

print(f"Received from Arduino: {data}")

data = {

"sensor\_id": "sensor\_1", # Replace with your sensor ID

"noise\_level": data,

"timestamp": int(time.time())

}

response = requests.post(api\_url, json=data)

if response.status\_code == 200:

print("Data sent successfully")

else:

print("Failed to send data")

except Exception as e:

print(f"An error occurred: {str(e)}")

**1. Platform Development (Web)**

**Front-End: HTML, CSS, and JavaScript**

**Design the User Interface (UI):** Create wireframes and mockups for the web platform. Consider a clean and intuitive design that provides real-time noise level data.

**HTML and CSS:** Build the structure and style of the web platform using HTML and CSS. Ensure its responsive for various screen sizes.

**JavaScript:** Implement the functionality for real-time noise level updates using JavaScript.

**API Integration:** Connect the platform to a source of real-time noise data (e.g., noise level sensors, government databases, or crowdsourced data).

**Real-Time Data Display**

Use technologies like WebSocket or server-sent events (SSE) to provide real-time updates to the user interface.

Visualize noise level data using charts or graphs to make it easy to understand.

**2. Mobile App Development (iOS and Android)**

**Front-End:** Mobile App Development Frameworks

**iOS App:** Use Swift and Xcode for iOS app development.

**Android App:** Use Kotlin and Android Studio for Android app development. Implement the user interface for the mobile apps, ensuring a consistent design with the web platform.

**Real-Time Updates:** Implement features that allow users to receive real-time noise level updates. Use push notifications to alert users to significant changes in noise levels or other relevant information.

**User Profiles and Preferences :**Develop user profile sections in the mobile apps, allowing users to customize their notification preferences and view their historical data.

User Authentication Implement user authentication for the mobile apps, allowing users to log in using their platform credentials.

**Testing and Quality Assurance:** Test the web platform and mobile apps thoroughly to identify and resolve any bugs or issues. Consider usability testing to ensure a user-friendly experience.

**Deployment and Hosting:**Deploy the web platform on a web server or cloud hosting service. Publish the mobile apps on the App Store (iOS) and Google Play Store (Android).

**Real-time data display platform:-**

**Programming code:**

## <!DOCTYPE html>

## <html lang="en">

## <head>

## <meta charset="UTF-8">

## <meta name="viewport" content="width=device-width, initial-scale=1.0">

## <title>Noise Pollution Monitoring</title>

## <style>

## body {

## font-family: Arial, sans-serif;

## text-align: center;

## background-color: #f00505;

## margin: 0;

## }

## header {

## background-color: #333;

## color: #fff;

## padding: 20px;

## }

## h1 {

## font-size: 24px;

## }

## #noise-container {

## background-color: #fff;

## border-radius: 10px;

## box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

## padding: 20px;

## margin: 20px;

## }

## #noise-level {

## font-size: 36px;

## color: #333;

## }

## </style>

## </head>

## <body>

## <header>

## <h1>Noise Pollution Monitoring</h1>

## </header>

## <div id="noise-container">

## <h2>Real-Time Noise Level</h2>

## <div id="noise-level">Loading...</div>

## </div>

## <script>

## function updateNoiseLevel() {

## // Simulate fetching real-time noise data (replace with actual data source)

## const noiseLevel = Math.floor(Math.random() \* 100); // Random value for demonstration

## // Display the noise level in the HTML element

## const noiseDisplay = document.getElementById("noise-level");

## noiseDisplay.innerHTML = `Noise Level: ${noiseLevel} dB`;

## }

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

## setInterval(updateNoiseLevel, 5000);

## // Initial data update when the page loads

## updateNoiseLevel();

## </script>

## </body>

## </html>

