Tab 1

**Project 5: Wireless Buzzer Activation System**

**Presented by Group 5**

**Due Date: Apr 17, 2025**

**Module: Computer Science 311**

**Lecturer: Dr A. Henny**

**Group 5 members:**

Modupi Dlamini (4154846)

Ryan Henry(4388306)

Kaamiel Isaacs(4129581)

Khanya Khosana(4353124)

Jessie Amankwah(4351431)

Bennedict Mkhonto(4226279)

Phathutshedzo Munyai(4363258)

Naeem Peters(4200568)

Jayden-Lee Van Rooyen(4069341)

Bunono Xashimba(4383824)

**Table of Contents**

[Abstract](#_digan668vq2j)………………………………………3

[Introduction](#_vhkymz26t5ya)…………………………………..4

[Literature Review…](#_7vijrrvpnol5)…………………………5-7

[Hardware and Design](#_c7fz667350s0)………………………8-10

[Technical Implementation](#_bck6l3jgx36k)………………….11-14

[Conclusion](#_iouzzvq3gezn)…………………………………..15

[References](#_ipy67v8elevp)………………………………….16-17

## **Abstract**

This project presents the design and implementation of a wireless buzzer system, capable of remote activation via Wi-Fi network using a Raspberry Pi. The system can be employed in security, industrial monitoring, and emergency notification purposes, where immediate response is critical. Compared to traditional wired systems, this wireless solution provides simplified installation, greater flexibility, and cost-effective installation.

The system implements UDP (User Datagram Protocol) for network communication, favouring transmission speed over reliability. TCP guarantees complete data delivery, however, UDP's faster transmission better serves the requirement for instant alerts, where even the slightest delays outweigh concerns about occasional packet loss.

To operate the buzzer safely, the hardware configuration utilises a Raspberry Pi connected to a breadboard with fundamental components such as a transistor, a resistor, and jumper wires. This project validates that a low-overhead, UDP-based approach effectively serves wireless alert systems where speed is the primary requirement, while maintaining affordability and ease of setup for both residential and industrial applications.

## **Introduction**

In recent times, security and alert systems have become essential in both residential and industrial settings. Instant notifications during emergencies, like intrusions, equipment failures, or environmental hazards, can prevent serious damage and save lives. As our environments become more connected and automated, the demand for reliable, fast-response alert systems continue to grow.

This report presents a wireless buzzer activation system, designed to enable remote control of a passive buzzer via a Raspberry Pi-based server and a client device (e.g. laptop or smartphone) over a Wi-Fi network. By delivering real-time auditory alerts, the system is well-suited for applications that require immediate notifications, such as security alarms, industrial process monitoring, and smart home automation.

Wireless buzzer systems offer notable advantages over traditional wired setups, including enhanced security, simplified installation, and cost-effectiveness. Real-world applications include:

· Security and intrusion detection: Activating alarms to signal unauthorised access.

· Emergency detection: Issuing alerts for fire hazards or water leaks in industrial environments.

· Industrial automation: Generating auditory signals to indicate equipment malfunctions or process completions.

## **Literature Review**

**The importance of a wireless buzzer activation system**

In this paper, Group 5 introduces an advanced wireless Buzzer Activation System, a remote controlled sensor network designed to enhance alarm performances by activating or deactivating a passive buzzer through wireless communication. The system prioritizes flexibility, security and cost-effectiveness, making it adaptable to environments such as homes, offices and parks. Its convenience is that real-time monitoring and controlling are accessible via WI-FI connected devices. Reduced reliance on wires and labour costs makes it cost effective. Users can remotely manage alarms and integrate sensors for intrusion detection, fire, water hazards or environmental monitoring. Removing wired infrastructure makes the installation simple and enables mobility for rentals or temporary setups. While battery backup ensures reliability across varying spatial dimensions. Together, all these features address the importance of wireless buzzer activation systems in modern applications.

**What has been done before**

In recent research, previous projects focused on home automation, such as doorbell systems and home security cameras, have highlighted the potential of integrating remote-controlled features and passive buzzers. These systems provide foundational insights into how Group 5 can approach the project. For instance, Group 5’s wireless buzzer activation system utilizes user interaction similar to a doorbell button, which detects two specific states: whether the button has been pressed or not. By building on this concept, the team can enhance the project to better align with emerging trends and technologies in home security and automation. This literature review will explore these developments and assess how further innovations can be made in the design.

