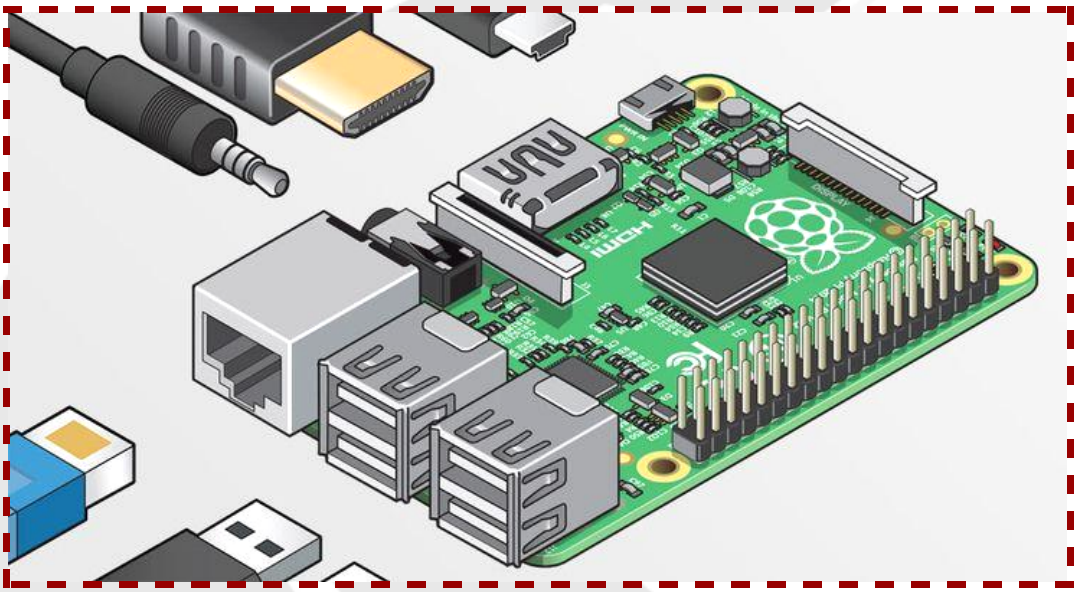
Client-controlled Pedestrian Priority Traffic Light System

CSC311 Networking Practical 2

Assignment Report



Deogracias Mufula, 4203232

Rolande Solomons, 4269355

Gugulethu Matsane, 4376487

Peace Khutso Molimo, 4352859

Eyethu Njemla, 4133541

Craig Antonio, 4323965

Jordan Eckleton, 4127304

Siyamthanda Tshetshe, 4327595

Lesedi Lebopa, 4276507

Sphesihle Ngubane, 4202730

Table of Contents

Below are links to the various headings:

Abstract

The **Client-Controlled Traffic Light System with Pedestrian Priority** addresses the limitations of traditional traffic light systems, which often fail to adapt to real-time pedestrian crossing requests. This project proposes a Client-Server application based on a Raspberry Pi Traffic Light Control System that would improve pedestrian safety and traffic efficiency. The system uses three LEDs (red, yellow, and green) to simulate traffic signals, where pedestrians can request crossings via a client interface on a laptop or PC. The primary goal is to prioritize pedestrian safety, particularly for disabled individuals. By constantly changing traffic light cycles based on pedestrian demand, the prioritization of pedestrians can be ensured. This approach reduces wait time by initiating a walk cycle immediately after a request is received. Communication between the client and the Raspberry Pi is facilitated using the Transmission Control Protocol (TCP), which was chosen for its flow control and reliable data transmission advantages, as well as its suitability for real-time control in small-scale applications. The system's goal is to improve urban transportation, relieve congestion, and promote equality using affordable and accessible technology. The anticipated goals include increased pedestrian safety, lower wait times, and a cost-effective solution for smart city infrastructure. This initiative supports sustainable urban development goals by providing a low-cost solution for enhancing traffic management and pedestrian accessibility in urban environments.

Keywords: Raspberry Pi, Traffic Light System, Pedestrian Priority, TCP Communication, Client-Server.

Introduction

Traffic light systems are a critical component of modern urban infrastructure, ensuring the safe and efficient movement of vehicles and pedestrians. Traditional traffic light systems operate on fixed timers or pre-programmed cycles, which may not always accommodate real-time demands, such as pedestrian crossing requests. With the advent of Internet of Things (IoT) technologies and low-cost computing platforms like the Raspberry Pi, it is now possible to create more dynamic and responsive traffic management systems. This project, titled "Client-Controlled Traffic Light System with Pedestrian Priority," aims to simulate a traffic light system that can be controlled remotely via a client interface, with the ability to prioritize pedestrian crossing requests.

Raspberry Pi is a single-board, small and inexpensive computer developed by Raspberry Foundation developed in the United Kingdom. The system leverages the Raspberry Pi as a central controller, managing the traffic light sequence while listening for incoming pedestrian crossing requests from a client device. A client-server application is initialized in Python which manages the network requests, and the LED control command. The Raspberry Pi acts as the server and the laptop or PC acts as the client. The Raspberry Pi controls three LEDs (Red, Yellow, Green) representing traffic lights, while the client interface (on the laptop or PC) allows pedestrians to request crossings.

