PROJECT TITLE : NOISE POLLUTION MONITORING

PHASE :4 : DEVELOPMENT PART -2

TEAM MEMBERS:

913021104035-TAMIZHSELVAN S

913021104034-SRIKRISHNARAJAN N

913021104029-SHIEKABUTHAHEER S

913021104017-NIRMALA M

Noise pollution monitoring is the process of measuring and analyzing noise levels in an area to assess its impact, identify sources, and develop strategies for noise reduction and compliance with regulations

WOKWI :

Wokwi is a versatile online platform that allows you to design ,simulate, and test electronic circuits in a virtual environment.

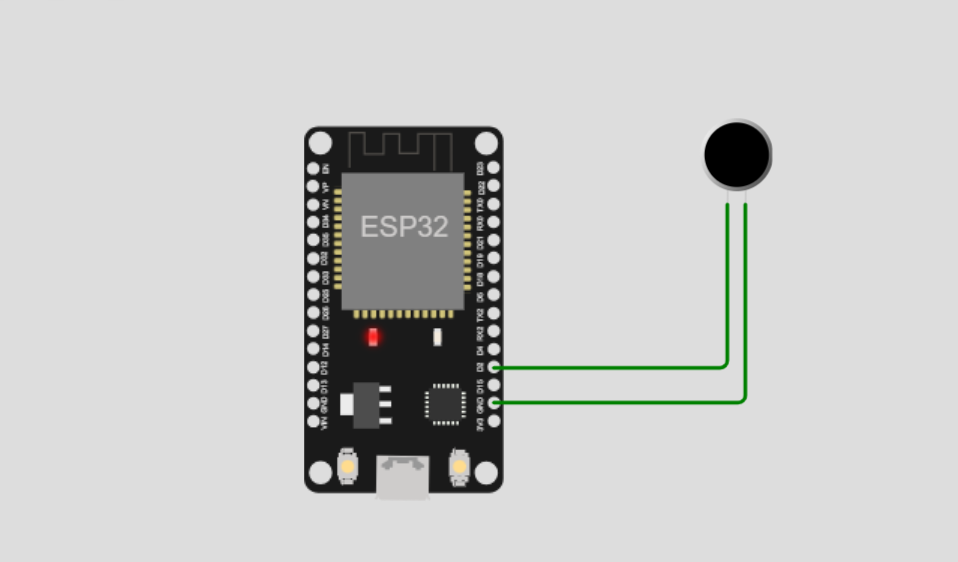
Website (<https://wokwi.com/arduino>)

Components Required :

1. ESP32(Micro controller)
2. Sound sensor

To monitoring noise pollution , you typically need a set of equipment and software to capture ,analyse and report sound levels .The above mentioned are some basis list of components required for noise pollution monitoring once you have these components ,you can set up a monitoring stations ,take measurements ,and analyse the data to draw conclusion about noise pollution in a specific area

Wiring connections



ESP32(Micro controller):

ESP32 is a versatile microcontroller with wireless connectivity, ideal for a wide range of applications.

Microphone :

A specialized microphone that captures sound accurately and is manually part of the SLM.

**Code**:

//main.py

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)}")

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")

//Diagram.json

{

  "version": 1,

  "author": "Tamizhselvan",

  "editor": "wokwi",

  "parts": [

    {

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

      "id": "esp",

      "top": -129.7,

      "left": 119.8,

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

    },

    { "type": "wokwi-microphone", "id": "mic", "top": -132.18, "left": 330.99, "attrs": {} }

  ],

  "connections": [

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

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

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

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

  ],

  "dependencies": {}

}

Result Analyse

This project aim to create a noise pollution monitoring using Wokwi , integrating a sound level or a compatible microcontroller with virtual components such as microphone ,ESP32 . The simulation successfully monitors sound level ,causing the led to blink . This project show case the ability of virtual components to emulate the functionality of a real word system within a simulated environment .

conclusion :

This wokwi – based noise pollution monitoring simulation project effectively illustrates the control and monitoring of sound level using a sound level meter and virtual components . The project demonstrate the practicality of using wok wi virtual environment for hardware simulation , allowing precise testing and visualization of system functionality without physical components. The LED and sound stimulator responded to sound level changes as expected , showcasing the potential for virtual hardware modelling .