

Computer Networks

Practical 2

Sound Monitoring Sensor

Report

Group 3

Due date: 18/04/25

Members:

Surname	Name	Student #
Habiyaremye	Patrick	4323665
Jones	Skye	4122217
Cloete	Josh	4024911
Cornelius	Uwais	4121659
Khuzwayo	Minenhle	4260321
Zulu	Sibusiswe	4270237
Mlambo	Makhanani	4270025
Mtsolongo	Lilitha	4341922
Buso	Sixolile	4241076
Dladla	Simamkele	4369463
Mano	Lavhelani	4165489

Contents

Contents	2
1. Abstract	3
2. Introduction.	3
3. Literature Review	4
Why is this project relevant?	4
What has been done before?	4
How does this project differ?	5
Key technologies	5
4. Hardware and Design	6
5. Technical Implementation	8
6. Conclusion	10
7. References	10
8. Group member contributions	11

1. Abstract

Noise pollution provides challenges in multiple settings, including classrooms, offices, homes and industrial environments. This project aims to develop a Sound Level Monitoring System using a Raspberry Pi and a sound sensor to measure and analyze ambient noise levels in real-time. The device captures sound intensity, processes the data, and transmits it to a client device (laptop/mobile) via a UDP (User Datagram Protocol) network protocol. If the noise exceeds a particular threshold (e.g., 80 dB), the user(s) will be notified of excessive noise levels.

For real-time, low-latency communication, UDP (User Datagram Protocol) is chosen as the transmission protocol due to its speed and efficiency. UDP ensures minimal delay in noise level updates, making it ideal for continuous monitoring applications. The expected outcome is a reliable, real-time noise monitoring system that can be used in noise-sensitive environments to improve awareness and compliance with acceptable noise limits.

2. Introduction.

Have you ever stopped to think about how much noise affects our daily lives? From workplaces to schools and even homes, excessive noise can have devastating consequences, reducing productivity, impacting health, and lowering quality of life. To address this issue, the Sound Level Monitoring System has been designed to provide real-time noise monitoring, enabling users to manage their sonic environment effectively. This system utilizes a Raspberry Pi and a sound sensor to detect ambient noise, process data, and transmit it to a client device via UDP. If noise levels exceed a predefined threshold, users receive instant alerts, allowing them to take timely action.

This technology has several real-world applications. In workplaces, it helps prevent hearing damage from prolonged noise exposure. In classrooms and offices, it promotes a more focused environment by monitoring and controlling noise levels. For new parents, it can be especially valuable. Newborns require frequent feeding and diaper changes, which disrupt parents' sleep patterns, leading to exhaustion. The Sound Level Monitoring System allows parents to monitor and manage noise levels, creating a better sleep environment for both babies and parents. Additionally, it can be integrated into hospitals to maintain a peaceful atmosphere, supporting patient recovery.

Ultimately, this system not only enhances safety and focus on various environments but also promotes better noise management and awareness, making it a valuable tool for improving overall well-being.

3. Literature Review

Why is this project relevant?

Have you ever found yourself in an environment where you just want to relax, take a deep breath and just be lost in your train of thoughts? But all that takes a complete turn because the environment is noise polluted. Noise pollution is a huge problem that affects different areas of life like health, productivity, mental health and also well-being of individuals in different environments such as, workplace, schools, homes and Industrial areas.

This project is relevant because it tackles the problem individuals face with noise pollution hence giving them the peace, concentration and tranquility they need. This project is most beneficial to those in the workplace and at school because they need to be focused and undisturbed, so the Sound Level Monitoring System improves concentration and productivity. It also aids in helping new parents seeking to establish a calmer sleep setting for their babies and in medical facilities where ensuring a serene environment aids in patient healing.

What has been done before?

According to Mydlarz, Salamon, and Bello, in the last 10 years, people have come up with different ways to deal with noise in the environment. One way is by using microphones (called acoustic sensors) that are placed in certain spots to listen to sounds. Some of these setups are very expensive and professional, while others are much cheaper and use everyday technology. Because of improvements in microphones, small computers, and wireless internet, even the low-cost sensors are now much smarter. They can hear sounds, figure out what they are right away using special processing, and then send that information wirelessly to other systems. (Mydlarz, Salamon, & Bello, 2016). This makes it easier to track noise across large areas in real time. Because these sensors can vary a lot in how much they cost and what they can do, they are usually divided into three types (dedicated monitoring station, moderately scalable sensor network, and low-cost sensor network). This helps people choose the right kind depending on what kind of noise monitoring they need.

How does this project differ?

