NOISE POLLUTION MONITORING

Noise pollution monitoring is the process of measuring, analysing, and assessing levels of unwanted and disruptive noise in an environment to understand its impact on human health, well-being, and the surrounding ecosystem. This monitoring involves the use of specialized equipment such as noise level meters and sensors to collect data on noise levels over time.

**Problem Definition :**

The Smart Noise project aims to address the growing concern of noise pollution in urban environments by deploying IoT sensors on to measure noise levels in public areas. The project's primary goal is to provide real-time noise data to the public through a user-friendly platform or mobile app. By doing so, it seeks to raise awareness about noise pollution and empower individuals and communities to make informed decisions to mitigate its impact. Understanding the Project

**Project Objective:**

The objective of a Noise Pollution Monitoring Project is typically to assess and mitigate the impact of noise pollution on a specific area or community. This may include:

1. Measuring and analysing noise levels in different locations.

2. Identifying sources of excessive noise.

3. Assessing the health and environmental effects of noise pollution.

4. Developing strategies to reduce noise pollution.

5. Raising awareness and educating the public about noise pollution and its effects.

The specific goals and scope of such a project can vary depending on the location and the issues related to noise pollution that need to be addressed

**How does the project Works? :**

Noise pollution monitoring using a Raspberry Pi Pico works by using the Pico as the brains of the operation to collect noise data from a sound sensor (like a microphone) and transmit that data for analysis or storage.

1. Device Setup

2. Data Acquisition

3. Data Processing

4. Data Transmission

5. Data Analysis

6.Visualization and Alerting

**Working of the Project:**

The noise pollution monitoring system operates by utilizing sensors, typically microphones or sound sensors, to constantly capture sound data from the environment. This data is then processed to measure the noise level in decibels (dB). The collected noise information can be transmitted to a central system or database using technologies such as the Internet of Things (IoT) for analysis and storage. Through data analysis, the system identifies noise patterns, peak levels, and trends over time, offering insights into when and where noise pollution is most pronounced. This information is crucial for decision-making, implementing noise control measures, enforcing legal regulations, and improving public awareness of noise-related issues to create quieter and more harmonious living environments.

**IoT Device setup:**

Wokwi is a platform that allows users to set up and simulate IoT (Internet of Things) devices for development and testing purposes. It provides a virtual environment for creating and testing IoT projects. Some of the key features and objectives of the Wokwi IoT device setup platform:

1. Virtual Hardware Simulation: Wokwi enables users to simulate IoT hardware components and devices in a virtual environment, eliminating the need for physical hardware during the development and testing phase.
2. IoT Prototyping: Users can prototype and experiment with IoT projects, including the setup and interaction of sensors, actuators, microcontrollers, and communication protocols like Wi-Fi or Bluetooth.
3. Code Development: Developers can write, test, and debug code for IoT devices using a platform-supported programming language or framework.
4. Integration: The platform may offer integrations with popular IoT development boards and platforms, such as Arduino, Raspberry Pi, or ESP8266/ESP32, making it easier to work with specific IoT hardware.
5. Select Components: In the Wokwi platform, select the components you need for your IoT project. This may include a Raspberry Pi Pico, a microphone sensor, and any other components you plan to use.
6. Wire Components: Use the virtual breadboard to wire up the components as you would in a physical circuit. Connect the microphone sensor to the Raspberry Pi Pico.
7. Programming Raspberry Pi Pico: Write the code for your Raspberry Pi Pico using MicroPython or CircuitPython. You can use the built-in code editor or import code from your development environment.
8. Test and Simulate: Run the simulation to test your code and see how your IoT setup behaves in a virtual environment. You can interact with the simulated components to check for noise level readings.
9. Data Visualization (Optional): Set up data visualization within the Wokwi platform to view the noise pollution data generated during the simulation.
10. Iterate and Debug: Make adjustments to your code and circuit as needed, and continue testing and simulating until you are satisfied with the results.
11. Save and Share: Save your project on Wokwi, and you can also share it with others for collaboration or educational purposes.

**Components required:**

Microcontroller-Raspberry Pi Pico

Raspberry Pi is a series of small single-board computers (SBCs) developed in the United Kingdom by Raspberry Pi Ltd in association with Broadcom. The Raspberry Pi project originally leaned toward the promotion of teaching basic computer science in schools .The original model became more popular than anticipated,selling outside its target market for uses such as robotics. It is widely used in many areas, such as for weather monitoring,because of its low cost, modularity, and open design. It is typically used by computer and electronic hobbyists, due to its adoption of the HDMI and USB standards.

