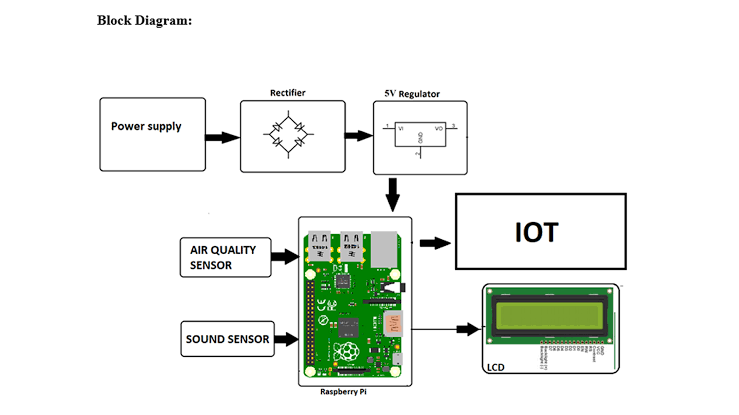
**Noise pollution monitoring**

****

**Python program**

# Import necessary libraries⁹

import os

import time

import datetime

import requests

import sounddevice as sd

import numpy as np

# Define API endpoint for data transmission

API\_ENDPOINT = "https://your-api-url-for-storing-data.com"

# Define configuration parameters

SAMPLE\_RATE = 44100 # Sample rate for audio recording

RECORDING\_DURATION = 10 # Duration of each recording in seconds

THRESHOLD = 70 # Noise threshold level for alert (adjust as needed)

# Function to record audio and analyze noise level

def record\_and\_analyze\_noise():

print("Recording...")

audio\_data = sd.rec(int(SAMPLE\_RATE \* RECORDING\_DURATION), samplerate=SAMPLE\_RATE, channels=1)

sd.wait()

# Calculate the root mean square (RMS) value to assess noise level

rms = np.sqrt(np.mean(audio\_data\*\*2))

print(f"Noise level (RMS): {rms} dB")

# Send data to the server

send\_data\_to\_server(rms)

# Check if noise level exceeds the threshold

if rms > THRESHOLD:

alert\_noise()

# Function to send data to the server

def send\_data\_to\_server(noise\_level):

data = {

"timestamp": datetime.datetime.now().isoformat(),

"noise\_level": noise\_level

}

response = requests.post(API\_ENDPOINT, json=data)

if response.status\_code == 200:

print("Data sent to server successfully.")

else:

print("Failed to send data to the server.")

# Function to trigger an alert for high noise levels

def alert\_noise():

print("High noise level detected! Triggering an alert...")

# Implement your alert mechanism here (e.g., email, SMS, IoT device)

# Main loop for continuous noise monitoring

while True:

record\_and\_analyze\_noise()

time.sleep(60) # Record and analyze noise every 60 seconds

**1. Hardware Setup:**

- Choose suitable noise sensors (e.g., microphones).

- Connect the sensors to a microcontroller (e.g., Raspberry Pi or Arduino) with ADC capabilities.

**2. Data Acquisition:**

- Write code to capture analog data from the noise sensor.

- Convert analog data to digital values using ADC.

**3. Data Processing:**

- Implement algorithms to filter and process the noise data.

- Calculate noise levels in decibels (dB).

**4. Data Storage:**

- Store data in a database or file for future analysis.

**5. Real-time Monitoring:**

- Set up continuous monitoring and data collection.

**6. Alerting System:**

- Implement thresholds for noise levels.

- Send alerts (e.g., email or SMS) when noise exceeds a defined limit.

**Conclusion**

This is a high-level overview of the process. Each step will involve writing Python code, and the specifics will depend on your chosen hardware and the level of complexity you want to achieve. You may also need to consider legal and ethical aspects of noise monitoring, especially if it involves monitoring public spaces.