This project differs from existing solutions like high-quality and scalable acoustic sensor networks in that it is designed for small-scale environments such as classrooms, offices, and homes. It uses a Raspberry Pi to measure and analyze noise levels in real time. When the noise exceeds a predefined threshold, it alerts the user so they can take immediate action. Unlike larger-scale sensor networks, which collect extensive data, track noise trends over time, and support long-term decision-making, this project focuses on real-time responsiveness and simplicity.

Key technologies

In this project, we will use different types of hardwares and softwares to create an efficient Sound Level Monitoring System. Here are some of the key technologies below:

Raspberry Pi

Raspberry Pi is compact, affordable and a versatile single board computer that we will be using as our central processing unit for our project. It analyses inputs from the sound sensors and checks if those noises made do exceed the limit, and if it does pass the limit, then it will produce a sound.

Sound Sensors

Sound sensors or microphones detect surrounding noises and then send the signals to the Raspberry Pi. They assist in gauging sound levels to allow the system to identify when the noise becomes extremely loud.

Network Protocols (UDP/TCP)

UDP - it is faster and very convenient for real-time signals.

TCP - it ensures reliable data delivery that is also very helpful in logging.

Python Implementation

Python is quite easy to use and it will be very handy in our project. There are also quite a few libraries in python that might be helpful in completing our project like:

socket - to implement UDP and TCP.

RPi.GPIO or gpiozero - to interact with the GPIO pins of the Raspberry Pis.

numpy - to process the signals.

4. Hardware and Design

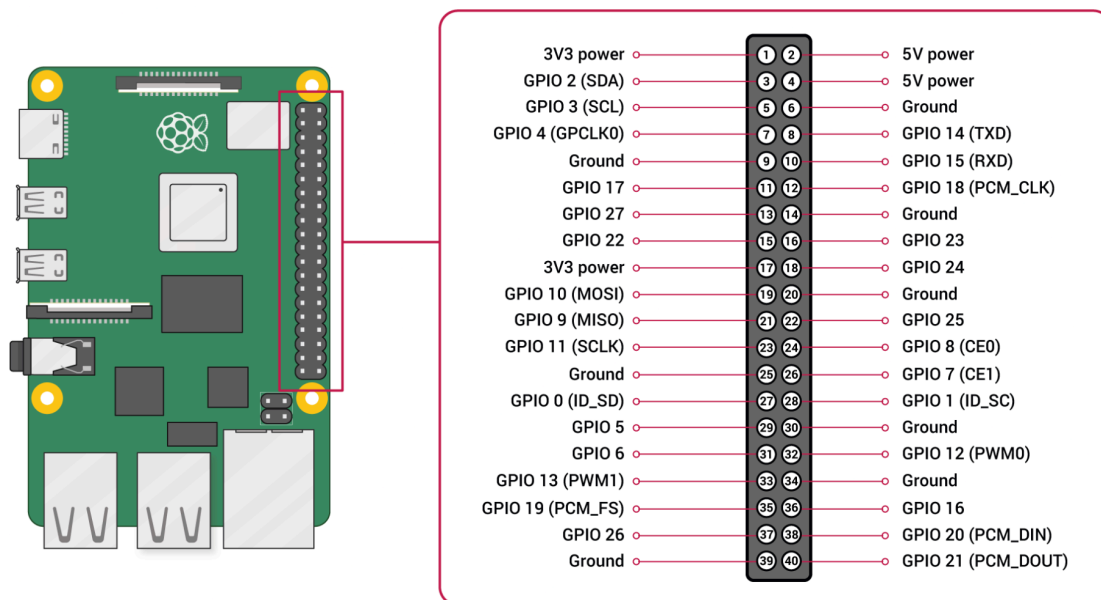
The following hardware components were used in the development of the system:

Raspberry Pi 3: Connected to the sound sensor, it acts as a server to constantly listen to the sound level (whether the threshold has been exceeded or not).

Digital sound sensor: Sends a signal to the Raspberry Pi when noise above the set threshold is detected. The threshold can be adjusted using the potentiometer on the sensor. Using a digital sensor removes the need for analogue to digital conversion, which would be needed when using an analogue sound sensor.

Jumper wires: Establishes physical (electrical) connections between the Raspberry Pi and the sound sensor.