Sound Sensor-Microphone

A microphone is a device that translates sound vibrations in the air into electronic signals and scribes them to a recording medium or over a loudspeaker.

These components are enough for the simulating in Wokwi Simulation platform.

For physical hardware basis we need Microcontroller,LM393 Sound Sensor Module,Display and required components for Connectivity.

**Main Program code :**

/\*

\* Microphone Noise Level Measurement in Decibels (dB SPL)

\*/

const int microphonePin = A0; // Analog pin connected to the microphone output

const float referenceVoltage = 3.3; // Reference voltage of the analog-to-digital converter (in volts)

const float sensitivity = 40; // Microphone sensitivity in mV/Pa (adjust this value based on your microphone's datasheet)

const float referencePressure = 20E-6; // Reference sound pressure level in Pa (corresponding to 0 dB SPL)

void setup() {

Serial.begin(115200);

pinMode(16, OUTPUT); // GPIO 16 on Raspberry Pi Pico

}

void loop() {

int sensorValue = analogRead(microphonePin); // Read analog input from the microphone

float voltage = (sensorValue / 1023.0) \* referenceVoltage; // Convert analog reading to voltage

// Calculate sound pressure level in dB SPL

float pressure = voltage / sensitivity; // Calculate sound pressure in Pa

float dBspl = 20 \* log10(pressure / referencePressure); // Calculate dB SPL

Serial.println("Sound Pressure Level: " + String(dBspl) + " dB SPL"); // Print dB SPL to the serial monitor

digitalWrite(16, sensorValue > 50); // Blink LED if noise level is above a threshold (adjust as needed)

delay(100); // Delay for a short duration before reading the microphone again

}

**Diagram.Json Code :**

{

  "version": 1,

  "author": "premasagar K",

  "editor": "wokwi",

  "parts": [

    { "type": "wokwi-pi-pico", "id": "pico", "top": -41.55, "left": -34.8, "attrs": {} },

    { "type": "wokwi-microphone", "id": "mic", "top": -132.18, "left": -14.61, "attrs": {} }

  ],

  "connections": [

    [ "pico:A0", "mic:1", "blue", [ "v0" ] ],

    [ "pico:GND.1", "mic:2", "black", [ "v29", "h149" ] ],

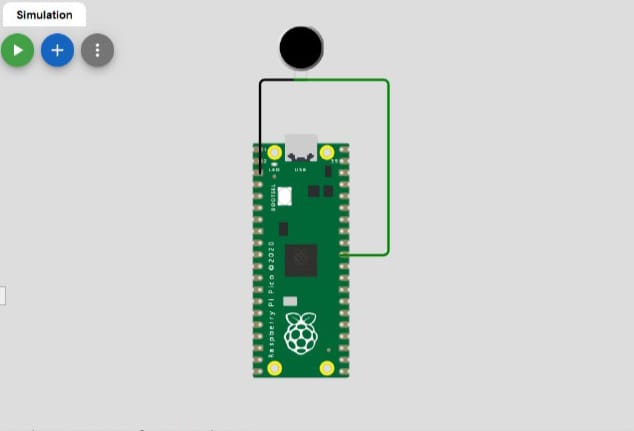
    [ "mic:1", "pico:GP26", "green", [ "h76.8", "v144" ] ]

  ],

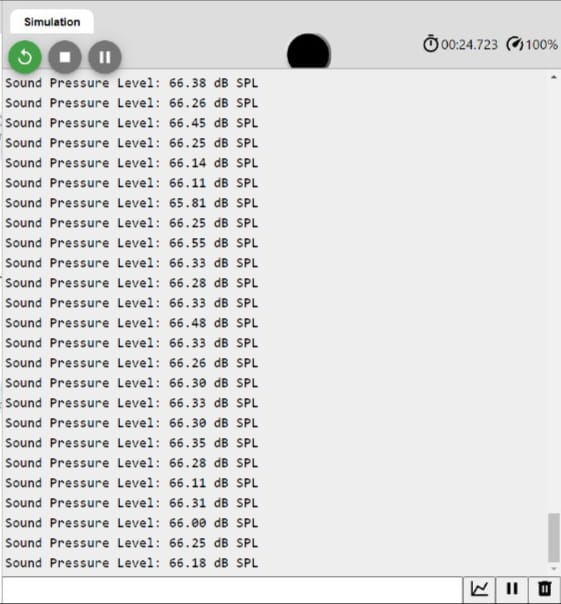
  "serialMonitor": { "display": "plotter" },