The importance of this project in real-world application is to assist people, especially those who are disabled, to cross the road safely. It becomes particularly relevant in scenarios near schools, hospitals, or busy urban intersections. Giving priority to pedestrians at crossings ensures they have dedicated time to cross safely. This project not only demonstrates the integration of hardware (LEDs, resistors) and software (Python scripts) but also explores the use of network communication protocols (UDP or TCP) to enable real-time interaction between the client and server.

It is impossible to overestimate the significance of pedestrian priority in contemporary traffic systems since it guarantees accessibility for everyone, including those with disabilities. Other pros include the possibility of lowering accident rates, and encouraging safer, more walkable communities. The system makes roads safer and traffic more efficient by allowing pedestrians to regulate crossings and integrating smart technology into urban infrastructure.

Literature Review

Why is this project relevant? Explain the importance of the project in modern applications.

The findings derived from this project are crucial because they address urgent real-world issues in today's cities, such as traffic congestion, pedestrian safety, and the need for better, more equal urban infrastructure. As more people choose to walk or bike to work every day, the traffic systems need to adapt to accommodate this growing trend. Unfortunately, many traditional layouts still prioritize cars over pedestrians, which can jeopardize safety, particularly for individuals with disabilities. Rapid Urbanization causes traffic congestion, higher accident rates, and worse standards of living. Efficient traffic management systems are vital, with pedestrian safety being the primary priority.

Smart cities, which use IoT devices, are gaining popularity. Modern urban design focuses on inclusivity by making infrastructure accessible to everyone, including the disabled. Promoting walking and cycling as alternatives to driving lowers carbon emissions while increasing health. AI and sensor improvements allow for sophisticated traffic control solutions, but many are expensive, complex, and lack user-friendly features. Fixed scheduling and automated detection may be inefficient, especially in smaller cities or developing areas. The United Nations' Sustainable Development Goals prioritize sustainable cities, decreased inequities, pedestrian safety, and effective traffic management. As cities evolve, transportation infrastructure must adapt to new challenges, such as increased pedestrian traffic and self-driving cars.

Similar to the above-mentioned smart cities, the client-controlled traffic light system developed in this project optimizes traffic flow by altering signal timings in real time to accommodate pedestrian and vehicle traffic. This lowers stops, reduces congestion, and increases traffic efficiency. It addresses pedestrian safety concerns, particularly among disadvantaged groups. It aligns with the goals of smart cities by utilizing Raspberry Pi and Python technology. It enables pedestrians to request crosswalks, fostering inclusion and barrier-free cities. It lowers wait times, encourages walking, and reduces carbon emissions.

What has been done before? Review existing projects or research papers related to your project.

Traffic congestion is an escalating issue in cities worldwide, leading to wasted time, increased pollution, and heightened accident risks. Traditional traffic lights, which operate on fixed timers, are increasingly inadequate for managing modern urban traffic demands. To understand the current state of smart traffic light technology, this section examines prior research, real-world implementations, and key findings from academic and industry sources. By analyzing existing projects and studies, one is

able to identify trends, successes, and unresolved challenges in adaptive traffic signal systems. Smart traffic lights offer a dynamic alternative by modifying the timing of signals according to live traffic conditions. These systems maximize the flow of traffic, reduce delays, and enhance safety, making them a promising solution for future urban mobility.

How Smart Traffic Lights Work

Smart traffic lights rely on a network of sensors, cameras, and AI to monitor and respond to real-time conditions. Radar and LiDAR detect vehicle speed and position, while AI-powered cameras analyze traffic density, pedestrian movement, and red-light violations. Environmental sensors further refine operations by adjusting signals for adverse weather. Data processing occurs at two levels: edge computing handles immediate adjustments at individual intersections, while cloud-based AI analyzes city-wide patterns to synchronize signals across multiple junctions. Adaptive Traffic Signal Control (ATSC) algorithms enable predictive adjustments, such as prioritizing emergency vehicles, extending green lights during peak hours, and creating coordinated "green waves" to improve traffic flow.

Advantages Over Traditional Systems

The benefits of smart traffic lights are substantial. Research from institutions like Carnegie Mellon University indicates they can reduce travel times by 25%, cut waiting periods at signals by 40%, and lower emissions by 20% due to smoother traffic flow. Safety improvements are another key advantage, with red-light violation detection reducing intersection collisions. Additionally, these systems integrate seamlessly with smart city ecosystems, enabling vehicle-to-infrastructure (V2X) communication for autonomous vehicles and prioritizing public transport to enhance efficiency.

Challenges and Limitations

Despite their potential, smart traffic lights face significant hurdles. High implementation costs—exemplified by projects like Manchester’s £30M upgrade—pose a barrier for many cities still reliant on outdated infrastructure. Technical challenges include false detections, such as failing to recognize cyclists or pedestrians, while human factors like driver non-compliance persist, as seen in New York City’s surge in red-light running incidents. Privacy concerns also arise from constant surveillance via traffic cameras, and cybersecurity risks emerge if hackers gain control of signal timings.

Real-World Success Stories

Several cities have demonstrated the effectiveness of smart traffic systems. London’s Sitraffic FUSION, developed by Siemens, uses real-time vehicle data to optimize traffic flow, reducing delays for buses and emergency responders. Meanwhile, Manchester’s AI-driven trial employs 5G-connected sensors to

dynamically adjust signals, alleviating congestion. These examples highlight the tangible benefits of adopting intelligent traffic management solutions.

