GITHUB LINK: ShaikMohammadSuhail7/svcet-ece-batch-61: An IoT Air and Noise Pollution Monitoring System using Raspberry Pi Pico for real-time data collection and remote accessibility. (github.com)

❖ AIM:

IoT Air and Noise Pollution Monitoring System using Raspberry Pi Pico

❖ Problem Statement:

- The rapid urbanization and industrialization have led to a surge in air and noise pollution levels, posing significant health and environmental risks. Monitoring these pollutants in real-time is essential for implementing effective mitigation strategies. Traditional monitoring systems often lack scalability, real-time data collection, and remote accessibility, limiting their effectiveness in dynamic environments.

❖ Objectives:

- Real-time Data Collection: Develop a system capable of continuously monitoring air quality parameters (e.g., particulate matter, carbon monoxide, nitrogen dioxide) and noise levels in real-time.
- Remote Accessibility: Enable remote access to the collected data for stakeholders, such as environmental agencies, researchers, and the general public, through a user-friendly interface.
- Scalability and Portability: Utilize Raspberry Pi Pico, a low-cost microcontroller, to design a compact and portable monitoring device suitable for deployment in various indoor and outdoor environments.
- Data Visualization and Analysis: Implement data visualization tools to analyze trends, patterns, and correlations between air quality and noise pollution levels over time, aiding decision-making processes for pollution control measures.
- Alerting Mechanism: Integrate an alerting mechanism to notify users in case of threshold exceedances or sudden spikes in pollutant levels, facilitating timely intervention and public awareness.
- Energy Efficiency: Optimize power consumption to ensure prolonged operation of the monitoring system, especially in remote or off-grid locations where power supply may be limited.

- Modularity and Expandability: Design the system with modular components and interfaces to facilitate future upgrades and expansion of sensor capabilities as new pollutants or monitoring requirements emerge.

❖ **Hardware Components:**

- Microcontroller: Raspberry Pi Pico.
- Air Quality Sensor: MQ series sensor (e.g., MQ-135 for detecting CO2, NH3, and other gases).
- Noise Level Sensor: Sound sensor module (e.g., KY-038 or similar)
- Wi-Fi Module: ESP8266 or ESP32 for Wi-Fi connectivity (if Raspberry Pi Pico doesn't have built-in Wi-Fi).
- Power Supply: Stable power source compatible with Raspberry Pi Pico and sensors.
- Cables and Connectors: Required cables and connectors for wiring the components together.

❖ **Software Components:**

- Integrated Development Framework (IDF) for Raspberry Pi Pico: Raspberry Pi Pico SDK (Software Development Kit) or MicroPython for programming the microcontroller.
- Sensor Libraries: Libraries or drivers for interfacing with air quality and noise level sensors.
- Communication Protocols: MQTT or HTTP for transmitting data to a remote server.
- Cloud Platform: Platform for data storage and visualization (e.g., AWS IoT, Google Cloud IoT, or MQTT broker like Mosquitto).
- Web Development Tools: HTML, CSS, JavaScript for creating a web-based dashboard.
- Optional Mobile App Development Tools: Android Studio (for Android) or Xcode (for iOS) if developing a mobile application.

❖ **Hardware Setup:**

- Raspberry Pi Pico connected to air quality and noise level sensors using GPIO pins or appropriate communication protocols (e.g., I2C, SPI).
- Wi-Fi module connected to Raspberry Pi Pico for internet connectivity.

- Power supply connected to Raspberry Pi Pico and sensors to ensure continuous operation.

