Name: Lingeshwaran G

PHASE:3

NOISE POLLUTION MONITORING

DEVELOPMENT PART:1

- Start building the IOT enabled noise pollution monitoring system.

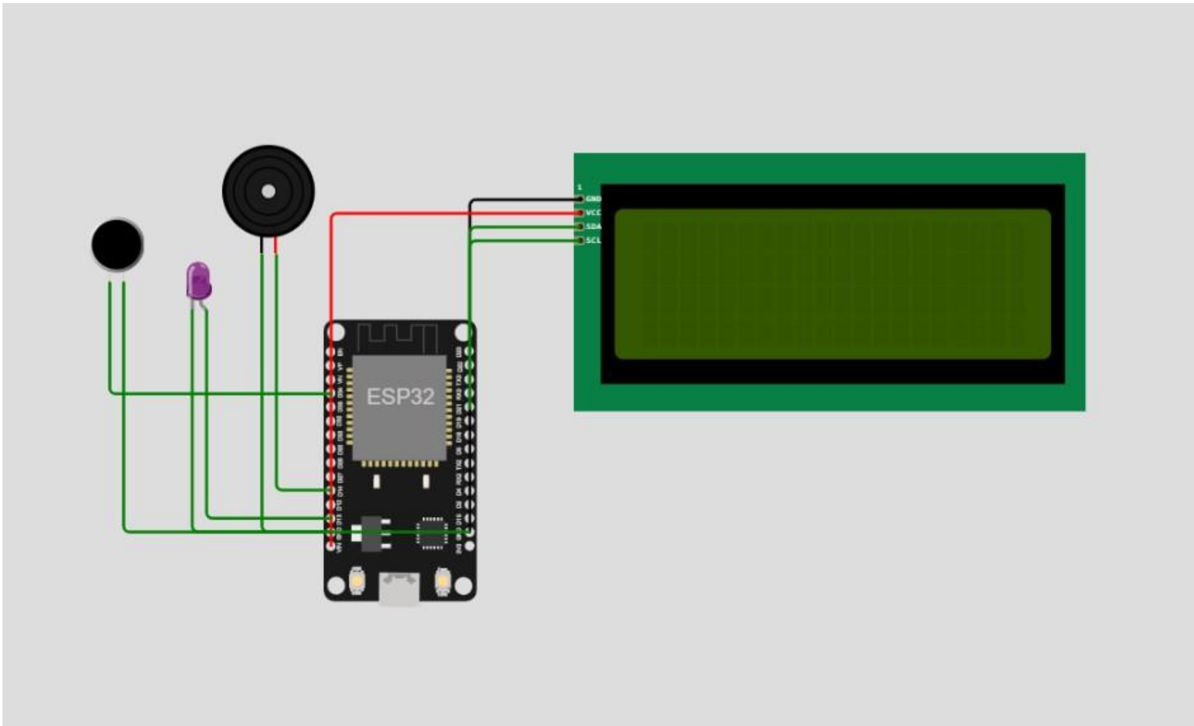
Creating the entire codebase for an IoT-enabled noise pollution monitoring system is a complex task that goes beyond the scope of a single response. However, I can provide you with a simplified example of code that you can use as a starting point. This example will demonstrate how to read data from a simple microphone sensor connected to a Raspberry Pi and send it to a server. Please note that this is a basic implementation, and a real-world system would be more complex.

COMPONENTS REQUIRED:

- ESP32 Board
- LCD Display
- Microphone Sensor
- Buzzer
- LED

Name: Lingeshwaran G

CIRCUIT DIAGRAM:



PYTHON CODE:

```
import time
import math
from machine import ADC, Pin, I2C, PWM
from lcd_api import LcdApi
from pico_i2c_lcd import I2cLcd
import network
import urequests as requests

# Define ADC pin for the microphone
mic_pin = ADC(Pin(34))
```

Name: Lingeshwaran G

Initialize I2C

i2c = I2C(0, sda=Pin(21), scl=Pin(22), freq=400000)

Initialize LCD display with your specific settings

I2C_ADDR = 39

I2C_ROWS = 4

I2C_COLS = 20

lcd = I2cLcd(i2c, I2C_ADDR, I2C_ROWS, I2C_COLS)

Define the buzzer and LED pins

buzzer_pin = Pin(14)

led_pin = Pin(13, Pin.OUT)

Define the noise threshold in dB

noise_threshold = 60 # Adjusted to 60 dB

Microphone sensitivity (in dB per Volt) - replace with your microphone's sensitivity

