**Noise pollution monitoring**

**Project Objectives:**

**The primary objectives of this Noise Pollution IoT project are to:**

* Measure Noise Levels: Develop a system to accurately measure noise levels in real-time.
* Data Collection: Collect noise data from multiple sensor nodes deployed in different locations.
* Data Analysis: Process and analyze the collected noise data to detect noise pollution events and patterns.
* Data Visualization: Create a user-friendly web-based platform to display noise data with charts and graphs.
* Alerting: Implement a notification system to alert relevant authorities and the public about noise pollution incidents.

**IoT Device Setup:**

**Hardware Components:**

* Noise Sensors: Deploy noise sensors (e.g., microphones) in various locations to capture noise levels.
* Microcontrollers: Connect microcontrollers (e.g., Arduino, Raspberry Pi) to the sensors to read and process data.
* Communication Modules: Equip each device with communication modules (e.g., Wi-Fi, GSM) to send data to the cloud.

**Software Components:**

* Sensor Code: Write code for the microcontrollers to read sensor data and transmit it to the cloud.
* Data Processing: Develop algorithms to filter, process, and store the data.

**Platform Development:**

**Cloud Infrastructure:**

* Cloud Server: Set up a cloud server (e.g., AWS, Azure) to receive and store data from the IoT devices.
* Database: Create a database (e.g., MySQL, NoSQL) to store the collected noise data.

**Data Processing:**

* Implement data processing pipelines to clean and normalize the data.
* Apply data analysis techniques to identify noise pollution events, calculate average noise levels, and generate reports.

**Web-Based Dashboard:**

* Develop a web-based dashboard using HTML, CSS, and JavaScript to visualize the data.
* Create charts and graphs to display noise levels over time, location-based heatmaps, and historical trends.
* Include user authentication for authorized access.