Another example is one found close to home. Cape Town has become the first city in South Africa to install thermal sensors at four pedestrian crossings in Blaauwberg. The thermal sensors have been installed in busy routes, which were along Marine Drive opposite Milky Lane, along Otto du Plessis Drive opposite Seal Road, along Otto du Plessis Drive at Shell Road and Along Otto du Plessis Drive close to Hill Road, opposite Doodles. This system reduces unnecessary vehicle stops and minimizes vandalism associated with traditional push-button mechanisms.

The Future of Smart Traffic Lights

The evolution of smart traffic lights will likely focus on three key areas. First, expanded V2X communication will enable direct interaction between autonomous vehicles and traffic infrastructure, allowing predictive traffic management. Second, advancements in AI and machine learning will improve detection accuracy for pedestrians and cyclists while enabling self-learning algorithms that adapt to long-term traffic trends. Finally, policy and infrastructure development such as government incentives and standardized cybersecurity regulations—will be crucial for widespread adoption.

Smart traffic lights represent a significant leap forward in urban traffic management, offering solutions for congestion, safety, and environmental impact. While challenges like cost, technology limitations, and public acceptance remain, continued advancements in AI, connectivity, and policy frameworks could solidify their role in shaping the future of transportation. If these obstacles are addressed, smart traffic lights may well revolutionize urban mobility, making commutes faster, greener, and safer for all.

How does your project differ? Highlight any contributions or improvements over existing work.

The project leverages a Raspberry Pi as the server and a laptop/PC as the client, connected via the UDP protocol, enabling real-time communication between pedestrians and the traffic light controller. Upon receiving the request, the Raspberry Pi dynamically adjusts the traffic light cycle to include a pedestrian walk phase without unnecessary delay. This adaptive approach represents a significant improvement over existing systems, enhancing efficiency and responsiveness to pedestrian needs. The enhancement ensures that pedestrians experience minimal waiting time, improving overall user experience and safety.

Additionally, this design is future-proof, with potential enhancements such as integrating motion sensors for pedestrian detection or scaling the system to handle multiple crossing points simultaneously. Which makes this solution more responsive, adaptable, and scalable compared to standalone or fixed-timer systems.

Unlike many traditional traffic light systems, the primary distinction of the project lies in how pedestrian crossing requests are handled. While most systems treat pedestrian requests as a lower priority or integrate them into a fixed cycle, the approach places pedestrians at the forefront of the traffic control process.

The integration of client-controlled traffic light systems with pedestrian priority presents a promising avenue for improving urban mobility and safety. However, several challenges and research gaps persist in this domain, namely the following:

1. *Detection and Prioritization of Pedestrians*

Accurately detecting pedestrian presence and effectively prioritizing their movement remains a significant challenge. Traditional traffic systems often lack the capacity to assess the number of pedestrians waiting at crossings, leading to inefficient signal timings. Professor Vinayak Dixit from UNSW highlights that current technologies, such as loop detectors and cameras, primarily focus on vehicular traffic, often overlooking pedestrian dynamics. This oversight can result in suboptimal pedestrian experiences and potential safety risks. (Cecillia Duong, 2021)

2. *Integration of Adaptive Signal Control Systems*

While Adaptive Signal Control Systems (ASCS) aim to optimize traffic flow by modifying signal timings in response to current (real-time) conditions, their effectiveness in accommodating pedestrian movements is limited. Challenges include the complexity of accurately detecting pedestrian presence and movement, which is crucial for ensuring pedestrian safety and minimizing delays.

3. *Public Acceptance and Behavioral Adaptation*

The success of client-controlled traffic light systems with pedestrian priority depends on public acceptance and behavioural adaptation. Users must trust and understand the system's functionality to ensure effective interaction (Lee and Kim, 2023). Studies should focus on analysing user perception, compliance rates, and overall impacts on pedestrian and driver behaviour.

4. *Data Privacy and Ethical Considerations*

The integration of AI and sensor-based pedestrian detection technologies raises concerns about data privacy and ethical considerations. Ensuring these systems do not infringe on individual privacy rights while maintaining efficient traffic management is a critical research challenge (Smith and Johnso, 2023). Future studies must establish robust data protection frameworks and policies to address these concerns.

Key Technologies: Discuss Raspberry Pi, network protocols (UDP/TCP), sensors, and Python-based implementations.

Raspberry Pi

The Raspberry Pi has become a popular platform for developing embedded systems due to its low cost, versatility, and ease of use. Richardson and Wallace (2017) highlighted Raspberry Pi's ability to interface with various sensors and actuators, making it ideal for IoT applications. In the context of traffic light systems, the Raspberry Pi can serve as a central controller, managing both hardware components (e.g., LEDs) and network communication.

Several studies have demonstrated the use of Raspberry Pi in traffic management. For example, Singh et al. (2020) developed a Raspberry Pi-based traffic light system that uses image processing to detect vehicles and adjust traffic light timings. Similarly, Gupta et al. (2021) created a smart traffic light system that integrates weather data to optimize traffic flow. These studies underscore Raspberry Pi's potential as a cost-effective solution for intelligent traffic management.