MIC_SENSITIVITY = 3.0

Define your Wi-Fi credentials

WIFI_SSID = "Wokwi-GUEST"

WIFI_PASS = ""

Initialize Wi-Fi

Name: Lingeshwaran G

```
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(WIFI_SSID, WIFI_PASS)

# Wait until Wi-Fi connection is established
while not wifi.isconnected():
    pass

# Create a PWM object for the buzzer
buzzer_pwm = PWM(buzzer_pin)
buzzer_pwm.deinit()  # Turn off the buzzer at the beginning

# Define a function to calculate sound pressure level (SPL)
def calculate_noise_level(adc_value, reference_voltage, sensitivity):
    # Calculate noise level in decibels (dB) based on sensitivity and voltage
    noise_db = 20 * math.log10(adc_value / (reference_voltage * sensitivity))
    return noise_db

# Define a function to update noise level and control the buzzer and LED
def update_noise_level():
    global noise_level

    # Read ADC values from the microphone
    mic_value = mic_pin.read()
```

Name: Lingeshwaran G

```
    # Set a constant reference voltage based on your system's maximum voltage
    reference_voltage = 5.0  # Assuming 5V maximum reference voltage

    # Calculate noise level
        noise_level = calculate_noise_level(mic_value, reference_voltage,
MIC_SENSITIVITY)  # Update noise_level directly

    # Control the buzzer and LED based on the noise level
    if noise_level > noise_threshold:
        # Turn on the buzzer and LED
        buzzer_pwm.freq(1000)
        led_pin.on()
    else:
        # Turn off the buzzer and LED
        buzzer_pwm.deinit()
        led_pin.off()

# Define a function to display noise level status
def display_noise_status():
    if noise_level < noise_threshold - 10:
        status = "Quiet"
    elif noise_level >= noise_threshold - 10 and noise_level < noise_threshold + 10:
        status = "Normal"
    else:
        status = "High"
```

Name: Lingeshwaran G

```
    # Display noise level status and the actual noise level on the LCD and serial monitor
    lcd.clear()
    lcd.move_to(0,0)
    lcd.putstr("Loudness: {:.2f}dB".format(noise_level))
    lcd.move_to(0,2)
    lcd.putstr("Level: {}".format(status))

    print("Loudness: {:.2f} dB".format(noise_level))
    print("Level: {}".format(status))

# Main loop with continuous data transmission
first_data_sent = False  # Track if the first data is sent

while True:
    # Update noise level
    update_noise_level()

    # Display noise level status
    display_noise_status()

    # Send data to your server continuously after the first data
    if first_data_sent:
        data = {
            "noise_level": noise_level
        }
```

```
        response = requests.post(SERVER_URL, json=data)

    if not first_data_sent:
        first_data_sent = True
# Mark the first data as sent
```

SENSOR DEPLOYMENT:

Sensor deployment in an IoT-enabled noise pollution monitoring system typically involves configuring and installing sensors in the desired locations. Here’s a simplified Python code snippet that simulates sensor deployment. In a real-world scenario, you would need to physically install the sensors and configure them accordingly.

DATA ACQUISITION:

Data acquisition in an IoT-enabled noise pollution monitoring system involves collecting data from sensors. In a real-world implementation, you would interface with specific sensor hardware, but I can provide you with a simple example of data acquisition code that reads simulated noise levels from sensors.

Name: Lingeshwaran G

ALERTING SYSTEM:

Certainly, here’s a simplified Python code snippet for an alerting system that sends an alert message if a noise level threshold is exceeded.

USER INTERFACE:

Creating a user interface for your noise pollution monitoring system typically involves using web-based technologies. Here’s a simplified example of how to create a basic web-based user interface using HTML and Python’s Flask framework for a local implementation.

CONCLUSION:

In conclusion, the Noise Pollution Monitoring system, built with an ESP32 board and Micro Python, provides a robust and versatile solution for real-time noise level measurement and data transmission. This project combines hardware components, threshold-based alerts, data management, and ethical considerations to offer a valuable tool for monitoring noise pollution in public spaces. With continuous noise measurement, user-friendly interfaces, and data transmission capabilities, it empowers users to make informed decisions, contributing to environmental awareness and public health considerations. The system's adaptable design ensures it can be employed for a range of applications, from urban planning to research, fostering a deeper understanding of noise pollution's impact on our surroundings.

Name: Lingeshwaran G

THANK YOU