**How does our project differ**

**TCP vs UDP**

Transmission Control Protocol (TCP) is a standard internet protocol that ensures reliable, connection- oriented data transmission between devices, prioritizing data integrity through error-checking and retransmission. In contrast, User Datagram Protocol(UDP) is a connectionless protocol optimized for speed and low latency, making it ideal for time-sensitive applications like gaming or live streaming. While TCP guarantees complete and ordered delivery of data, UDP sacrifices reliability for faster transmission, which is critical in scenarios where delays are not accepted.[8]

For the wireless buzzer activation system, UDP is the preferred choice. Its minimal latency ensures rapid communication, enabling real-time alerts to clients during emergencies. Though packet loss may occasionally occur, the priority is on ensuring that clients receive immediate notification. [8]

**Active vs Passive buzzer**

Other existing projects utilize an active buzzer, while group 5 has decided to use a passive buzzer for the current project. An active buzzer only produces a single tone compared to a passive buzzer that produces a variety of tones. A passive buzzer is versatile, which gives the ability to implement a different tone for every possible scenario to code for, thus giving the team flexibility and control over an active buzzer rigidity.[3]

**Raspberry Pi**

Raspberry Pi is a small, powerful, cheap, hackable, and education-oriented computer board introduced in 2012.[4]

Raspberry Pi is very flexible and can be used in almost any project. The advantage of using the board is that it is a small, independent computer that can run Linux, it has a large memory that can be expanded upon. The ability to customize the board allows the addition of Wifi and Bluetooth adapters, and the expansion and communication with network devices over a LAN adapter are possible. It has a processor that supports a wide variety of instructions and has support for USB 2.0, which gives an expansion of the number of peripherals. Raspberry Pi has operable speeds up to 700MHz to 1000MHz.[4]

Despite all the listed advantages, Raspberry Pi does have limitations, such as lacking a real-time clock with a backup battery. It always boots from the SD card, which means that as much as an external storage device can be used, it cannot boot the Raspberry Pi. It does not support Bluetooth or Wi-Fi out of the box. Most Linux distributions are picky about hardware. Depending on how busy the Raspberry Pi is, power consumption can be a concern.[4]

Overall, Raspberry Pi is a reliable device suited for the project in terms of accessibility and cost-effectiveness.[4]

**Python-based Implementation**

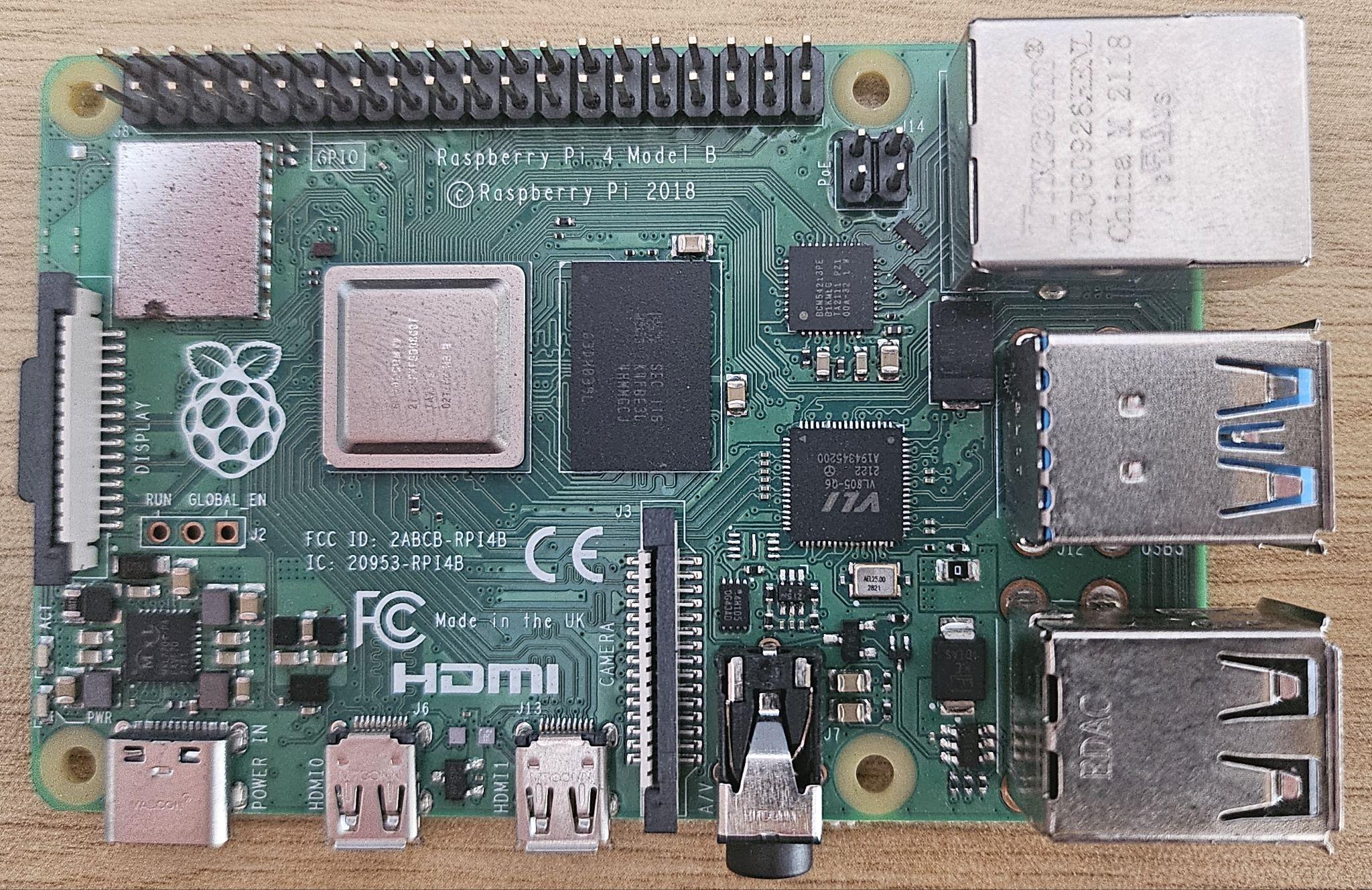
Group 5's project explored various code implementations that facilitated the development of a Python-based solution. A particularly noteworthy implementation features a buzzer connected to a breadboard, which interfaces with a Raspberry Pi. This setup involves initializing the buzzer and defining its functions, such as turning it on and off, while also establishing a timer for the duration of these actions. Additionally, network-based communication is utilized through sockets, Flask, or MQTT for client-server code models. The advantage of using Python lies in its ability to enhance customization, enable real-time data handling, and support web dashboard integration.