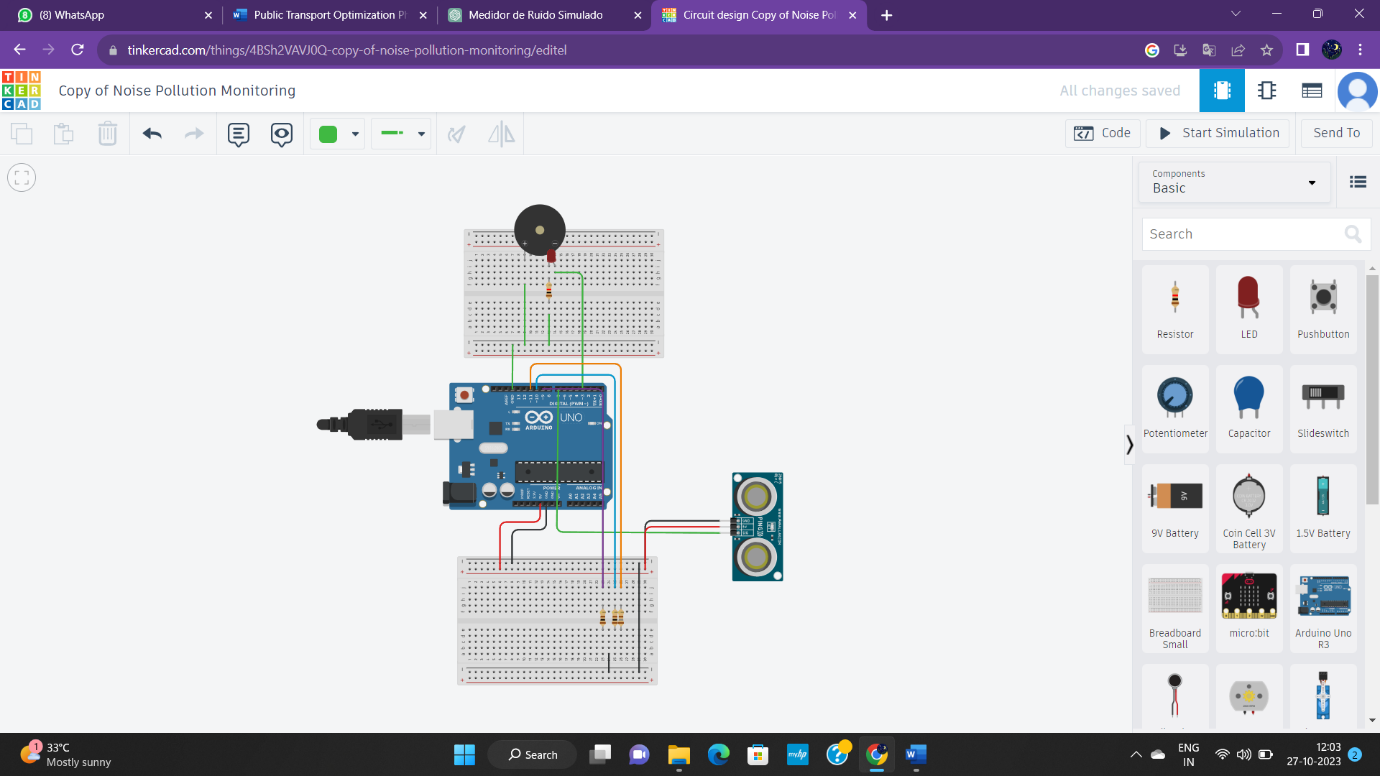
**Alerting System:**

* Implement an alerting system that can trigger notifications (emails, SMS, or push notifications) when noise pollution thresholds are exceeded.

**Code Implementation**:

* IoT Device Code: Write code for the IoT devices to collect noise data from sensors and send it to the cloud. Use libraries and frameworks compatible with the microcontroller platform.
* Cloud Server Code: Develop server-side code to receive and process incoming data. Store the data in a database for analysis.
* Data Processing Code: Create scripts or programs to clean, filter, and analyze the noise data. Implement algorithms for noise pollution detection and event identification.
* Web Dashboard Code: Build the web-based dashboard using HTML, CSS, and JavaScript. Utilize libraries and frameworks like React, Angular, or Vue.js for front-end development.
* Alerting Code: Set up the alerting system, configure thresholds, and implement code to trigger notifications when necessary.
* Security: Ensure data security and user authentication in both the IoT devices and the web platform.

**screenshot of noise pollution monitoring:**

****

**Code For Running above Circuit:**

import requests

import random

import time

# Define ThingSpeak channel settings

THINGSPEAK\_API\_KEY = " FN7CI48PVD9BXZPE"

THINGSPEAK\_CHANNEL\_ID = " https://thingspeak.com/channels/2303451/api\_keys"

# Simulated noise sensor

class NoiseSensor:

def \_init\_(self, sensor\_id):

self.sensor\_id = sensor\_id

def read\_noise\_level(self):

# Simulate noise level (replace this with actual sensor reading)

noise\_level = random.uniform(50, 100)

return noise\_level

# Send noise sensor data to ThingSpeak

def send\_to\_thingspeak(api\_key, channel\_id, field, data):

url = f"https://api.thingspeak.com/update?api\_key={api\_key}&field{field}={data}"

response = requests.get(url)

if response.status\_code == 200:

print(f"Sent to ThingSpeak: {data}")

else:

print(f"Failed to send data to ThingSpeak: {response.status\_code}")

if \_name\_ == "\_main\_":

try:

sensor = NoiseSensor(sensor\_id="sensor\_1")

field\_number = 1 # Change this to the correct field number in your ThingSpeak channel

while True:

noise\_level = sensor.read\_noise\_level()

