**PROJECT TITLE: NOISE POLLUTION MONITORING**

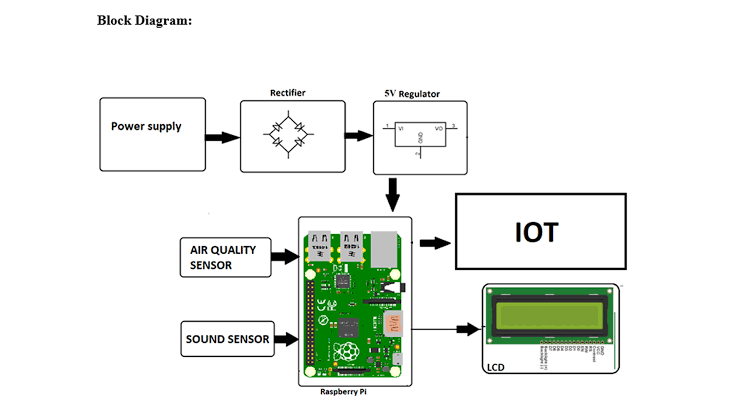
**Phase 4: Development Part 2**

**INTRODUCTION:**

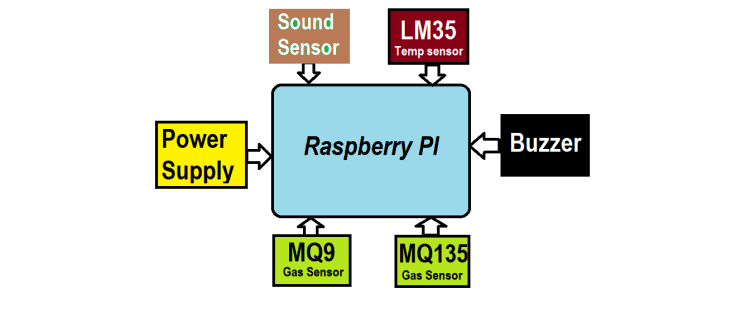
Noise pollution, a pervasive and often overlooked environmental issue, can have detrimental effects on health, well-being, and the quality of life in urban and even rural areas. To address this concern and promote awareness, we introduce a Raspberry Pi-based noise pollution monitoring project. This project utilizes the capabilities of the Raspberry Pi, coupled with a USB microphone and Python programming, to continuously monitor and analyze ambient noise levels.The Raspberry Pi, a versatile single-board computer, is an ideal platform for this endeavor due to its affordability, low power consumption, and ease of integration with various sensors and peripherals. By deploying this solution, individuals, communities, or organizations can gain valuable insights into the noise pollution in their surroundings, allowing them to make informed decisions, take corrective actions, and advocate for a quieter environment.

**BLOCK DIAGRAM:**

The Proposed model of the system is as follows. Figure shows how the whole system will work. The block diagram of the system is showing that for a particular area selected how will it work. The device will be set up to take the environmental data and there will be a base standard value. The device will collect data and based on the set values it

[](https://www.google.com/search?safe=active&q=block+diagram+for+noise+pollution+monitoring+using+raspberry+pi+#vhid=n1wciyVw-ZdeMM&vssid=l)

The system is controlled using Raspberry Pi 3b+ which is interfaced with gas sensors like MQ135, MQ7, MQ4 and also the Dust particle sensor. The DHT11 sensor updates the Temperature and Humidity values on real time basis. The complete monitored values from the sensors are updated on the cloud i.e., Smart Core. The entire data is stored in the cloud and the values will be updated on real time basis. The data updating and the flow its works is as shown in the below flow diagram

[](https://www.google.com/search?safe=active&q=block+diagram+for+noise+pollution+monitoring+using+raspberry+pi+#vhid=7S0Rp_37qGOoEM&vssid=l)

#### **PLATFORM REQUIRED:**

#### **HARDWARE Components Used:**

###### **Raspberry Pi 3b+:**

The Raspberry Pi 3 Model B+ is the latest product in the Raspberry Pi 3 range, boasting a 64-bit quad core processor running at 1.4GHz, dual-band 2.4GHz and 5GHz wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and PoE capability via a separate PoE HAT The dual-band wireless LAN comes with modular compliance certification, allowing the board to be designed into end products with significantly reduced wireless LAN compliance testing, improving both cost and time to market. The Raspberry Pi 3 Model B+ maintains the same mechanical footprint as both the Raspberry Pi 2 Model B and the Raspberry Pi 3 Model B. 

