**NOISE POLLUTION MONITORING**

**Requirements for the Code:**

The code is a MicroPython script for monitoring noise pollution and sending the collected data to a Firebase Realtime Database. Here are the requirements and a brief overview of the code:

Requirements:

1. Microcontroller: The code is designed to run on a microcontroller with Wi-Fi capabilities, such as the ESP8266 or ESP32.

2. Wi-Fi Network: You need to have access to a Wi-Fi network and know the SSID and password to connect to it.

3. Ultrasonic Sensor: The code uses an ultrasonic sensor to measure distances. You need to connect the sensor to specific GPIO pins on the microcontroller.

4. Microphone: A microphone sensor is used to measure noise levels. Connect the microphone to an analog input pin on the microcontroller.

5. Firebase Account: You need to have a Firebase account and create a Realtime Database. Replace the `firebase\_url` with the URL of your Firebase Realtime Database.

Brief Overview of the Code:

1. Wi-Fi Connection: The code sets up Wi-Fi credentials and establishes a connection to the specified Wi-Fi network. It waits until the connection is established before proceeding.

2. Sensor Configuration: The code configures the pins for the ultrasonic sensor (Trig and Echo pins) and the microphone (analog input pin).

3. Calibration Constants: A calibration constant is defined to convert analog microphone readings into dB values, and a noise threshold is set to specify the limit above which noise pollution is considered excessive.

4. Firebase Configuration: The Firebase Realtime Database URL is specified. You should replace this URL with your Firebase Database's URL.

5. Measurement Functions:

- `measure\_distance()`: Measures the distance using the ultrasonic sensor and calculates it in centimeters.

- `measure\_noise\_level()`: Reads the analog value from the microphone, calculates the noise level in dB using the calibration constant, and returns both the raw noise level and the dB value.

6. Data Sending to Firebase:

- `send\_data\_to\_firebase()`: Constructs a dictionary with distance and noise level data and sends it to the Firebase Realtime Database using a PATCH request.

7. Alert Function: The code includes a simple `trigger\_alert()` function that prints an "Alert" message to the console when the noise level exceeds the specified threshold. You can replace this function with your specific alert mechanism.

8. Main Loop: The main loop repeatedly performs the following actions:

- Measures distance and noise level.

- Prints the measurements to the console.

- If the noise level exceeds the threshold, it triggers an alert.

- Sends the collected data to the Firebase Realtime Database.

- Sleeps for a specified duration (1 second by default) before repeating the measurements.

9. Exception Handling: The code includes a KeyboardInterrupt exception handling block to allow you to stop the monitoring by pressing Ctrl+C.

This code is a basic framework for monitoring noise pollution and sending the data to Firebase. You can customize it to suit your specific project's needs and integrate more advanced alerting mechanisms or data processing as required.

# Here's a code snippet outline for Noise pollution monitoring :

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-monitori-7a445-default-rtdb.firebaseio.com/noise\_pollution\_monitoring.json' # Replace with your Firebase URL

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'

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(e}")

# Function to trigger an alert

def trigger\_alert():

print("Alert: Noise pollution exceeds threshold!")

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:

trigger\_alert()

# 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")