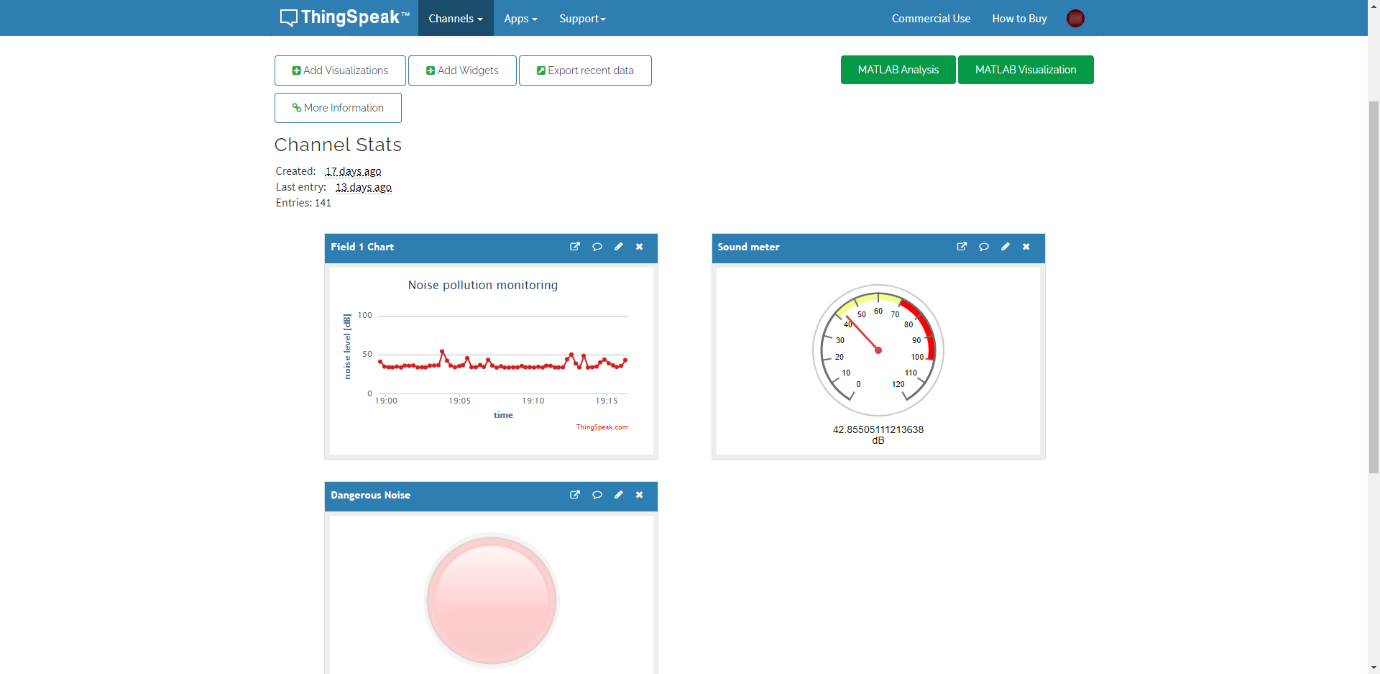
send\_to\_thingspeak(THINGSPEAK\_API\_KEY, THINGSPEAK\_CHANNEL\_ID, field\_number, noise\_level)

time.sleep(15) # Adjust the time interval as needed

except KeyboardInterrupt:

print("Terminating the sensor.")

**Thinkspeak Channel Stat of Noise Pollution:**

****

**Platform UI Code For Noise Pollution Monitoring:**

**HTML:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<link rel="stylesheet" href="styles.css">

<title>Noise Pollution Meter</title>

</head>

<body>

<div class="noise-meter">

<div class="needle"></div>

<input type="range" id="noise-input" min="0" max="100" value="50">

</div>

<p>Noise Level: <span id="noise-level">50</span> dB</p>

<script src="script.js"></script>

</body>

</html>

**CSS:**

body {

font-family: Arial, sans-serif;

text-align: center;

background-color: #f0f0f0;

}

.noise-meter {

width: 200px;

height: 400px;

margin: 50px auto;

background-color: #f5f5f5;

border: 2px solid #333;

position: relative;

}

.needle {

width: 2px;

height: 200px;

background-color: red;

position: absolute;

top: 50%;

left: 50%;

transform-origin: 50% 100%;

transform: translateX(-50%);

transition: transform 0.5s;

}

input[type="range"] {

width: 100%;

}

p {

font-size: 18px;

**JAVASCRIPT:**

const noiseInput = document.getElementById("noise-input");