Network protocols (UDP/TCP)

The use of network communication protocols, such as UDP and TCP, is a fundamental aspect of IoT systems. TCP (Transmission Control Protocol) provides reliable, connection-oriented communication, ensuring that data packets are delivered in the correct order and without errors. On the other hand, UDP (User Datagram Protocol) offers faster, connectionless communication, making it suitable for real-time applications where speed is more critical than reliability.

In the context of traffic light systems, Al-Fuqaha et al. (2015) discussed the use of TCP for reliable communication between traffic controllers and central management systems. While UDP offers lower latency and is sometimes considered for real-time applications like pedestrian crossing requests, this project prioritizes reliability and data integrity over minimal delay.

Communication between the client and Raspberry Pi leverages the Transmission Control Protocol (TCP) to ensure that all pedestrian requests are accurately transmitted and acknowledged. Although UDP can provide faster communication, TCP is chosen in this project to avoid the risk of lost or incorrectly sequenced requests, which is critical in safety-focused applications such as traffic signal control. The implementation focuses on maintaining a robust and reliable operation, making TCP the more suitable choice.

Sensors

The system can be further improved with additional sensors like thermal sensors, cameras, and LiDAR to detect pedestrians more effectively. Thermal sensors capture the infrared radiation emitted by objects, enabling them to identify pedestrians through their body heat. This approach remains effective in low-light environments and can detect individuals even when they are stationary, overcoming the constraints of conventional motion sensors. LiDAR sensors calculate distances by emitting laser pulses and measuring their reflections. They generate detailed 3D

maps of the surroundings, enabling reliable detection of pedestrians and obstacles. Although LiDAR provides high precision, factors such as cost and integration complexity must be considered. There are also Standard Cameras which offer visual validation and can be integrated with AI to analyze pedestrian behavior, such as identifying the intent to cross.

Although these sensors improve automation, factors like cost, environmental disruptions (e.g., fog affecting LiDAR), and processing demands should be taken into account. These technologies enhance decision-making and align with the trend of modernizing urban traffic systems through innovative sensor solutions. As much as this project relies on the user clicking the button for the request to be approved one can also use physical sensors to detect pedestrians who are about to cross the road but that will be difficult to implement .

Push-button sensors offer a simple and cost-effective solution. Instead of using a computer, pedestrians could press a real button at the crossing to send the request directly to the Raspberry Pi. The GPIO pins on the Raspberry Pi allow it to read signals from these sensors and trigger the pedestrian phase accordingly.

Python-based implementations

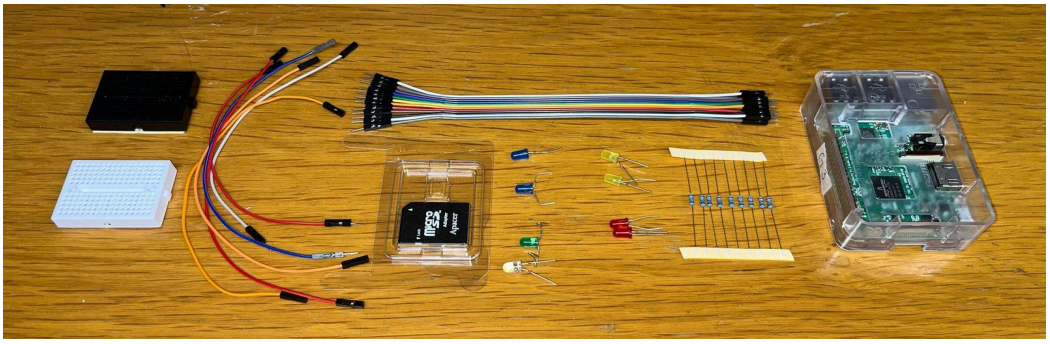
When it comes to Internet of Things (IoT), Raspberry Pi excels as a core component for various automation projects. Many of these systems are built using Python-based implementations, leveraging its flexibility and compatibility with Raspberry Pi hardware. For instance, Python scripts enable real-time communication between sensors and client devices in projects like the Real-Time Obstacle Distance Monitor, which uses an ultrasonic sensor and Python networking to send alerts when an object is too close. Similarly, the Client-Controlled Traffic Light System for this assignment is built using Python, to be able to send messages efficiently using network communication and hardware components. Namely,

- Network communication: Python's socket module helps send and receive messages between the laptop and Raspberry Pi.
- LED control: The RPi.GPIO will then make the Raspberry Pi to turn the traffic lights both on and off depending on whatever color is required .
- Handling multiple tasks: This is advantageous because it will help save time by using the threading module , which helps the Raspberry pi to listen for messages and control the traffic lights at the same time.

Additionally, Python-based frameworks like Flask can be used to develop web applications and APIs for controlling IoT devices remotely. These implementations showcase how Python is integral to building efficient, responsive, and network-driven IoT systems on Raspberry Pi.

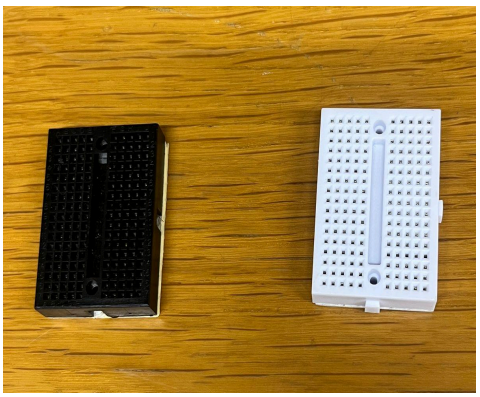
Hardware and Design

Overall view of all the hardware components used.