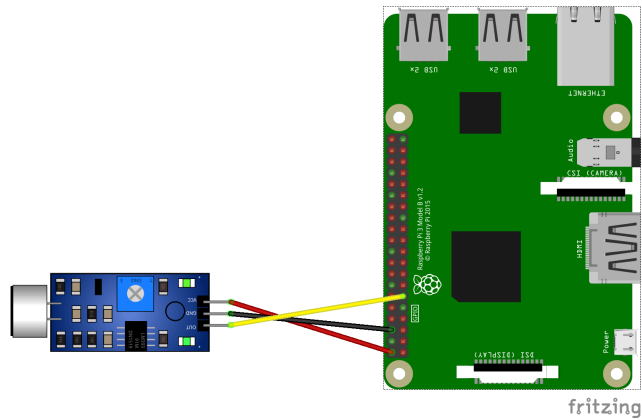
Client Device: A secondary device (e.g., laptop or mobile device) used to receive notifications from the Raspberry Pi server.



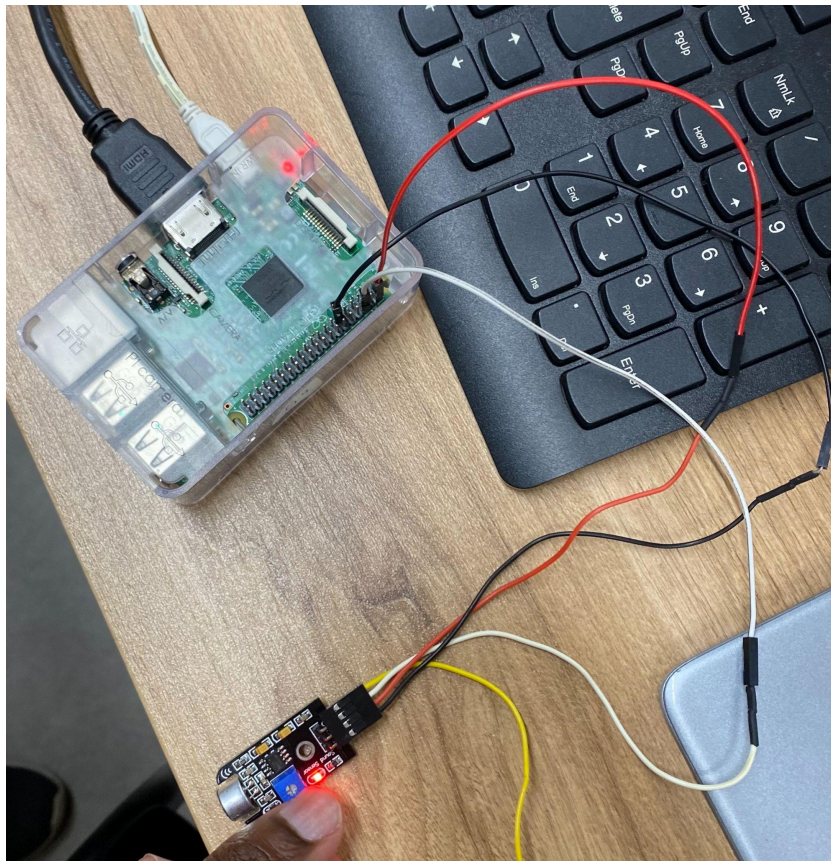
The Raspberry Pi has a 40-pin GPIO header with the layout shown in the image above (Raspberry Pi, 2024).

The sound sensor has three pins, VCC, GND (ground), and digital output. Using the jumper cables, the project was set up as follows:

- The VCC pin on the sensor connects to a 5V pin on the Raspberry Pi (pin 2 was used).
- The GND pin on the sensor connects to a GND pin on the Raspberry Pi (pin 6 was used).
- The digital output pin on the sensor connects to a GPIO pin on the Raspberry Pi (pin 11, GPIO 17, was used).



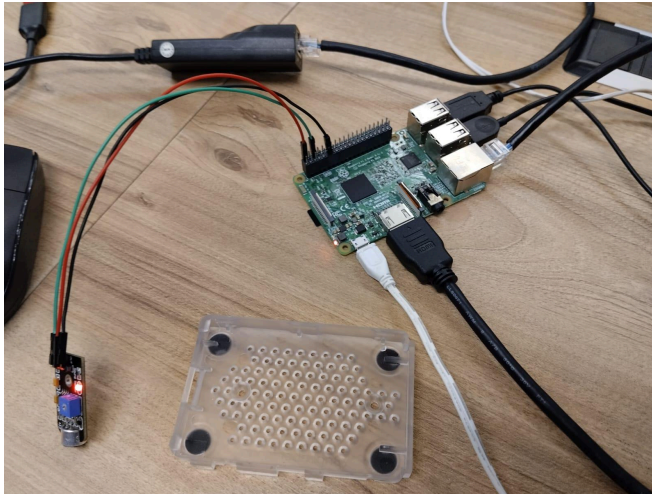
The GPIO connection was used as a communication medium between the input received from the sensor and the Raspberry Pi. This means that the sensor could send a signal to the Raspberry Pi when a certain threshold is reached (The Pi Hut, 2018).