The Python code employs various libraries to send activation and deactivation signals to the buzzer, ensuring reliable operation and tracking the timing of the sound production. This approach builds on insights from previous works, contributing to a robust framework that aligns with Group 5's objectives for monitoring and sound signaling[12].

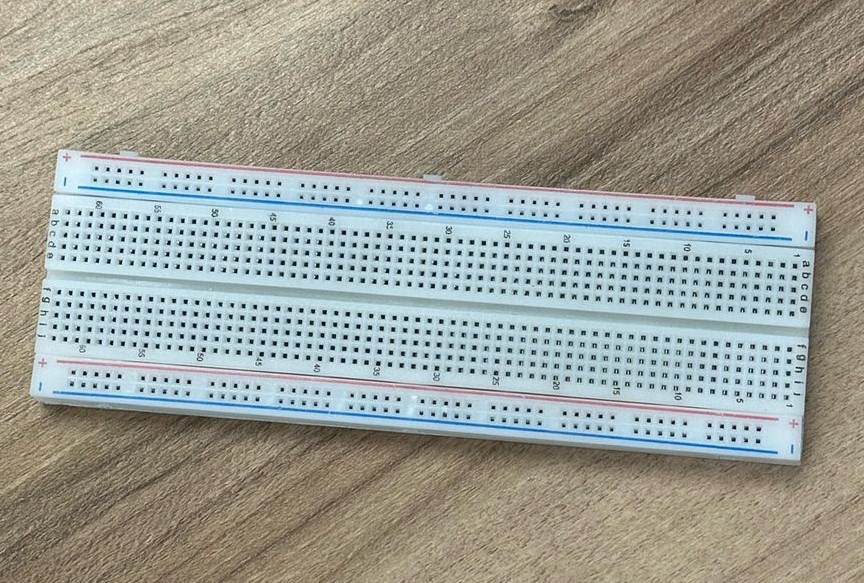
## **Hardware and Design**

The setup consists of a few components: a Raspberry Pi, three male-to-female jumper cables, one male-to-male jumper cable, a 1K resistor, a transistor, and a breadboard.

**Hardware components**

**Raspberry Pi**

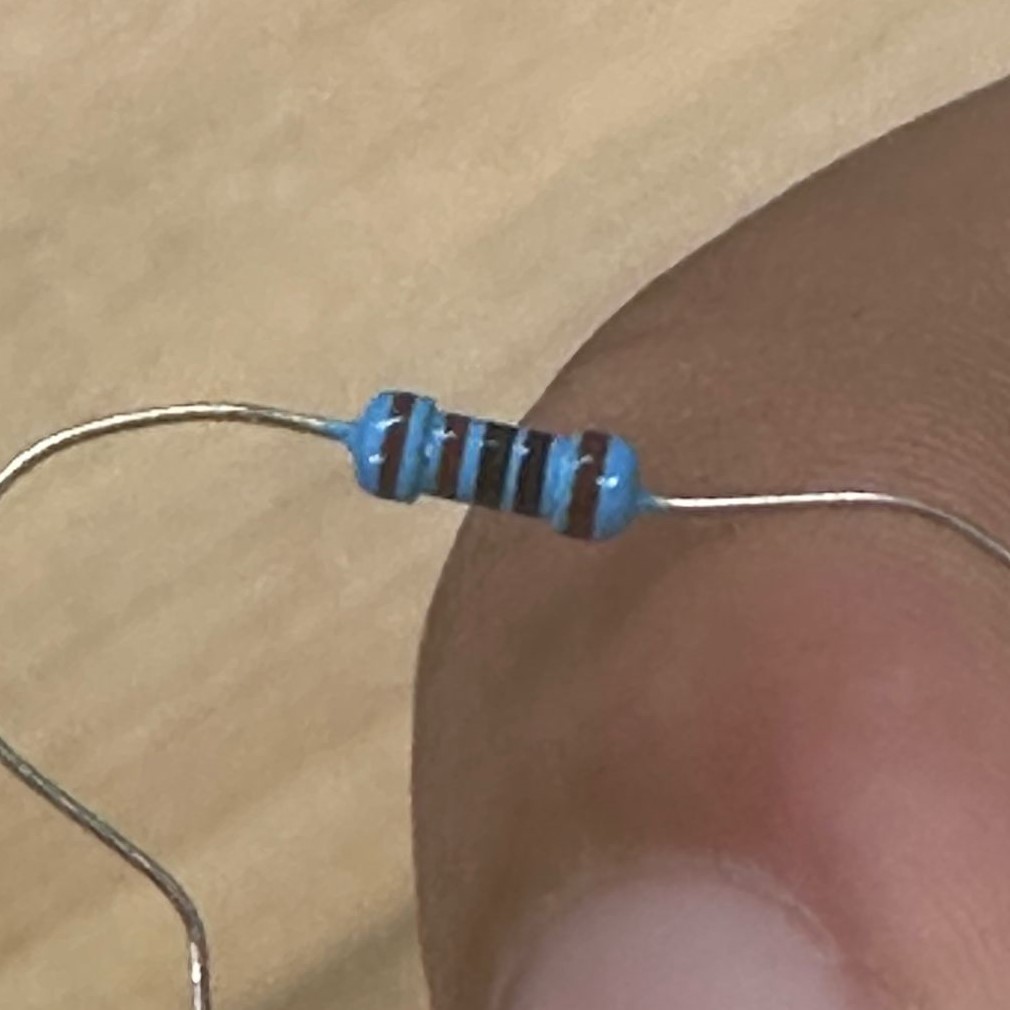
* Serves as the central hub, hosting the server and managing all interactions between hardware and software
* There is a 40-pin GPIO header for sending signals, supplying power, and grounds any circuits.**[1]**
* The GPIO pins allow external components to interact with the device as well as one another
* This setup uses GPIO pins 1 (3.3V power), 16 (GPIO 23), and 20 (Ground)