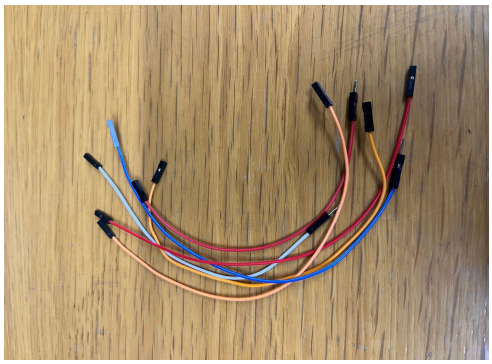
Detailed descriptions of each hardware component:

Breadboard



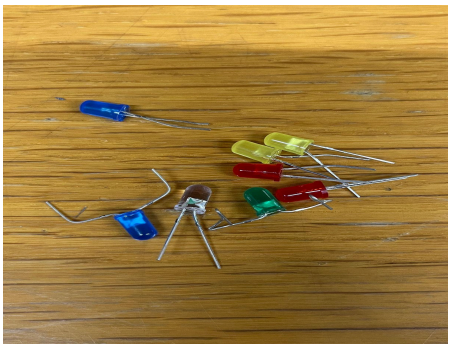
This project uses the breadboard to prototype the test circuit connections without soldering. This allows for easy connections and modifications. It consists of “holes” which allow for easy insertion of components into the prototype. Each of the 5 vertical rows on the board consist of a group of 5 holes, which are all connected. The central isolation gap between the two horizontal groups ensures proper separation for integrated circuits, these groups are not connected.

Wires (male-to-female jumper wires)



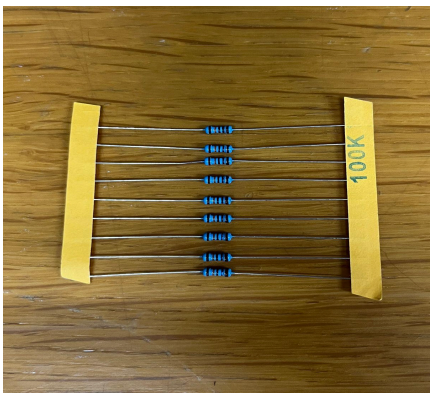
The wires consist of male and female pins on each end, which features a 3mm pitch connector designed for prototyping applications with breadboards. The cable assembly consists of individually insulated conductors with male and female pin headers on each end. This allows interconnections between circuit and electric components.

LED Lights



The LED lights serve as visual indicators to show the status of the lights, indicating to pedestrians when to cross over and when not to. This project makes use of conventional colours being Red, Green and Yellow lights to indicate to pedestrians when to cross over and when to wait. When connected correctly, they emit brightly coloured lights during the day and night, making them ideal for signalling.

100k Ohm Carbon Resistor



The 100K Ohm resistors are essential components designed to precisely limit and control current flow within the circuit. They ensure the voltage stays within safe operating ranges. As well as ensuring sufficient current enters the components, avoiding potential burnouts within components. For the purpose of this project, the resistors are connected across the LED lights, ensuring sufficient current is flowing. Each resistor has a maximum voltage of 250V and a tolerance of 5%.

Raspberry PI

The Raspberry Pi 400 (est. 2021) ultimately serves as the central processing unit for the traffic light system. It integrates hardware control, software execution and network communication into a single-board computing platform. It holds the GPIO controls, ensuring the LEDS on/off purpose. It is responsible for the client-server communication between the user and server which is executed on the Operating System using Python. It also influences the resistor behavior set in GPIO pins and LED performance.

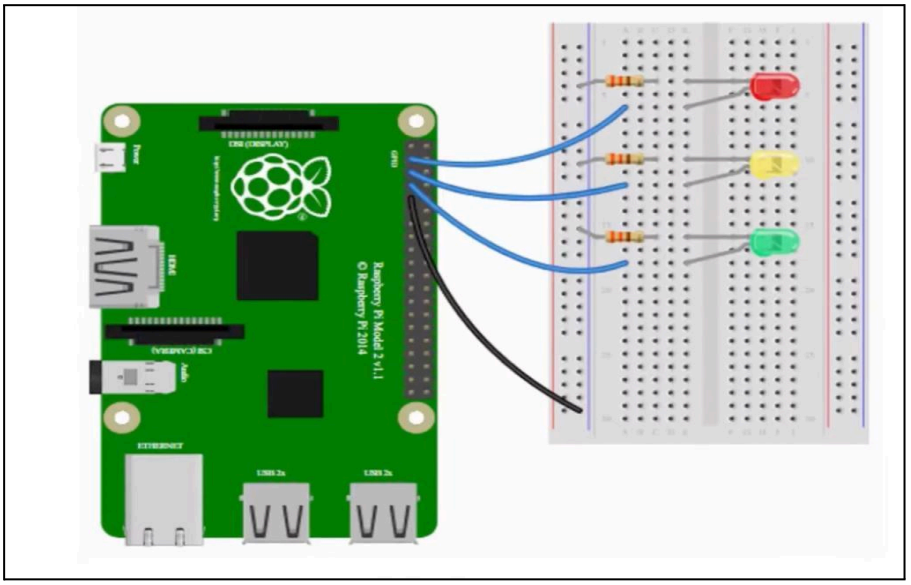
MicroSD Card/Adapter

The MicroSD card/adapter serves as the primary storage unit for the Raspberry Pi. It is a non-volatile memory component which holds the location for the Python scripts and libraries. The SD card/adapter may also host the operating system, with Python residing on the partition.