**Specifications:**

Processor: Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4GHz

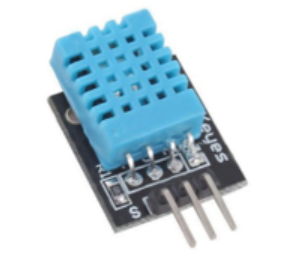
Memory: 1GB LPDDR2 SDRAM

**Connectivity:**

* 4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
* Gigabit Ethernet over USB 2.0 (maximum throughput 300Mbps)
* 4 × USB 2.0 portsSpecifications

**DHT11:**

### **DHT11 Specifications:**

* Operating Voltage: 3.5V to 5.5V
* Operating current: 0.3mA (measuring) 60uA (standby)
* Output: Serial data
* Temperature Range: 0°C to 50°C
* Humidity Range: 20% to 90%
* Resolution: Temperature and Humidity both are 16-bit
* Accuracy: ±1°C and ±1%

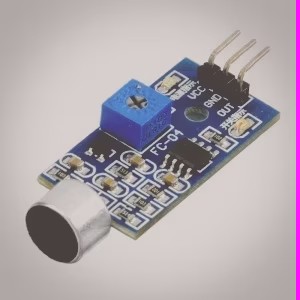
##### **MQ135 Air quality Sensor:**

**Features:**

* High Sensitivity
* High sensitivity to Ammonia, Sulfide and Benze
* Stable and Long Life
* Detection Range: 10 – 300 ppm NH3, 10 – 1000 ppm Benzene, 10 – 300 Alcohol
* Heater Voltage: 5.0V
* Dimensions: 18mm Diameter, 17mm High excluding pins, Pins – 6mm High
* Long life and low cost

# **Sound Sensor**:

The sound sensor is the same as other sensors like IR and Soil moisture but as always functioning of each sensor is the different but basic structure and design are common. Sound sensors are available in various variants ranging from the basic one built out of LM393 IC to the Sparkfun module. Also, analog and digital sensors differ very much in their function and performance.



* **Air Quality Sensor:** Choose a sensor like the SDS011, CCS811, or Nova PM SDS012 for measuring particulate matter (PM2.5 and PM10) and other pollutants.
* **Temperature and Humidity Sensor:** DHT11 or DHT22 sensors are commonly used for monitoring temperature and humidity levels.



* **Power Supply:** A stable power supply for the Raspberry Pi and the sensors.
* **MicroSD Card:** To install the operating system and software.
* **Wiring and Breadboard:** For connecting sensors to the Raspberry Pi GPIO pins.
* **Enclosure:** To protect your Raspberry Pi and sensors from environmental factors