**Breadboard**

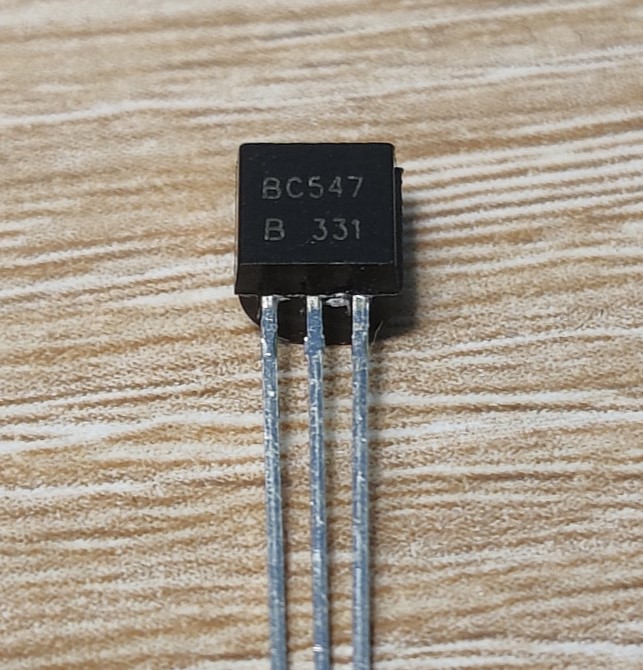
* Acts as an interface that allows us to connect multiple external components without soldering.**[2]**
* Used to connect the Raspberry Pi with the other components such as the buzzer and transistor
* Three jumper cables(male to female) are used to transmit power, ground, and GPIO signals. There is also have one small jumper cable(male to male) connecting the transistor to the buzzer

**Buzzer (Passive)**

* The buzzer emits a sound based on the frequency of the PWM (Pulse Width Modulation) signals from the Raspberry Pi.**[3]**
* Allows for various tones but requires more complex code when compared to an active buzzer
* Due to the complexities, it requires a power source as well as a control signal to play a sound/tone.