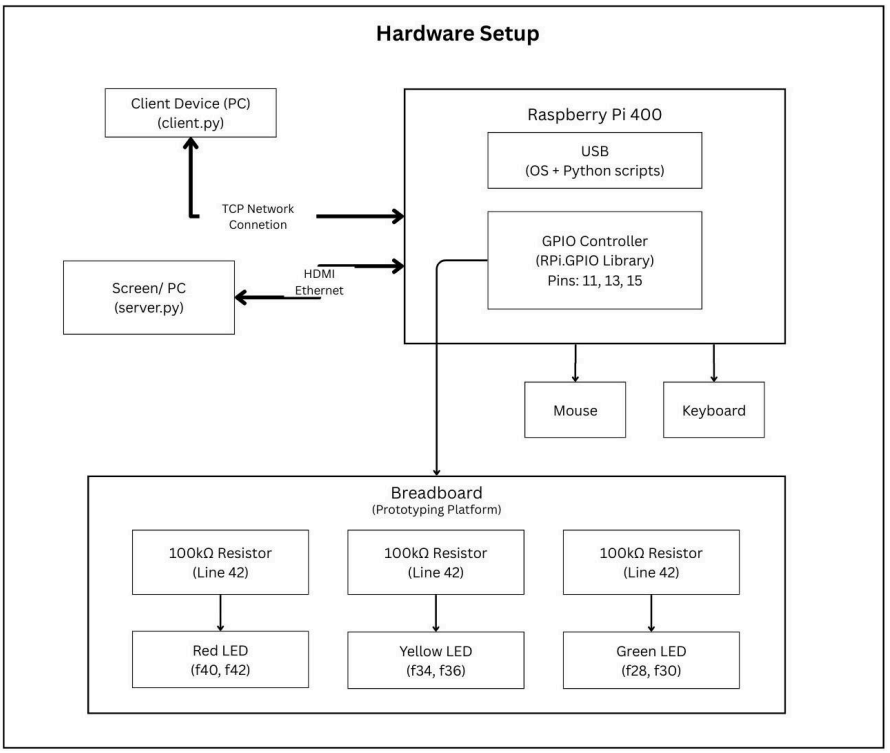
Circuit (Block) Diagram

The following diagram explains the interaction between the hardware components:

Fritzing Design

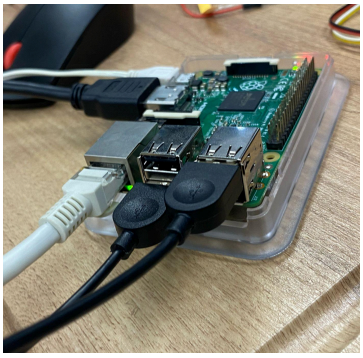


Circuit (Block) Design

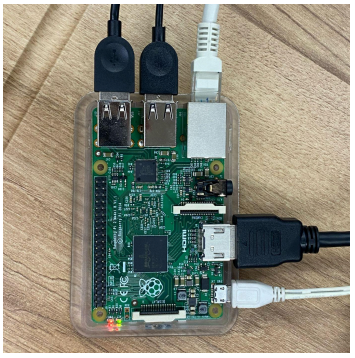


Technical Implementation

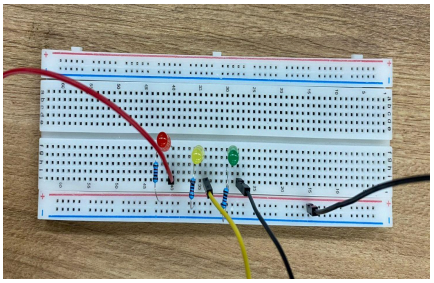
The following is a step-by-step guide on how this project was implemented.



The hardware setup on the Raspberry Pi involved connecting an Ethernet cable (thick cream cable), an HDMI cable (black cable), and standard USB cables for the mouse and keyboard. As the system relies on a connection-oriented protocol, **TCP** was used for communication between the client and server.

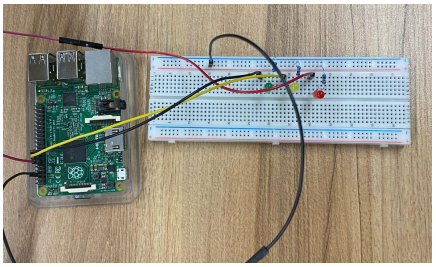


Three LED lights (red, yellow, and green) were integrated into the system using a breadboard. Each LED was connected with a resistor as follows (note 'f#' represents a specific hole on the breadboard):



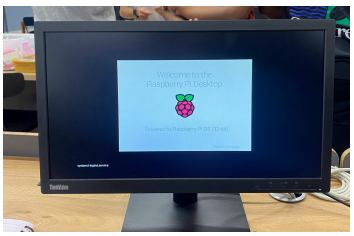
- The red LED was connected to f40 and f42, with its resistor positioned on line 42.
- The yellow LED was connected to f34 and f36, with its resistor positioned on line 35.
- The green LED was connected to f28 and f30, with its resistor positioned on line 30.