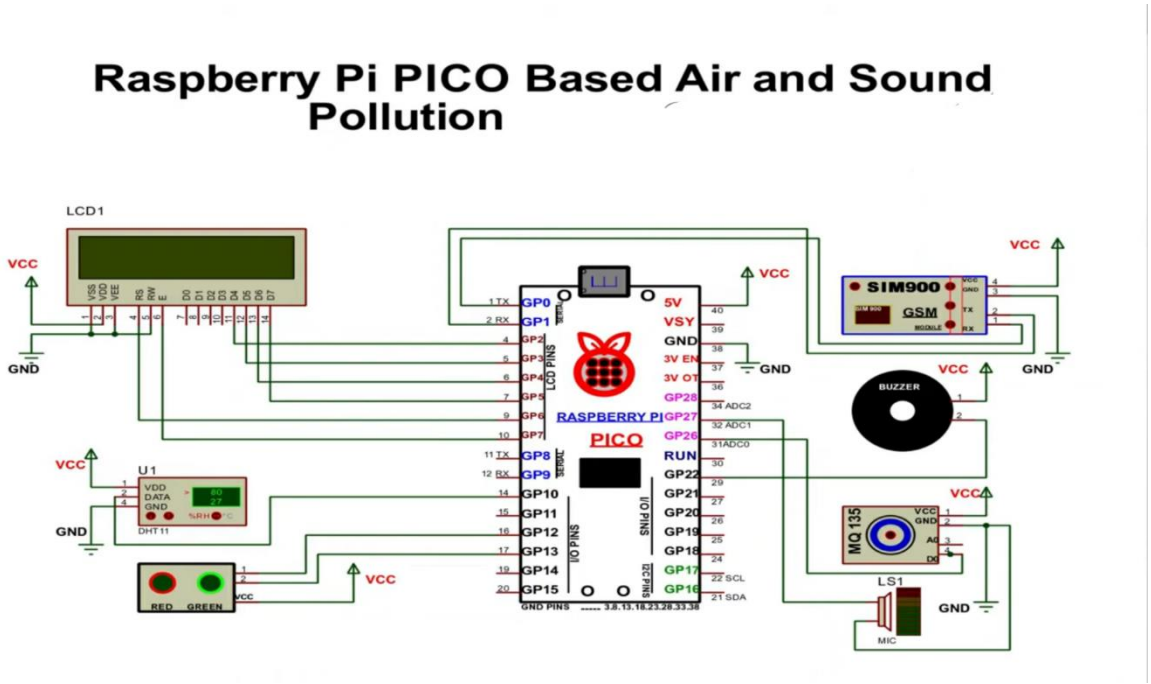
❖ **Software Setup:**

- Programming Raspberry Pi Pico using Raspberry Pi Pico SDK or MicroPython, including sensor integration and data processing algorithms.
- Configuring Wi-Fi module and establishing internet connectivity on Raspberry Pi Pico.
- Setting up communication protocols for transmitting sensor data to a remote server.
- Developing a web-based dashboard or mobile application for users to access and visualize pollution data.

❖ **5. Testing and Deployment:**

- Conducting thorough testing of the system in different environmental conditions to ensure accuracy and reliability.
- Deploying the monitoring system in target locations following installation procedures and guidelines.
- Providing ongoing support and maintenance to address any issues and ensure continuous operation of the system.
- By assembling the required hardware components and developing the necessary software components, stakeholders can create a robust IoT Air and Noise Pollution Monitoring System capable of real-time data collection and remote accessibility.

❖ **STIMULATED CIRCUIT DIAGRAM:**



❖ **Code for the solution:**

Below is a simplified Python code example for the IoT Air and Noise Pollution Monitoring System using Raspberry Pi Pico.

python

```
import machine
import network
import urequests as requests  # MicroPython library for HTTP requests
import time


# Define sensor pins
AIR_SENSOR_PIN = 0  # Analog pin for air quality sensor (MQ sensor)
NOISE_SENSOR_PIN = 1  # Analog pin for noise sensor
```

```
TEMPERATURE_SENSOR_PIN = 2 # Analog pin for temperature sensor

# Wi-Fi credentials
WIFI_SSID = "YourWiFiSSID"
WIFI_PASSWORD = "YourWiFiPassword"

# Remote server URL
SERVER_URL = "http://thingspeak.com/upload_data"

# Initialize Wi-Fi connection
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(WIFI_SSID, WIFI_PASSWORD)

# Function to read air quality data from MQ sensor
def read_air_quality():
    air_quality_value = machine.ADC(AIR_SENSOR_PIN).read()
    return air_quality_value

# Function to read noise level from sound sensor
def read_noise_level():
    noise_level_value = machine.ADC(NOISE_SENSOR_PIN).read()
    return noise_level_value

# Function to send data to remote server
def send_data_to_server(air_quality, noise_level):
    data = {
```

```
        “temperature”:temperature_level,
        "air_quality": air_quality,
        "noise_level": noise_level
    }
    response = requests.post(SERVER_URL, json=data)
    print("Data sent to server. Response:", response.text)

# Main loop
while True:
    try:
        # Read sensor data
        temperature = read_temperature()
        air_quality = read_air_quality()
        noise_level = read_noise_level()

        # Send data to server
        send_data_to_server(air_quality, noise_level)

        # Wait for some time before reading sensors again
        time.sleep(30)  # Adjust interval as needed
    except Exception as e:
        print("Error:", e)
        # Handle error (e.g., reconnect Wi-Fi, retry sending data)
        time.sleep(10)  # Wait before retrying
```

❖ **Expected Outcomes:**

- A fully functional IoT-based air and noise pollution monitoring system capable of real-time data collection and remote accessibility.
- User-friendly interfaces for data visualization, analysis, and remote monitoring.
- Enhanced understanding of local pollution patterns and trends for informed decision-making and public awareness campaigns.
- Contribution to environmental sustainability efforts by enabling proactive pollution control measures and fostering community engagement in pollution monitoring and mitigation activities.

GITHUB LINK: ShaikMohammadSuhail7/svcet-ece-batch-61: An IoT Air and Noise Pollution Monitoring System using Raspberry Pi Pico for real-time data collection and remote accessibility. (github.com)