**Team members**

Aravindan p(210621106006)

Avinash k(210621106008)

Bathmanaban v(210621106009

Ajai m(210621106003)

**Title:** noise pollution monitoring

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)

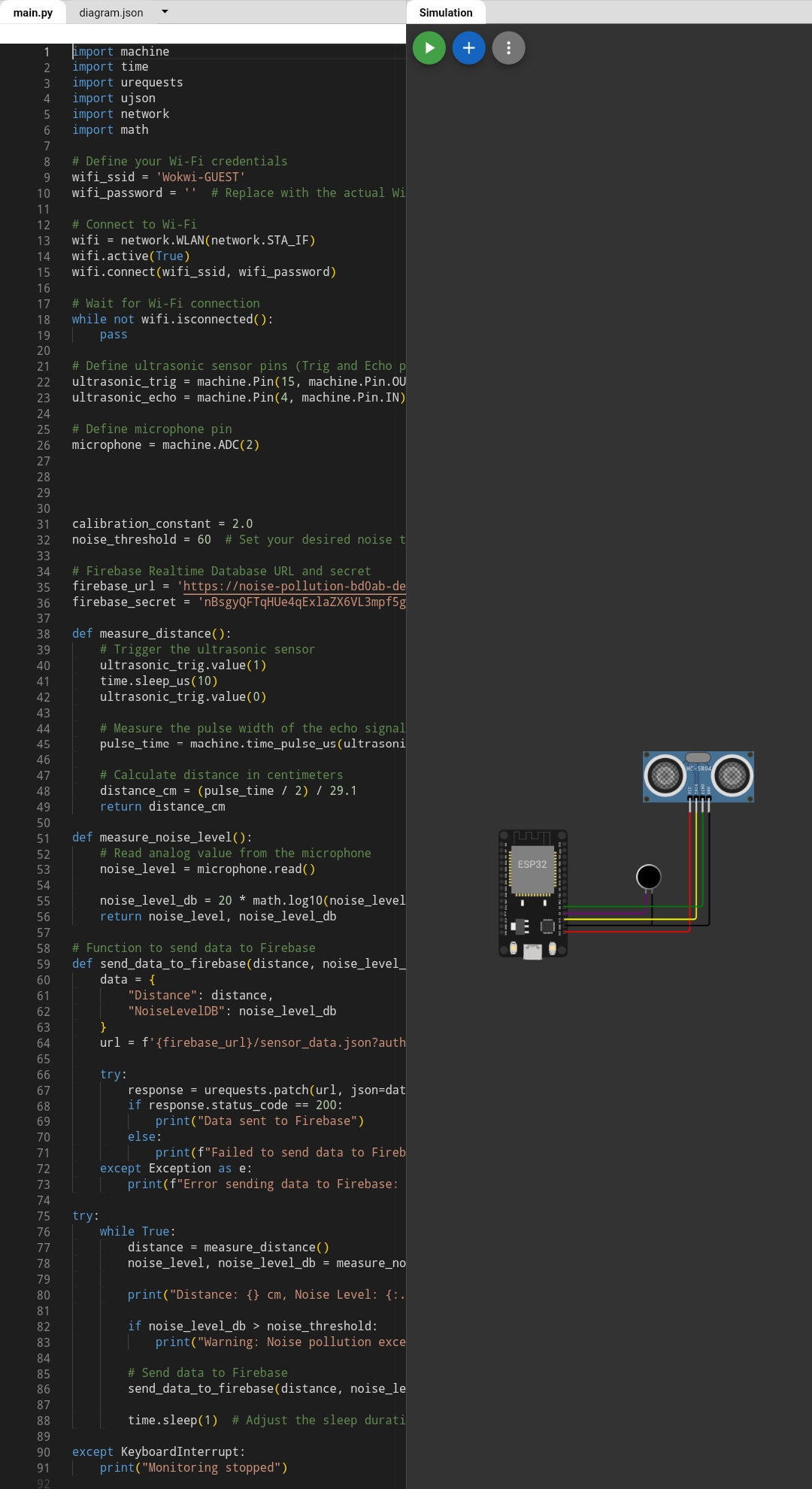
Time.sleep(1) # Adjust the sleep duration as needed

Except KeyboardInterrupt:

Print(“Monitoring stopped”)

**Simulation link:**

**https://wokwi.com/projects/378892802870919169**



**CONCLUSION**

In the pursuit of addressing noise pollution, this project has provided valuable insights into the noise environment of the target area. Through meticulous data collection, analysis, and engagement with the community. It also provides useful data of noise pollution by using these data, we can reduce noise pollution effectively.