Abstract

The “Treasure X” project utilizes portable microcontrollers with advanced tracking technologies to create an engaging and educational interactive experience. Utilizing a Raspberry Pi 4 Model B, the system is designed to locate eight distinct Bluetooth beacons by leveraging Bluetooth Low Energy (BLE) technology and measuring the Received Signal Strength Indicator (RSSI) to estimate the distance to each beacon. Upon detection, the GPS coordinates of each beacon are recorded and mapped on Google Maps, revealing the location of a “treasure” at the calculated center. This project highlights the coaction between hardware and software, featuring a 4G/GPS module for enhanced connectivity and real-time data transmission. Custom scripts were developed for BLE scanning, GPS data retrieval, and Google Maps integration. Encased in a 3D-printed housing for portability and durability, and powered by a rechargeable battery for extended use, the “Treasure X” project exemplifies how microcontroller-based systems can revolutionize learning environments, turning traditional education into dynamic, interactive experiences.

Contents

List of Figures

1. Introduction

The “Treasure X” project aims to revolutionize interactive and educational experiences by employing the power of portable microcontrollers and modern tracking technologies. Traditional methods of engaging students in learning activities often lack the dynamic and interactive elements needed to captivate their interest and provide practical, hands-on experiences. This project offers an innovative solution: a system that uses a Raspberry Pi 4 Model B to locate Bluetooth beacons, record their GPS coordinates, and map them on Google Maps to reveal a hidden “treasure” at the calculated center point.

By employing Bluetooth Low Energy (BLE) technology and measuring the Received Signal Strength Indicator (RSSI), the system accurately determines the distance to each beacon. The integration of a 4G/GPS module provides reliable connectivity and real-time data transmission, enhancing the overall efficiency of the process. Custom-developed scripts for BLE scanning, GPS data retrieval, and Google Maps API integration are used to store and visualize the beacon locations.

An interactive graphical user interface (GUI) system enhances user engagement by providing a welcoming trivia game, where users answer randomly picked multiple-choice questions to the beacons they find. This element adds an educational and entertaining layer to the experience, making the search for the treasure more engaging and fun.

The system is housed in a capaciously designed 3D-printed enclosure, making it portable and suitable for various applications beyond the initial educational-trivia game concept. This project represents a significant step forward in blending education with technology, offering a practical demonstration of how microcontroller-based systems can be used to create engaging and interactive learning environments.