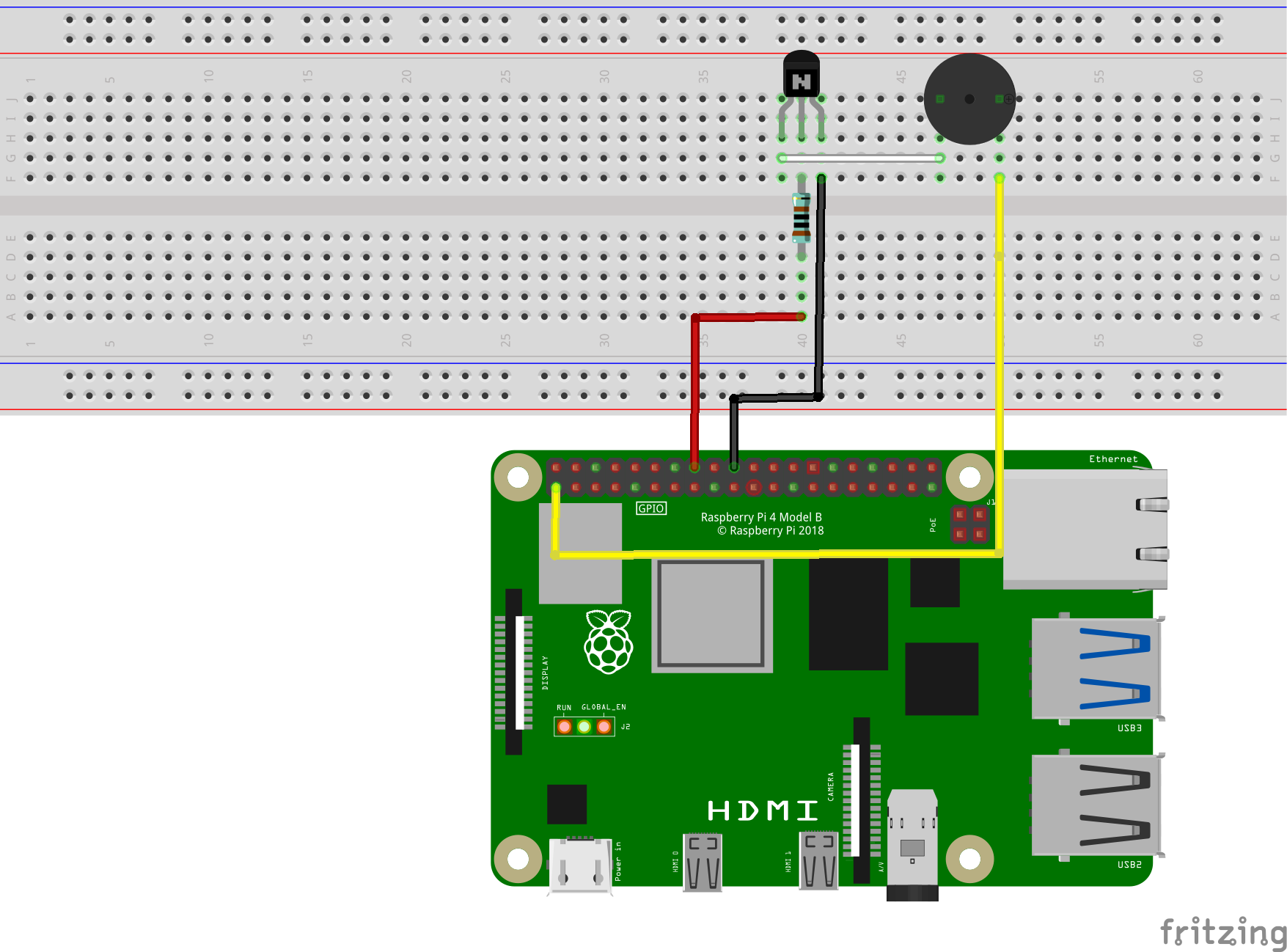
**1KΩ Resistor**

* Limits the amount of current flowing from the GPIO pin, it will limit it to Voltage/Resistance = Amps. In this case it will be roughly 3 mA
* If the current goes unregulated it can damage the GPIO pin on the Raspberry Pi or damage the entire Raspberry Pi

**Transistor**

* Used as a switch in this case to control current flow to the buzzer, breaking or completing the circuit.**[4]**
* The transistor has three legs. It will depend on the model, but we are using an NPN transistor, specifically a BC547, where we have a Collector (Left Leg), Base (Middle Leg), and Emitter (Right Leg).**[5]**
* The base is connected to the GPIO pin (via resistor), the collector to the buzzer’s negative terminal, and the emitter to the ground pin.

**How it works**

****

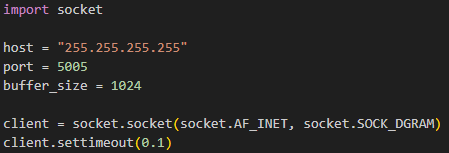
We have a main circuit which goes from the 3.3V power pin to the buzzer to the transistor and to the ground pin. When the transistor is on, it completes the circuit; this will allow current to flow from the 3.3V power pin through the buzzer, transistor, and finally to the ground pin which produces a tone. When the transistor is off, it breaks the circuit, which stops the current flow, and the buzzer turns off. The Raspberry Pi sends a PWM signal from a GPIO pin to the base of the NPN transistor through a 1KΩ resistor. PWN switches the GPIO pin on and off rapidly. This signal is sent to the transistor base, which in turn turns the transistor on and off rapidly. This also means that the current to the buzzer is turned off and on rapidly to the same rate.**[6]** This pulsing of the current causes the buzzer to vibrate, producing a tone. The frequency of the PWM signal determines the frequency of the tone; for example, if a 1000 Hz tone is needed, the transistor will switch on and off 1000 times per second, producing a 1000 Hz tone.

**Safety Considerations**

A key note here is that the GPIO pins are designed to control something and not directly power it, so if we have a major component like a motor and we directly connect it to the GPIO pin, it will burn out the pin or even damage the entire Raspberry Pi. This circuit design prioritizes protection of the Raspberry Pi's GPIO pins by limiting current through a 1KΩ resistor and using a transistor to switch the circuit on and off. This way we avoid directly powering the components from the GPIO pins and ensure the safety of both the components as well as the Raspberry Pi.