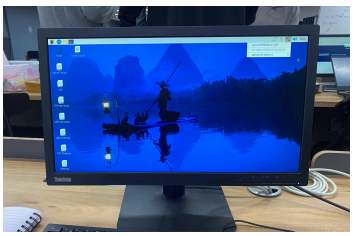
The GPIO (General Purpose Input/Output) connections from the Raspberry Pi to the breadboard were as follows:



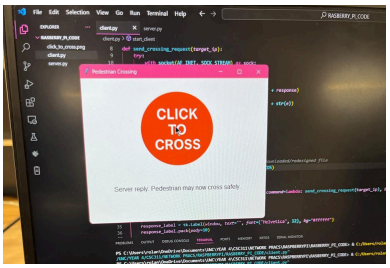
The yellow LED circuit, located at j34 on the breadboard, was connected to GPIO 27 (Pin 13 on the Raspberry Pi). The red LED circuit, located at j40, was connected to GPIO 17 (Pin 11 on the Raspberry Pi). The green LED circuit, located at f28, was connected to GPIO 22 (Pin 15 on the Raspberry Pi).

Subsequently, to the hardware setup, the software component of the implementation was prepared. The Raspberry Pi was accessed through its graphical interface as seen below:

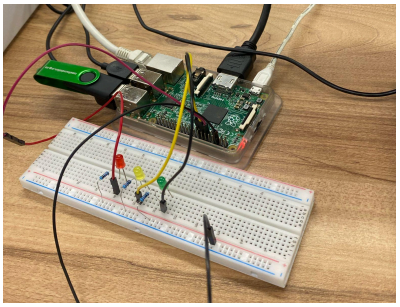




Communication between the Raspberry Pi and the client device (a teammate's PC) was established using a Python-based TCP client-server model. The server.py script was executed on the Raspberry Pi to continuously listen for incoming requests (connections), whilst the client.py script was executed on the teammate's PC to send these requests. The client code then introduces a GUI implementation of a red "click to cross" button, as seen in the image alongside. Which replicated a pedestrian pressing a button to cross the road, in other words "A Pedestrian Priority Traffic Light Simulation"



This TCP communication was implemented using Python's built-in socket module. The server was configured to bind to a specific IP address and port, accept client connections, and respond to requests by controlling the LED lights based on the respective received signals.

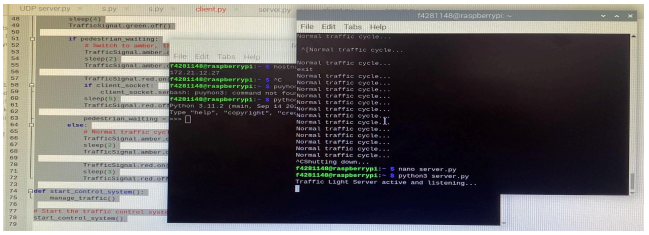


Process taken to install the python scripts onto the Raspberry Pi (RPi):

- The client.py and server.py files were first copied from a teammate's PC to an external USB drive.
  - This USB drive was then connected to the Raspberry Pi via a USB port on the RPi.
- Once the USB was recognized by the Raspberry Pi, the files were opened and transferred onto the RPi.

Thereafter, the following commands were used to install the necessary packages to run the server code on the RPi for implementation:

1. sudo apt update
2. sudo apt install python3-rpi.gpio
3. nano server.py then the server code is copied and pasted to the nano server.py
4. python3 server.py



Once the above was complete the following message was printed on the command terminal, "Traffic Light Server active and listening…". The next step was to the client.py script on the teammates PC, which was connected to the Raspberry Pi using TCP to start the LED control process.

This client-server setup allows the two devices to communicate in real time using a reliable method (TCP). This ensures the messages are delivered correctly and in the right order, which is important for controlling the traffic lights based on when a pedestrian presses the button.

Conclusion

This project successfully developed a client-controlled traffic light system with pedestrian priority using a Raspberry Pi server and TCP communication. It demonstrated how IoT technology can be applied to improve urban safety and traffic efficiency. The key achievements included establishing reliable client-server communication, implementing pedestrian priority functionality, and achieving full hardware-software integration.

Several technical challenges were encountered throughout development. Faulty components such as defective LEDs, loose jumper wires, and poor breadboard connections caused inconsistent outputs. System lag during boot-up due to an older SD card, as well as IP mismatches between the Raspberry Pi and the teammates PC, added further complexity. A particularly challenging issue occurred on 11/04/2025, when all the components were properly connected, yet the LED failed to respond, creating the impression of a system failure when the issue was due to a defective LED. Additionally, incorrect GPIO pin assignments between the physical setup and Python code required repeated testing and code updates. Difficulties in running the server code on the Raspberry Pi also hindered progress temporarily.

Despite these obstacles, all major issues were resolved through iterative testing, code refinement, and component replacement, resulting in a functional and scalable prototype ready for future enhancements.

