NOISE POLUTION MONITORING

Name : Arunkumar.M

Reg.No : 953021106008

Dept : ECE-III

College code : 9530

College Name : St.Mother Theresa Engineering College

Naan Mudhalvan id : au953021106008

Main Code :

1. import machine
2. import time
3. import urequests
4. import ujson
5. import network
6. import math
7. # Define your Wi-Fi credentials
8. wifi\_ssid = 'Wokwi-GUEST'
9. wifi\_password = '' # Replace with the actual Wi-Fi password
10. # Connect to Wi-Fi
11. wifi = network.WLAN(network.STA\_IF)
12. wifi.active(True)
13. wifi.connect(wifi\_ssid, wifi\_password)
14. # Wait for Wi-Fi connection
15. while not wifi.isconnected():
16. pass
17. # Define ultrasonic sensor pins (Trig and Echo pins)
18. ultrasonic\_trig = machine.Pin(15, machine.Pin.OUT)
19. ultrasonic\_echo = machine.Pin(4, machine.Pin.IN)
20. # Define microphone pin
21. microphone = machine.ADC(2)
22. calibration\_constant = 2.0
23. noise\_threshold = 60 # Set your desired noise threshold in dB
24. # Firebase Realtime Database URL and secret
25. firebase\_url = 'https://noise-pollution-bd0ab-default-rtdb.asia-southeast1.firebasedatabase.app/' # Replace with your Firebase URL
26. firebase\_secret = 'nBsgyQFTqHUe4qExlaZX6VL3mpf5gn6BlpnMiuR0' # Replace with your Firebase secret
27. def measure\_distance():
28. # Trigger the ultrasonic sensor
29. ultrasonic\_trig.value(1)
30. time.sleep\_us(10)
31. ultrasonic\_trig.value(0)
32. # Measure the pulse width of the echo signal
33. pulse\_time = machine.time\_pulse\_us(ultrasonic\_echo, 1, 30000)
34. # Calculate distance in centimeters
35. distance\_cm = (pulse\_time / 2) / 29.1
36. return distance\_cm
37. def measure\_noise\_level():
38. # Read analog value from the microphone
39. noise\_level = microphone.read()
40. noise\_level\_db = 20 \* math.log10(noise\_level / calibration\_constant)
41. return noise\_level, noise\_level\_db
42. # Function to send data to Firebase
43. def send\_data\_to\_firebase(distance, noise\_level\_db):
44. data = {
45. "Distance": distance,
46. "NoiseLevelDB": noise\_level\_db
47. }
48. url = f'{firebase\_url}/sensor\_data.json?auth={firebase\_secret}'
49. try:
50. response = urequests.patch(url, json=data) # Use 'patch' instead of 'put'
51. if response.status\_code == 200:
52. print("Data sent to Firebase")
53. else:
54. print(f"Failed to send data to Firebase. Status code: {response.status\_code}")
55. except Exception as e:
56. print(f"Error sending data to Firebase: {str(e)}")
57. try:
58. while True:
59. distance = measure\_distance()
60. noise\_level, noise\_level\_db = measure\_noise\_level()
61. print("Distance: {} cm, Noise Level: {:.2f} dB".format(distance, noise\_level\_db))
62. if noise\_level\_db > noise\_threshold:
63. print("Warning: Noise pollution exceeds threshold!")
64. # Send data to Firebase
65. send\_data\_to\_firebase(distance, noise\_level\_db)
66. time.sleep(1) # Adjust the sleep duration as needed
67. except KeyboardInterrupt:
68. print("Monitoring stopped")

Digram :

{

"version": 1,

"author": "Gokul Raja",

"editor": "wokwi",

"parts": [

{

"type": "wokwi-esp32-devkit-v1",

"id": "esp",

"top": -72.1,

"left": 52.6,

"attrs": { "env": "micropython-20231005-v1.21.0" }

},

{ "type": "wokwi-microphone", "id": "mic", "top": -16.98, "left": 263.79, "attrs": {} },

{

"type": "wokwi-hc-sr04",

"id": "ultrasonic1",

"top": -190.5,

"left": 274.3,

"attrs": { "distance": "88" }

}

],

"connections": [

[ "esp:TX0", "$serialMonitor:RX", "", [] ],

[ "esp:RX0", "$serialMonitor:TX", "", [] ],

[ "mic:1", "esp:D2", "purple", [ "v0" ] ],

[ "mic:2", "esp:GND.1", "black", [ "v0" ] ],

[ "ultrasonic1:VCC", "esp:3V3", "red", [ "v0" ] ],

[ "ultrasonic1:TRIG", "esp:D15", "yellow", [ "v0" ] ],

[ "ultrasonic1:ECHO", "esp:D4", "green", [ "v0" ] ],

[ "ultrasonic1:GND", "esp:GND.1", "black", [ "v0" ] ]

],

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

"dependencies": {}

}