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

const needle = document.querySelector(".needle");

noiseInput.addEventListener("input", () => {

const noiseLevel = noiseInput.value;

noiseLevelDisplay.textContent = noiseLevel;

rotateNeedle(noiseLevel);

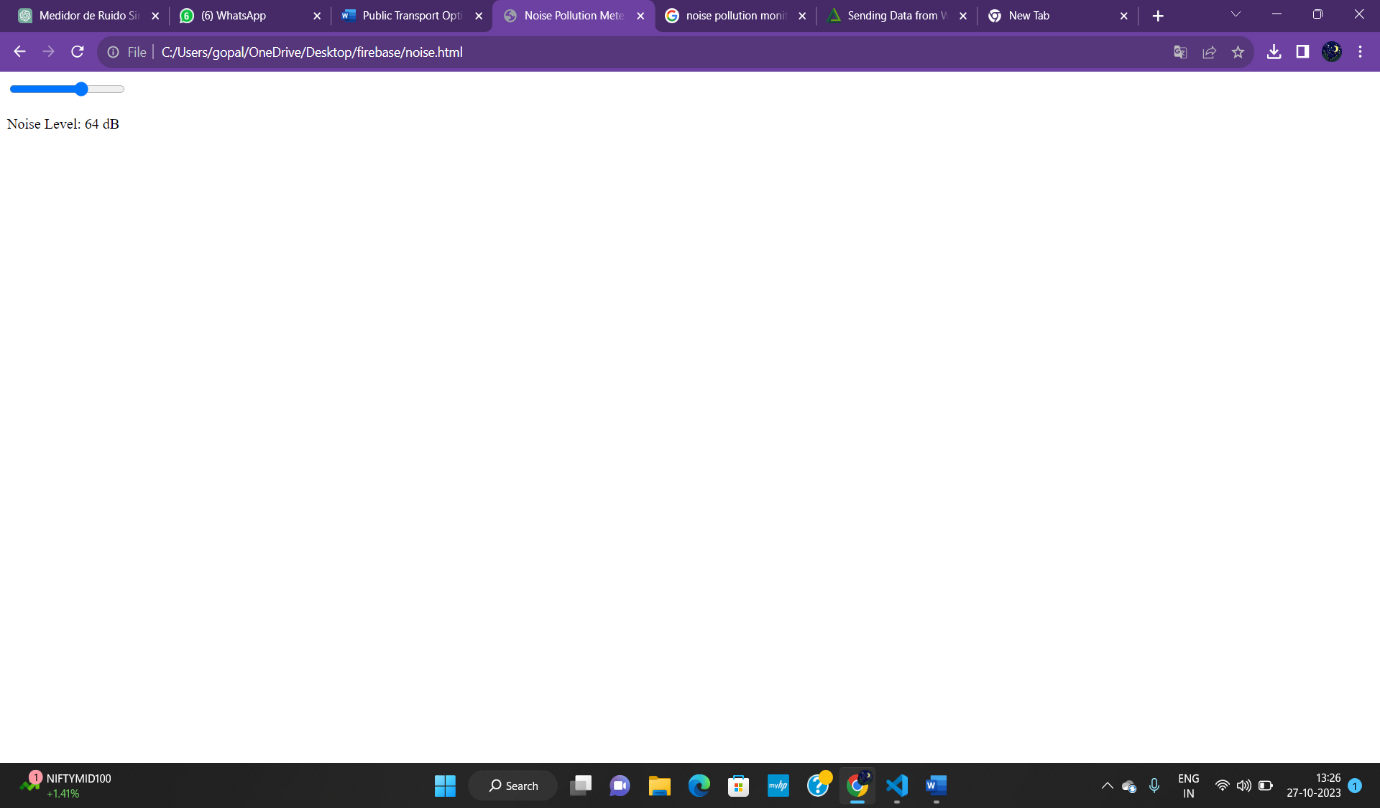
});

function rotateNeedle(degrees) {

needle.style.transform = `rotate(${(degrees - 50) \* 2}deg)`;

}

**Output For Above Program:**

****