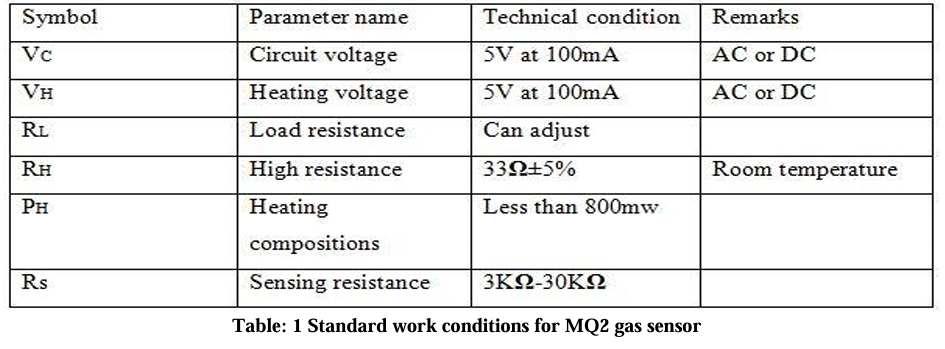
**Software Requirements:**

* **Raspberry Pi OS:** Install a Raspberry Pi-compatible operating system on the microSD card. Raspbian is the most popular choice.
* **Python:** You'll need Python installed on your Raspberry Pi to run the sensor scripts.
* **Sensor Libraries**: Depending on your sensors, you might need specific Python libraries to communicate with them. For example, `sds011\_python` for SDS011 sensor.
* **Database**: Choose a database system like MySQL, SQLite, or MongoDB to store the sensor data.
* **Web Server**: Flask or Django can be used to create a web interface for displaying real-time and historical data.
* **Cron Jobs**: If you want to schedule regular data readings, use cron jobs in Linux to automate the process.
* **Data Visualization Tools**: You can use tools like Grafana or Matplotlib (for Python) to create graphs and visualize the collected data.
* **Remote Monitoring (Optional):** If you want to monitor the air quality remotely, you might need services like Ngrok for secure tunneling or setting up a VPN for remote access.

**WORKING STEPS:**

**Step 1**: Connect the MQ-135 Sensor to Raspberry Pi

* Connect the VCC pin of the MQ-135 sensor to the 5V pin on the Raspberry Pi.



* Connect the GND pin of the MQ-135 sensor to the GND pin on the Raspberry Pi.
* Connect the AOUT pin of the MQ-135 sensor to any available GPIO pin on the Raspberry Pi (e.g., GPIO17).

**Step 2**: Install Required Python Libraries

Python Libraries:

PyAudio: For capturing audio data from the microphone.

numpy: Useful for data manipulation and analysis.

matplotlib: If you plan to visualize noise data

SciPy: It can be used for more advanced signal processing and analysis.

SoundMeter: A Python library specifically designed for noise level measurement.

**Step 3**: Write Python Code for Noise Pollution Monitoring

import sounddevice as sd

# Define the audio recording parameters

duration = 10 # Recording duration in seconds

sample\_rate = 44100 # Sampling rate (you may need to adjust this)

# Function to record audio and calculate noise level

def record\_audio\_and\_calculate\_noise\_level():

print("Recording audio...")

audio\_data = sd.rec(int(duration \* sample\_rate), samplerate=sample\_rate, channels=1)

sd.wait() # Wait for recording to complete

print("Recording complete")

# Calculate the noise level (you may need to adjust the threshold)

noise\_level = max(audio\_data)

print(f"Noise level: {noise\_level}")

if \_\_name\_\_ == "\_\_main\_\_":

record\_audio\_and\_calculate\_noise\_level()

**Step 4:** Run the Python Script

**Python Code for Noise Pollution Monitoring System:**

import alsaaudio # This library is used to capture audio on Raspberry Pi

import numpy as np

# Define the audio recording parameters

sample\_rate = 44100 # Sampling rate (you may need to adjust this)

duration = 10 # Recording duration in seconds

# Function to record audio and calculate noise level

def record\_audio\_and\_calculate\_noise\_level():

inp = alsaaudio.PCM(alsaaudio.PCM\_CAPTURE)

inp.setchannels(1)

inp.setrate(sample\_rate)

inp.setformat(alsaaudio.PCM\_FORMAT\_S16\_LE)

inp.setperiodsize(1024)

audio\_data = np.array([], dtype='int16')

print("Recording audio...")

for \_ in range(0, int(sample\_rate / 1024 \* duration)):

l, data = inp.read()

audio\_data = np.append(audio\_data, np.frombuffer(data, dtype=np.int16))

print("Recording complete")

# Calculate the noise level (you may need to adjust the threshold)

noise\_level = np.max(np.abs(audio\_data))

print(f"Noise level: {noise\_level}")

if \_\_name\_\_ == "\_\_main\_\_":

record\_audio\_and\_calculate\_noise\_level()

**CONCLUSION:**

In conclusion, noise pollution monitoring using a Raspberry Pi is a feasible and cost-effective solution for assessing and managing noise levels in various environments. This approach offers several advantages, including flexibility, scalability, and the ability to integrate with other systems.