## **Technical Implementation**

**CLIENT CODE EXPLANATION**



The code for the client side of the wireless passive buzzer activation system by GROUP 5 begins by importing the socket package and defining the host IP address as well as the port number in order to send commands to the server’s IP address/Port. A variable called “buffer\_size” is created in order to store the maximum amount of bytes that the socket will try to receive in a single operation. A UDP socket is then created using “SOCK\_DGRAM”, which is used to create the UDP socket, as well as “AF\_INET” and a timeout for the socket which is set at 0.1 seconds. This is a very short timeout but is also quite suitable for a LAN environment which was used when testing the client code.

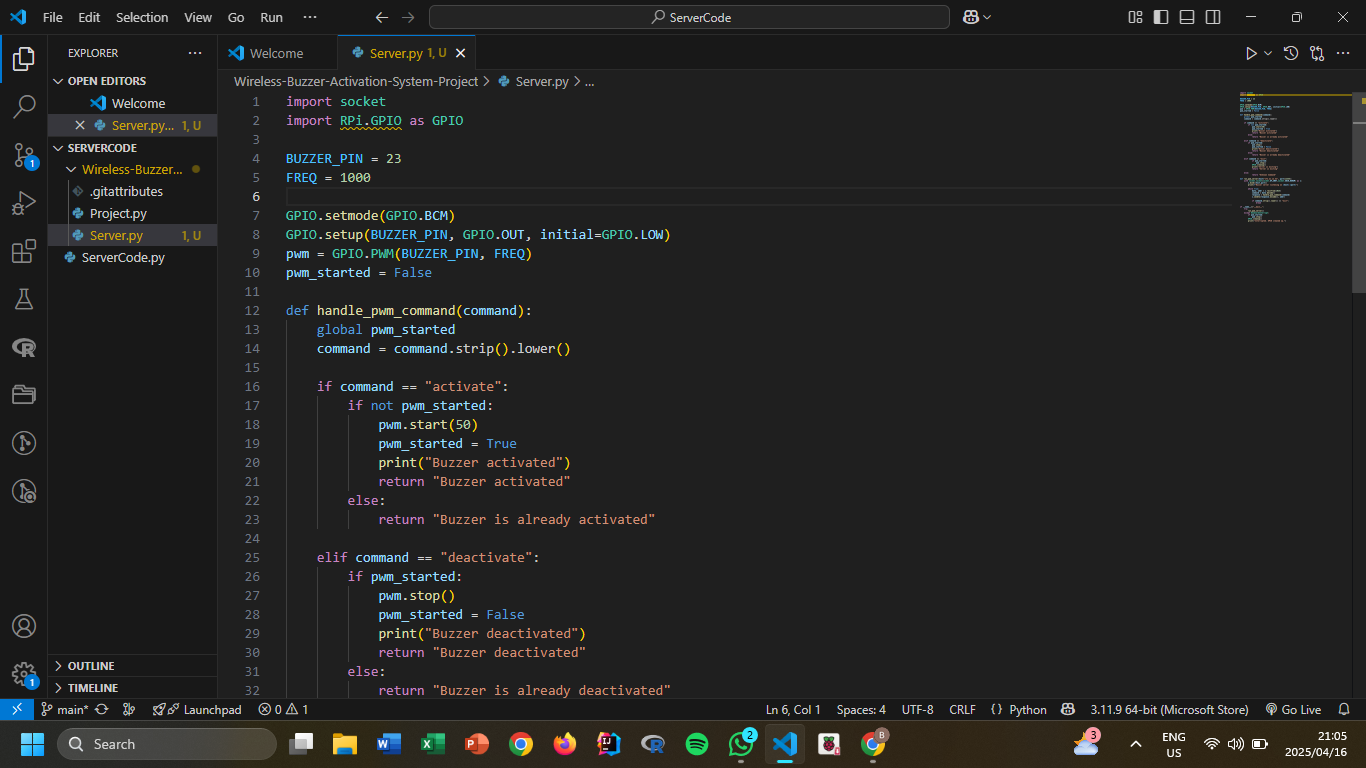


The client code will prompt the user to enter either “Activate”, which will send the command to the server telling it to activate the buzzer, ”Deactivate”, which will send the command to the server telling it to deactivate the buzzer, or ”Exit”, which will stop the client loop and close the socket. Based on the command entered, the client will make use of if and elif statements in order to handle the logic of the command and send it to the server. The client makes use of a try-except block which tries to send the command to the server by encoding it and receiving a response from the server. It catches the exception where the socket times out and provides an error message letting the user and server know that the socket has timed out, as well as a general exception catch for any other exceptions that occur.

The client code was tested on a local network on the UWC Campus in Lab E2 in the CAMS building with its respective server code.

**Server code explained**

This python code is designed to control a buzzer connected to Raspberry Pi using GPIO pins. It uses a UDP socket server that listens to for specific text-based based command (activate, deactivate and exit) from a client to control the buzzer’s PWM (Pulse Width Modulation) signal. The code uses the RPi.GPIO library for hardware control and the socket module for network communication.

The buzzer is connected to GPIO pin 23.