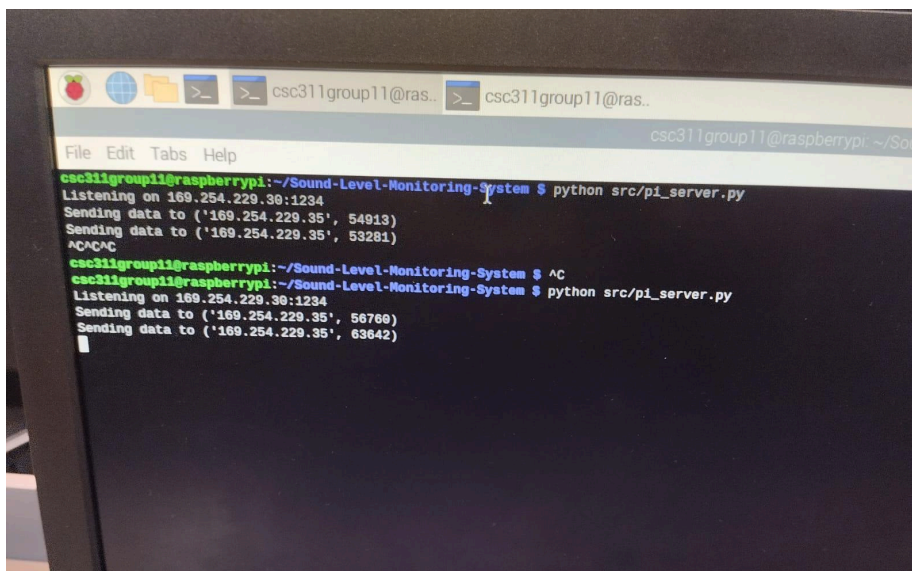
Shown above is the physical setup for the project.

5. Technical Implementation

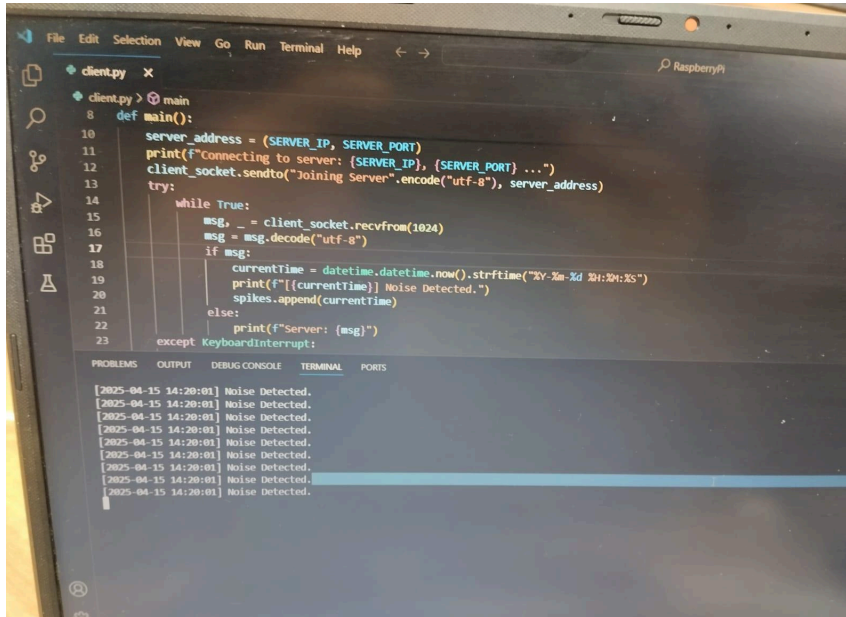
1. Connect with jumper wires the following pins from the sound sensor device to the Raspberry Pi:
 - a. Sound sensor VCC pin to the Raspberry Pi pin 2.
 - b. Sound sensor GND pin to the Raspberry Pi pin 6.
 - c. Sound sensor DOUT pin to the Raspberry Pi pin 17.



2. Connect the client (Laptop / Mobile) and Raspberry Pi on a local network, ensuring that each can communicate with each other.
3. Download the project from github (<https://github.com/mukize/Sound-Level-Monitoring-System>) on the client and Raspberry Pi and follow the instructions in the "README.md" file.
4. Run "src/pi_server.py" on the Raspberry Pi, this starts a UDP server that waits for any incoming UDP packet. Once received, it will send UDP packets back whenever the sound sensor detects noise.