There are several potential improvements to elevate the system’s functionality and adaptability. Namely, the following few enhancements: Real-time traffic adaptation, which entails integrating data from motion sensors or cameras to dynamically adjust light durations based on traffic flow, reducing congestion. Another improvement could be implementing emergency vehicle priority into the system, this allows automatic detection of emergency vehicles and granting them right-of-way. Furthermore a pedestrian and cyclist detection could be implemented to increase the safety of non-motorized users through automated crossing recognition. Another inclusive implementation would be providing accessibility features, which would include mobile notifications, tactile or voice alerts for the visually impaired, and multilingual support to accommodate diverse populations.

These amplifications would significantly improve the system's intelligence, responsiveness, and inclusivity, making it better suited for deployment in modern smart cities.

Group Members and Their Contributions

Name	Student Number	Project Contribution:
Deogracias (Team Leader)	4203232	Hardware, Software, Slide Creation, Literature Review Refinement
Rolande	4269355	Report Refinement, Python Code, Final Implementation, Cover Page Design
Gugulethu	4376487	Fritzing design, Literature Review and Abstract, Technical implementation, Final implementation (Hardware)
Eyethu	4133541	Fritzing design, Hardware connections, Literature review, Conclusion
Sphesihle	4202730	Literature Review: Key technologies used, Hardware descriptions
Jordan	4127304	Literature Review: Challenges and Gaps in research, Block Diagram, Conclusion
Craig	4323965	Literature Review, Hardware Descriptions
Lesedi	4276507	Literature Review: Why is this project relevant & Key technologies, Conclusion
Siyamthanda	4327595	Abstract, Literature Review: Why is the project relevant
Peace	4352859	Literature Review, Research Design-fritzing, Hardware connections

References

1. Chen, Y. and Cassandras, Christos G (2023). *Adaptive Traffic Light Control for Competing Vehicle and Pedestrian Flows*. [online] arXiv.org. Available at: https://arxiv.org/abs/2303.08173 [Accessed 20 Mar. 2025].
2. Clatworthy, B. (2024). *Hi-tech traffic lights could give cyclists the priority over cars*. [online] Thetimes.com. Available at: https://www.thetimes.com/uk/transport/article/traffic-lights-that-prioritise-cyclists-over-cars-to-be-trialled-rv3hqztwx [Accessed 20 Mar. 2025]
3. Li, Y., Kamalasanan, V., Batista, M. and Sester, M. (2022). *Improving Pedestrian Priority via Grouping and Virtual Lanes*. [online] arXiv.org. Available at: https://arxiv.org/abs/2205.08783 [Accessed 20 Mar. 2025]
4. Global Designing Cities Initiative (2025). *Pedestrian Priority Spaces.* [online] *Global Street Design Guide.* Available at: https://globaldesigningcities.org/publication/global-street-design-guide/streets/pedestrian-priority-spaces/ [Accessed 20 Mar. 2025].
5. Research.google. (2016). *Physical and Virtual Cell Phone Sensors for Traffic Control: Algorithms and Deployment Impact*. [online] Available at: https://research.google/pubs/physical-and-virtual-cell-phone-sensors-for-traffic-control-algorithms-and-deployment-impact/ [Accessed 20 Mar. 2025].
6. Peter Nimiac , Andrecj Krpic and Bostjan Batagelji(2022) *Pedestrian Traffic Light Control with Crosswalk FMCW Radar and Group Tracking Algorithm* [online] Available at : https://www.mdpi.com/1424-8220/22/5/1754 [Accessed 20 Mar. 2025]
7. City of Cape Town , Media Office(2022) *Innovative thermal pedestrian crossings sensing your moves* [online] Available at : https://www.capetown.gov.za/Media-and-news/Innovative%20thermal%20pedestrian%20crossings%20sensing%20your%20moves [Accessed 20 Mar. 2025]
8. TableTalk(2023) *Thermal sensors to keep the traffic flowing [online] Available at* : https://tabletalk.co.za/news/2022-10-19-thermal-sensors-to-keep-the-traffic-flowing/#google_vignette [Accessed 20 Mar. 2025]
9. Daniel Stofan(2023) *6 Smart Cities that get traffic control right*[online] Available at : https://blog.goodvisionlive.com/6-smart-cities-that-get-traffic-control-right [Accessed 20 Mar. 2025]
10. Stofan, D. (2023). *How Singapore's Smart Traffic System is Redefining Urban Mobility.* [online] Available at: https://www.intellistride.com/blog/how-singapores-smart-traffic-system-is-redefining-urban-mobility/ [Accessed 20 Mar. 2025].
11. Cecilia Duong (2023). Pedestrians should get the green light on traffic signal prioritisation [online] Available at: https://www.rciti.unsw.edu.au/news/pedestrians-should-get-the-green-light-on-traffic-signal-prioritisation [Accessed 20 Mar. 2025]
12. Challenges and Limitations of Adaptive Signal Control (n.d.) Available at: https://fastercapital.com/topics/challenges-and-limitations-of-adaptive-signal-control.html [Accessed 20 Mar. 2025]
13. Development and implementation of a Raspberry Pi-based….(2024) https://www.sciencedirect.com/science/article/pii/S2772375524001357 [Accessed 20 Mar. 2025].

Python Code

https://github.com/DeograciasM/Client-Controlled-Traffic-Light-System-with-Pedestrian-Priority is where all the relevant code pertaining to this project will be found.

.