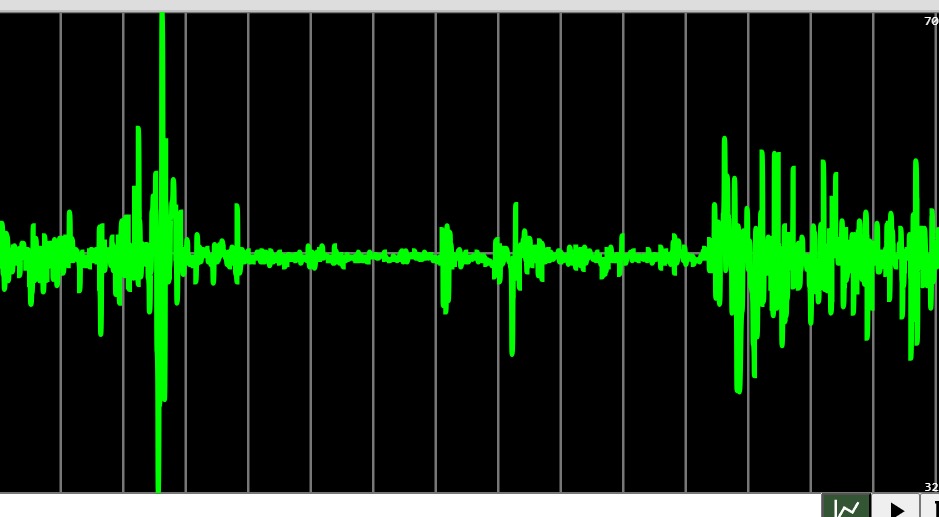
  "dependencies": {}

}

**IOT Device:** 

**Output:**





**Wokwi Simulation In VS code:**

At this Stage of Work,We need to store the data from IOT simulation in Wokwi to local CSV file called”data.csv”.

At first we need to download extension in VS code for Wokwi Simulator.After installing the extension We open the Wokwi simulation File .Then The following code responds for storing data from Wokwi simulator File.

import csv

# Sample data (replace this with your IoT simulation data)

iot\_data = [

{"timestamp": "2023-11-01 10:00:00", "sensor\_value": 75.2},

{"timestamp": "2023-11-01 10:05:00", "sensor\_value": 78.5},

# Add more data points as needed

]

# Specify the CSV file path

csv\_file\_path = 'data.csv'

# Write data to the CSV file

with open(csv\_file\_path, mode='w', newline='') as file:

fieldnames = ['timestamp', 'sensor\_value'] # Define CSV column headers

writer = csv.DictWriter(file, fieldnames=fieldnames)

# Write CSV header

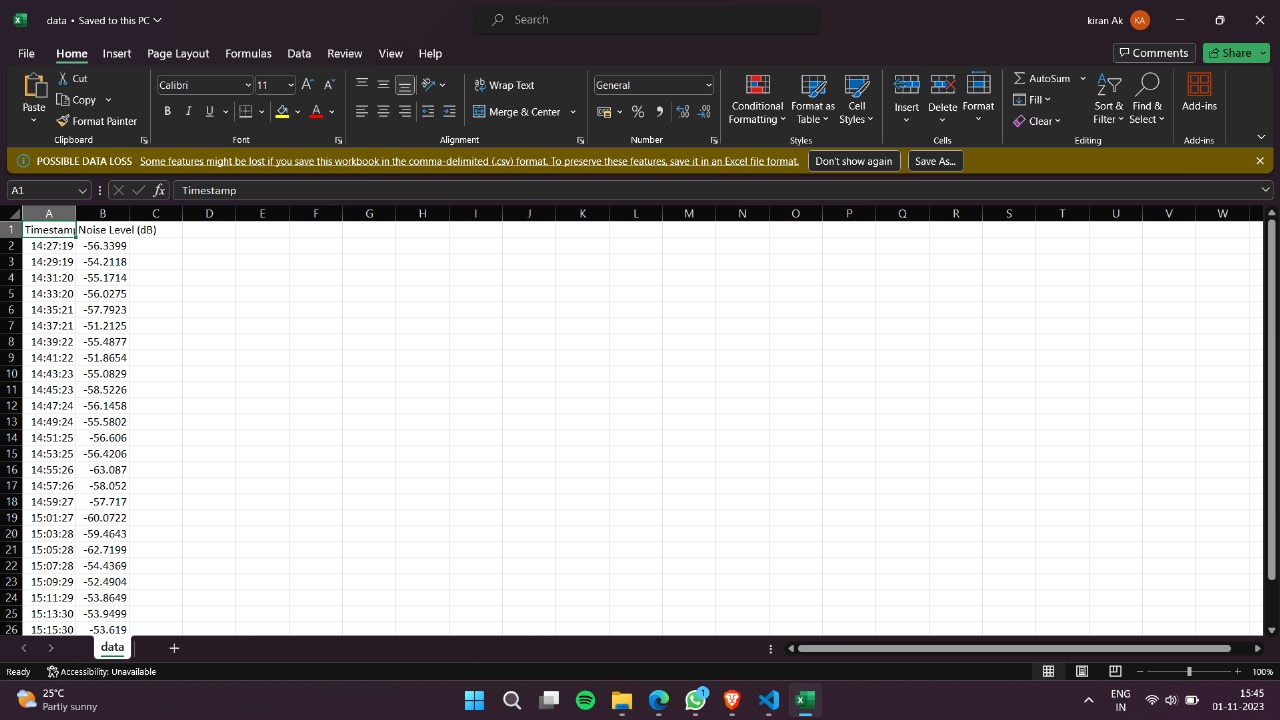
writer.writeheader()