5. Connect to the server by running “src/client.py” on the client device. Once connected to the server via “src/client.py” the UDP packets (indicating noise detection) sent by server to client will be picked up and timestamped.
6. This timestamp and noise detection message will be displayed in real-time to client via Command Line Interface (Laptop via CMD / Mobile via Pydroid3 API)



```
client.py x
client.py > main
8 def main():
10     server_address = (SERVER_IP, SERVER_PORT)
11     print(f'Connecting to server: {SERVER_IP}, {SERVER_PORT} ...')
12     client_socket.sendto("Joining Server".encode("utf-8"), server_address)
13     try:
14         while True:
15             msg, _ = client_socket.recvfrom(1024)
16             msg = msg.decode("utf-8")
17             if msg:
18                 currentTime = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
19                 print(f"[{currentTime}] Noise Detected.")
20                 spikes.append(currentTime)
21             else:
22                 print(f"Server: {msg}")
23     except KeyboardInterrupt:
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
[2025-04-15 14:20:01] Noise Detected.
[2025-04-15 14:20:01] Noise Detected.
[2025-04-15 14:20:01] Noise Detected.
[2025-04-15 14:20:01] Noise Detected.
[2025-04-15 14:20:01] Noise Detected.
[2025-04-15 14:20:01] Noise Detected.
[2025-04-15 14:20:01] Noise Detected.
[2025-04-15 14:20:01] Noise Detected.
[2025-04-15 14:20:01] Noise Detected.
```

6. Conclusion

This project successfully designed and implemented a Sound Level Monitoring System using a Raspberry Pi and a microphone sensor to detect and analyse ambient noise levels in real time. By utilizing UDP (User Datagram Protocol) for client-server communication, the system ensures low-latency data transmission, making it ideal for environments where immediate noise alerts are critical. Achievements of this project is that it developed a functional real-time noise monitoring system capable of detecting sound levels and triggering alerts when a certain threshold is exceeded. Demonstrated the practical application of IoT and networking concepts by integrating hardware (Raspberry Pi, microphone) with Python-based UDP communication. Offers a practical answer for noise-sensitive settings like homes, businesses, and hospitals. It was especially helpful to new parents by warning them of baby cries. Since it was the first time using the technology, it was difficult to get Pi to operate. Additionally, it was necessary to adjust the microphone sensitivity to ensure correct dB readings and to ensure that the equipment would not be damaged or fried. The system can be expanded in the future to alert several users at once. Developing an app to receive alerts instead of needing a command line would be one of the additional improvements.

7. References

Mydlarz, C., Salamon, J., & Bello, J. P. (2016). The implementation of low-cost urban acoustic monitoring devices. *Applied Acoustics*, 117, 207–218.
<https://doi.org/10.1016/j.apacoust.2016.06.010>

Raspberry Pi. (2024). *Raspberry Pi hardware - Raspberry Pi Documentation*. Raspberrypi.com.
<https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#gpio>

The Pi Hut. (2018, October 10). *An introduction to Raspberry Pi GPIO*. The Pi Hut.
<https://thepihut.com/blogs/raspberry-pi-tutorials/an-introduction-to-raspberry-pi-gpio>

8. Group member contributions

Abstract:

Skye	Jones
Minenhle	Khuzwayo
Sibusiswe	Zulu
Makhanani	Mlambo
Sixolile	Buso

Introduction:

Skye	Jones
Minenhle	Khuzwayo
Sibusiswe	Zulu
Makhanani	Mlambo
Sixolile	Buso

Literature Review:

Skye	Jones
Minenhle	Khuzwayo
Sibusiswe	Zulu
Makhanani	Mlambo
Sixolile	Buso

Hardware and Design:

Patrick	Habiyaremye
Josh	Cloete
Uwais	Cornelius
Lavhelani	Mano

Technical implementation:

Patrick	Habiyaremye
Josh	Cloete
Uwais	Cornelius
Lavhelani	Mano

Conclusion:

Lilitha	Mtsolongo
Simamkele	Dladla

Python code:

Patrick	Habiyaremye
Josh	Cloete
Uwais	Cornelius
Lavhelani	Mano

Project demo:

Patrick	Habiyaremye
Josh	Cloete
Uwais	Cornelius
Lavhelani	Mano

Presentation:

Patrick	Habiyaremye
Skye	Jones

Josh	Cloete
Uwais	Cornelius
Minenhle	Khuzwayo
Sibusiswe	Zulu
Makhanani	Mlambo
Lilitha	Mtsolongo
Sixolile	Buso
Simamkele	Dladla
Lavhelani	Mano