Overview :

We are creating the Noise Pollution Monitor using Raspberry Pi. Noise monitoring refers to the systematic process of measuring, recording, and assessing sound levels in various environments to understand the extent of noise pollution and its potential impact on human health and the surrounding ecosystem.

Components needed :

* Raspberry Pi (any model with GPIO pins)
* LM393 sound sensor module
* Jumper wires
* Battery

The Circuit Design:

The components were connected as given below,

* Connect the VCC (Voltage) and GND (Ground) pins of the LM393 to the corresponding 3.3V and GND pins on the Raspberry Pi.
* Connect the DO pin of the LM393 to any GPIO pin on the Raspberry Pi (e.g., GPIO17)

The Libraries we use in both the codes are mentioned below:

Start by importing the necessary libraries,

import sounddevice as sd

import numpy as np

import datetime

import time

import matplotlib.pyplot as plt

import pandas as pd

Python code :

import sounddevice as sd

import numpy as np

import datetime

import time

# Define the sample rate and monitoring interval

sample\_rate = 44100 # 44.1 kHz (CD quality)

monitoring\_interval = 60 # Log data every 60 seconds

# Define a function to capture audio and log noise level

def monitor\_and\_log\_noise(filename):

print("Continuous noise pollution monitoring... Press Ctrl+C to stop.")

with open(filename, 'a') as log\_file:

log\_file.write("Timestamp,Noise Level (dB)\n")

while True:

try:

# Capture audio data for the defined interval

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

sd.wait()

# Calculate the average amplitude (volume) of the captured audio

average\_amplitude = np.mean(np.abs(audio\_data))

# Convert amplitude to dB

noise\_level\_db = 20 \* np.log10(average\_amplitude + np.finfo(float).eps)

# Log the noise level along with a timestamp

timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")

with open(filename, 'a') as log\_file:

log\_file.write(f"{timestamp},{noise\_level\_db}\n")

time.sleep(monitoring\_interval) # Wait for the next monitoring interval

except KeyboardInterrupt:

print("Monitoring stopped. Data logged to", filename)

break

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

log\_file\_name = "continuous\_noise\_log.csv"

try:

monitor\_and\_log\_noise(log\_file\_name)

except KeyboardInterrupt:

pass

Output:

Timestamp,Noise Level (dB)

2023-10-18 14:27:19,-56.33993661839761

2023-10-18 14:29:19,-54.21177509240552

2023-10-18 14:31:20,-55.1714153729947

2023-10-18 14:33:20,-56.0274769244086

2023-10-18 14:35:21,-57.79226210252923

2023-10-18 14:37:21,-51.21254388956936

2023-10-18 14:39:22,-55.48773749912264

2023-10-18 14:41:22,-51.86537350467389

2023-10-18 14:43:23,-55.082891706144075

2023-10-18 14:45:23,-58.522554367577726

2023-10-18 14:47:24,-56.145810056652294

2023-10-18 14:49:24,-55.58021754572336

2023-10-18 14:51:25,-56.60603265216247

2023-10-18 14:53:25,-56.42059782033996

2023-10-18 14:55:26,-63.08700453831118

2023-10-18 14:57:26,-58.05196802353254

2023-10-18 14:59:27,-57.717039431714966

2023-10-18 15:01:27,-60.072241948784935

2023-10-18 15:03:28,-59.464334014026136

2023-10-18 15:05:28,-62.71987954751011

2023-10-18 15:07:28,-54.436930736318956

2023-10-18 15:09:29,-52.49044394231045

2023-10-18 15:11:29,-53.864939329007115

2023-10-18 15:13:30,-53.94988123889904

2023-10-18 15:15:30,-53.61903294622848

2023-10-18 15:17:31,-59.85362317821759

2023-10-18 15:19:31,-55.33882613133922

Python code to show the obtained data in graph :

import matplotlib.pyplot as plt

import pandas as pd

# Read the CSV file containing the noise level data

log\_file\_name = "continuous\_noise\_log.csv"

# Load the CSV data into a DataFrame

df = pd.read\_csv(log\_file\_name)

# Extract the timestamp and noise level columns

timestamps = df['Timestamp']

noise\_levels = df['Noise Level (dB)']

# Convert timestamps to datetime objects

timestamps = pd.to\_datetime(timestamps)

# Create a line plot to visualize the noise levels over time

plt.figure(figsize=(10, 6))

plt.plot(timestamps, noise\_levels, marker='o', linestyle='-', color='b')

plt.title("Continuous Noise Level Monitoring")

plt.xlabel("Timestamp")

plt.ylabel("Noise Level (dB)")

plt.grid(True)

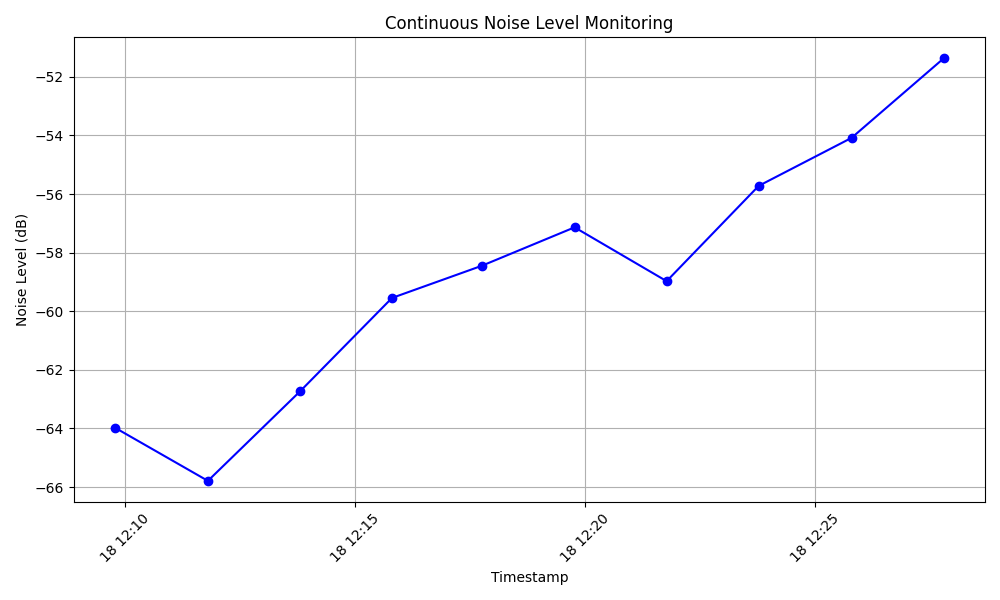
plt.xticks(rotation=45) # Rotate x-axis labels for better readability

# Show the graph

plt.tight\_layout()

plt.show()

Output:



The above chart is a representation of the continuous sound frequency in decibels.

Conclusion:

The code is used in the Raspberry Pi, to capture the sound levels and log the data in a CSV (Excel) file with date and time, the sound level is now currently set to update every 60 seconds (Can be changed according to our need).

This is noise modelling software which can simulate noise level at traffic and inplant noises, or any public places. This is good tool to create detailed noise map using data gathered from Noise sensors and computer models.