# Write IoT data to CSV file

for data\_point in iot\_data:

writer.writerow(data\_point)

print(f"Data has been successfully written to {csv\_file\_path}")



As like this the above code is used to store data from Wokwi simulation is used to store data .

**From CSV file to Webpage:**

For easy accessing, Webpage is created for displaying the Noise Pollution Monitoring. Here the data are displayed from the CSV file which is getting data from IOT simulation

Using Flask with Python program to make connection between the CSV file (data.csv) and Webpage.

**CODE:**

from flask import Flask, render\_template

import pandas as pd

app = Flask(\_name\_)

@app.route('/')

def display\_csv():

# Read the CSV file into a Pandas DataFrame

df = pd.read\_csv('data.csv')

# Convert the DataFrame to an HTML table

table\_html = df.to\_html(classes='table table-striped table-bordered table-hover')

# Create an HTML template to display the table

html\_template = f'''

<!DOCTYPE html>

<html>

<head>

<title>CSV to HTML</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">

</head>

<body>

<div class="container">

{table\_html}

</div>

</body>

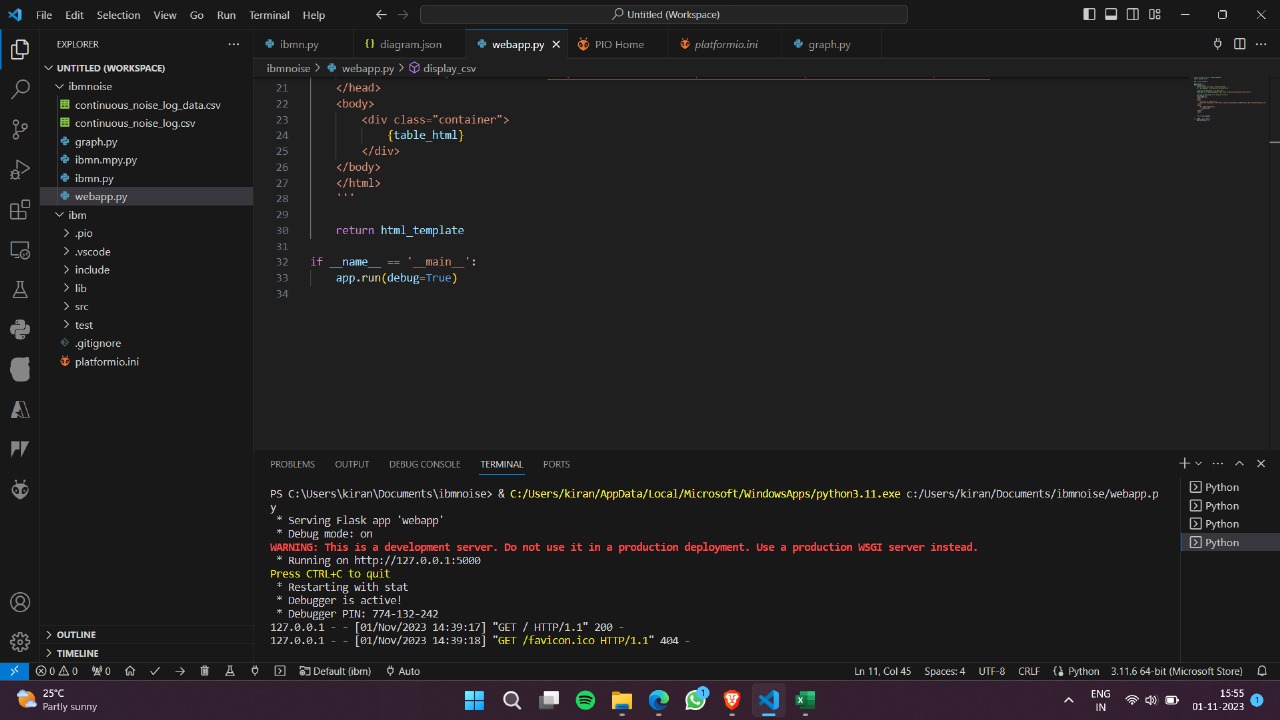
</html>

'''

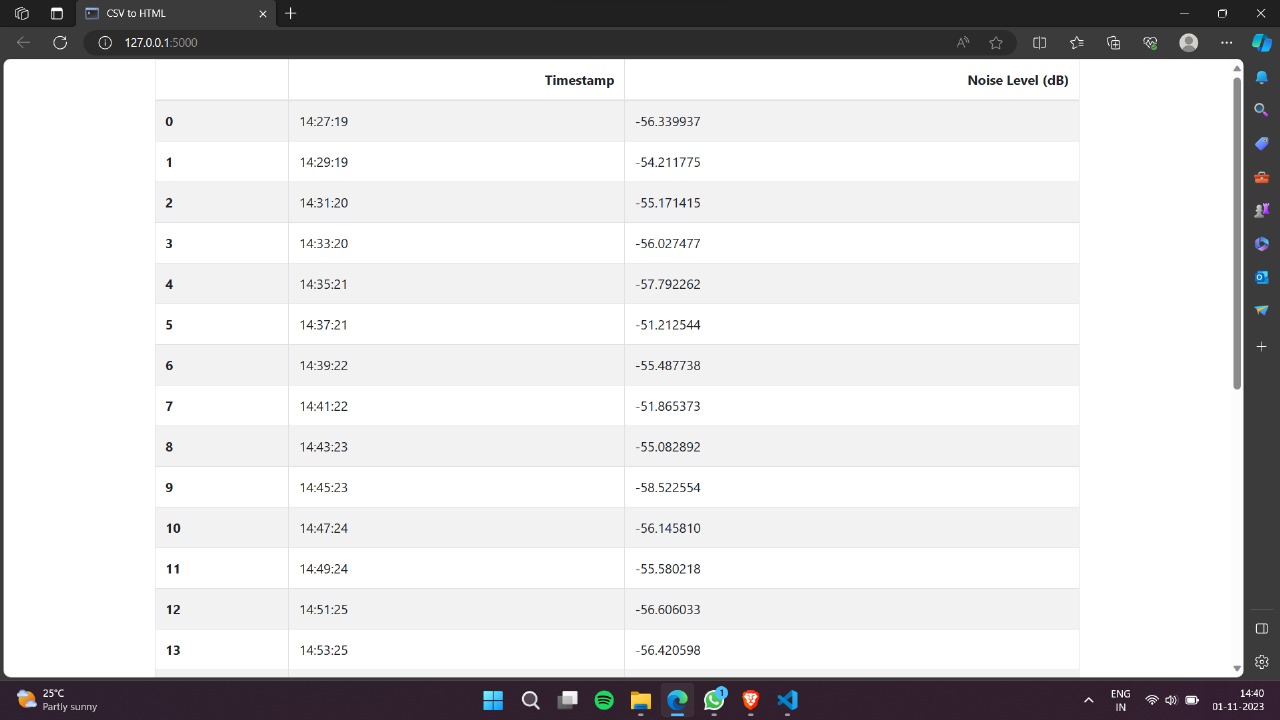
return html\_template

if \_name\_ == '\_main\_':

app.run(debug=True)



After the above code we got the http request to run the webpage and display the data in the webpage. ”**http;//127.0.0.1;5000**” Using this to view the data in webpage.



**Conclusion :**

Think of noise pollution monitoring as a diligent guardian watching over our well-being and the environment. It takes on multiple roles, such as evaluating how much noise we're exposed to, ensuring that industries and transportation systems follow noise regulations, and pinpointing the sources of noise disturbance. This guardian also educates the public about the impact of noise on our health, contributes to research and policymaking, and acts preventively to maintain a peaceful living environment. In essence, noise pollution monitoring plays the role of a caring protector, striving to create a quieter, healthier world for generations to come.