* **FREQ = 1000**: Sets the PWM frequency to 1 kHz.
* **GPIO.setmode(GPIO.BCM)**: Uses the Broadcom SOC channel numbering.
* **GPIO.setup(...)**: Configures the buzzer pin as an output.
* **pwm = GPIO.PWM(...)**: Initializes PWM on the specified pin and frequency.
* **pwm\_started**: A flag used to track whether PWM is running.

A computer screen shot of a program code

AI-generated content may be incorrect.

It is a method that handles incoming string commands and maps them to buzzer control actions.

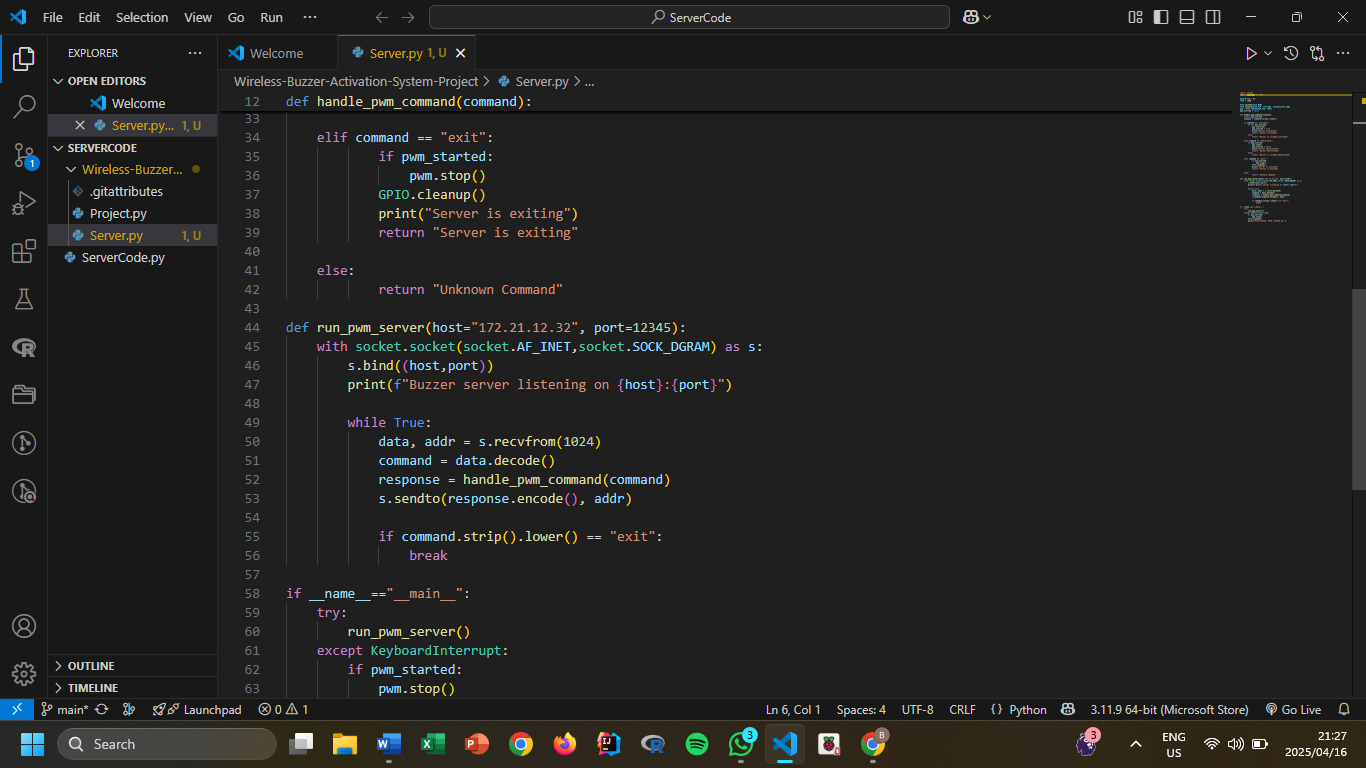
Supported commands/ commands that should be entered by the client:

"activate": Starts the PWM signal (50% duty cycle).

"deactivate": Stops the PWM signal.

"exit": Stops PWM (if running), cleans up GPIO and exits the server.

Any other input returns "Unknown Command".



The method creates a UDP server bound to a specified host and port and binds the socket to the provided IP and port.

It also waits to receive data via the method recvfrom().

Processes commands using the method handle\_pwm\_command().

Sends back a response to the client.

Break the loop and clean up if "exit" command is received from the client

UDP is chosen for its simplicity and low overhead, but it's not connection-oriented or reliable like TCP.

The method:

A computer screen shot of a black screen

AI-generated content may be incorrect.

Ensures the server runs only when the script is executed directly.

## **Conclusion**

The Wireless Buzzer Activation System project has effectively shown that controlling a passive buzzer remotely via a wireless network using a Raspberry Pi as a server is both possible and efficient[10]. The system's use of UDP-based socket connection guarantees rapid communication and low latency between the client and server[11], which makes it appropriate for a range of uses, including industrial automation, remote monitoring, and home security.

Achievements

● Rapid Communication: Commands to activate, deactivate, and exit are speedily communicated with minimal latency using the UDP protocol.

● User-Friendly Interface: The system is accessible and simple to use because of the client interface, which is implemented on a laptop or mobile device and offers an intuitive approach to regulating the buzzer.

● Versatile Application: The system's architecture makes it simple to adapt to a variety of situations, such as remote control applications, industrial monitoring, and home security systems.

Challenges faced

Network Latency: It was difficult to guarantee minimal latency in command transmission, particularly in settings with a lot of network traffic.

Hardware Restrictions: In order to achieve immediate control, the performance of the GPIO pin and the passive buzzer's reaction time were crucial.

Security Issues: Protecting the communication channel from unwanted access and guaranteeing data integrity was a key factor.

Possible improvements

Improved Security: Putting authentication and encryption in place to protect client-server communication.

Other functionality: Upgrading the user interface to give more feedback, connecting with other sensors, and adding additional functionality like various buzzer tones.

Scalability: Improving the system to accommodate more devices and clients, which makes it better suited for larger-scale applications.

## **Reference**

**Literature review references**

[1]*Raspberry Pi Foundation* - Official documentation on GPIO programming[online] Available:<https://www.raspberrypi.com/documentation/computers/os.html#gpio-and-the-40-pin-header> [Accessed Apr 11, 2025]

[2]S. Monk, *Programming the Raspberry Pi: Getting Started with Python*, 2nd ed. New York: McGraw-Hill, 2016.

[3] "Passive vs active buzzers," Electronics-Tutorials.ws, Jul. 2022. [Online]. Available:<https://www.electronics-tutorials.ws/> [accessed Apr 14, 2025]

[4] ResearchGate, “Raspberry Pi as a Wireless sensor node: Performance and constraints”,*37th International Convention on Information and Communication Technology, Electronics and Microelectronics(MIPRO),*[Online]. Available: <https://www.researchgate.net/publication/269291637_Raspberry_Pi_as_a_Wireless_Sensor_node_Performances_and_constraints> [Accessed: Apr. 1, 2025]

[5] Raspberry Pi Foundation. (n.d.). *Physical computing with Raspberry Pi*. <https://projects.raspberrypi.org/en/projects/physical-computing/0>

[6] Chen, W. (2017). *Development of the Raspberry Pi-based home automation system*. ProQuest Dissertations Publishing. <https://www.proquest.com/openview/41dfc6afa1ebbab7f41498a50c3191c1/1?cbl=2042732&pq-origsite=gscholar>

[7] Core Electronics. (n.d.). *PiicoDev buzzer module Raspberry Pi guide*. <https://core-electronics.com.au/guides/piicodev-buzzer-module-raspberry-pi-guide/>

[8], AL-Dhief1, F.T., Sabri, N. (2018) *Performance comparison between TCP and UDP protocols in different simulation scenarios. ResearchGate.*

Available:<https://www.researchgate.net/publication/329944111_Performance_comparison_between_TCP_and_udp_protocols_in_different_simulation_scenarios>

[9] Tejas, P. (2023) *IoT-based ultrasonic sensors and buzzer systems in home security.* ResearchGate. Available: <https://www.researchgate.net/publication/378841643_IoT-based_ultrasonic_sensors_and_buzzer_systems_in_home_security>

[10] E. Upton and G. Halfacree, *Raspberry Pi User Guide*, 3rd ed. Wiley, 2014.

[11] M. J. Donahoo and K. L. Calvert, *TCP/IP Sockets in C: Practical Guide for Programmers*, 2nd ed. Morgan Kaufmann, 2009.

[12] Electrocredible. (2020 May 13). Wireless buzzer system using Raspberry Pi and python [video]. YouTube. http:// [www.youtube](http://www.youtube).com/watch?v=gwS5SrN-AhE

**Hardware and Design References  
[1]** Raspberry Pi Foundation, "Physical computing with Python," *Raspberry Pi Projects*, [Online]. Available: <https://projects.raspberrypi.org/en/projects/physical-computing/1>. [Accessed: Apr. 10, 2025].

**[2]** SparkFun Electronics, "How to Use a Breadboard," *SparkFun Learn*, [Online]. Available:<https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard#why-use-breadboards>. [Accessed: Apr. 10, 2025].

**[3]** MadMachine, "Buzzer," *MadMachine Documentation*, [Online]. Available:<https://docs.madmachine.io/learn/peripherals/buzzer#buzzer>. [Accessed: Apr. 10, 2025].

**[4]** Ø. N. Holmeide, "Transistor as a switch," *Build Electronic Circuits*, [Online]. Available:<https://www.build-electronic-circuits.com/transistor-as-a-switch/>. [Accessed: Apr. 11, 2025].

**[5]** Components101, "BC547 Transistor Pinout, Specs, Equivalents & Datasheet," *Components101*, Mar. 30, 2021. [Online]. Available:<https://components101.com/transistors/bc547-transistor-pinout-datasheet>. [Accessed: Apr. 11, 2025].

**[6]** MadMachine, "What is PWM?" *MadMachine Documentation*, [Online]. Available:<https://docs.madmachine.io/learn/peripherals/buzzer#what-is-pwm>. [Accessed: Apr. 14, 2025].