2. Hardware

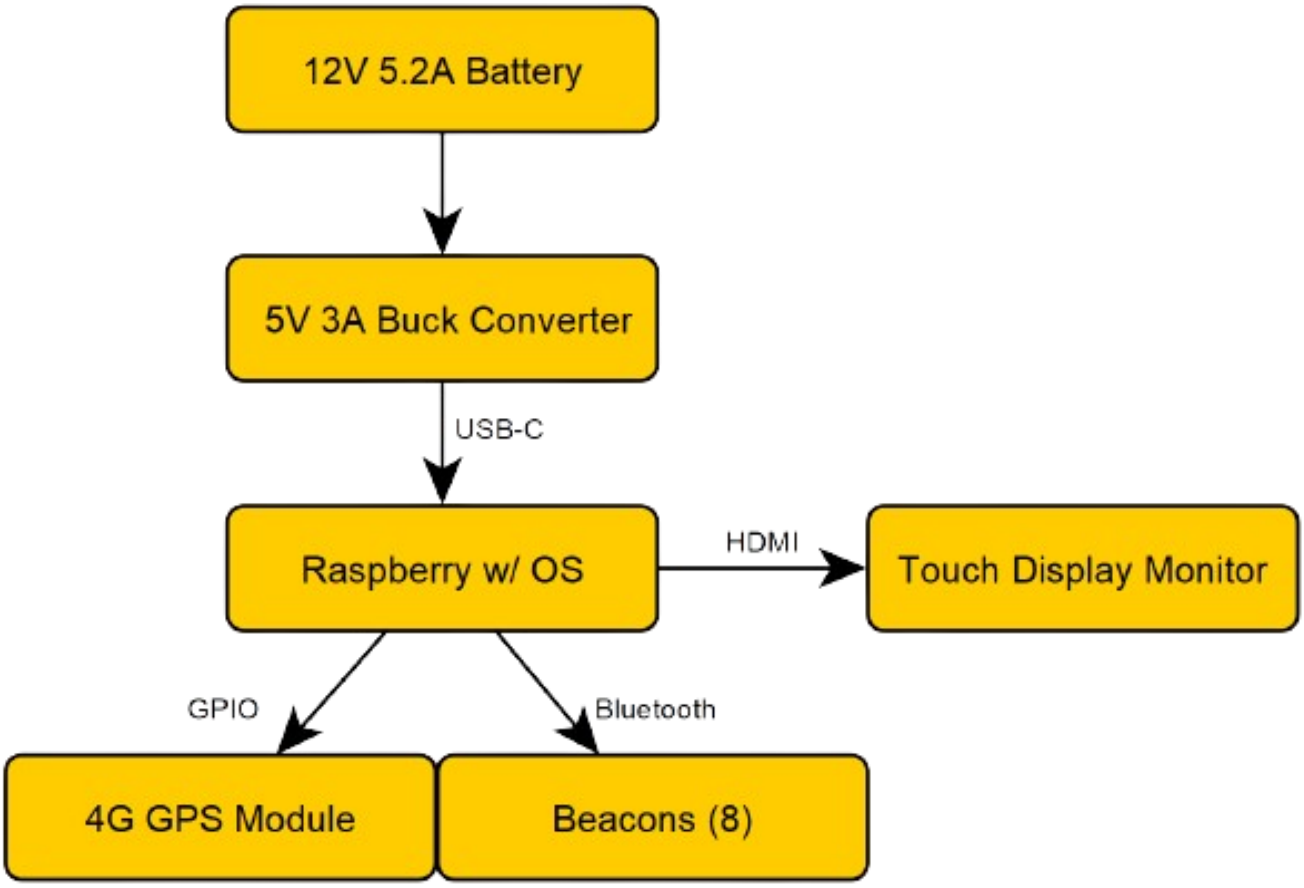


Figure 1: Hardware Design Schematic

2.1 Hardware Overview

The hardware architecture of the "Treasure X" project seamlessly utilizes essential components to provide functionality and portability. At the core of the system is the Raspberry Pi 4 Model B, powered by a rechargeable battery regulated by a buck module to convert power through a USB output safely. This setup ensures 6-9 hours of continuous operation. The Raspberry Pi is equipped with a 4G HAT module, connected via Micro-USB, that facilitates real-time data transmission and GPS functionality through the SIM7600A-H 4G/GPS module.

User interaction and real-time feedback are managed through a mounted touch display connected to the Raspberry Pi through Micro-USB and Nano-HDMI. The entire setup is housed in a custom-designed 3D-printed enclosure, featuring dedicated compartments to assemble the Raspberry Pi, battery, and monitor.

The integration of these components creates a reliable and portable system capable of locating Bluetooth beacons, recording their GPS coordinates, and mapping them on Google Maps. This makes the “Treasure X” project an innovative solution for interactive learning experiences.

2.2 Power Supply & Regulation

The power supply and regulation system of the project are designed to provide reliable and continuous operation, making the system suitable for extended use in portable environments. Central to this setup is the HiLetgo buck module, which is responsible for safely converting the power source to output via USB using the LTC4056. This module provides a stable and efficient power supply to the Raspberry Pi 4 Model B, ensuring that the microcontroller receives the appropriate voltage and current through a USB to USB-C connection.

In addition to the LTC4056 module, the system is equipped with a rechargeable battery that offers approximately 6-9 hours of continuous operation. This battery is intended for maintaining the portability of the system, allowing it to be used without the need for a constant external power source. The rechargeable nature of the battery also supports sustainable use, as it can be recharged and reused multiple times.

To facilitate user interaction and system monitoring, a micro-USB to USB connection and a Nano-HDMI to HDMI adapter are utilized to connect a touch display to the Raspberry Pi. These connections ensure that the display receives adequate power and proper signal transmission, allowing for seamless interaction and real-time feedback.

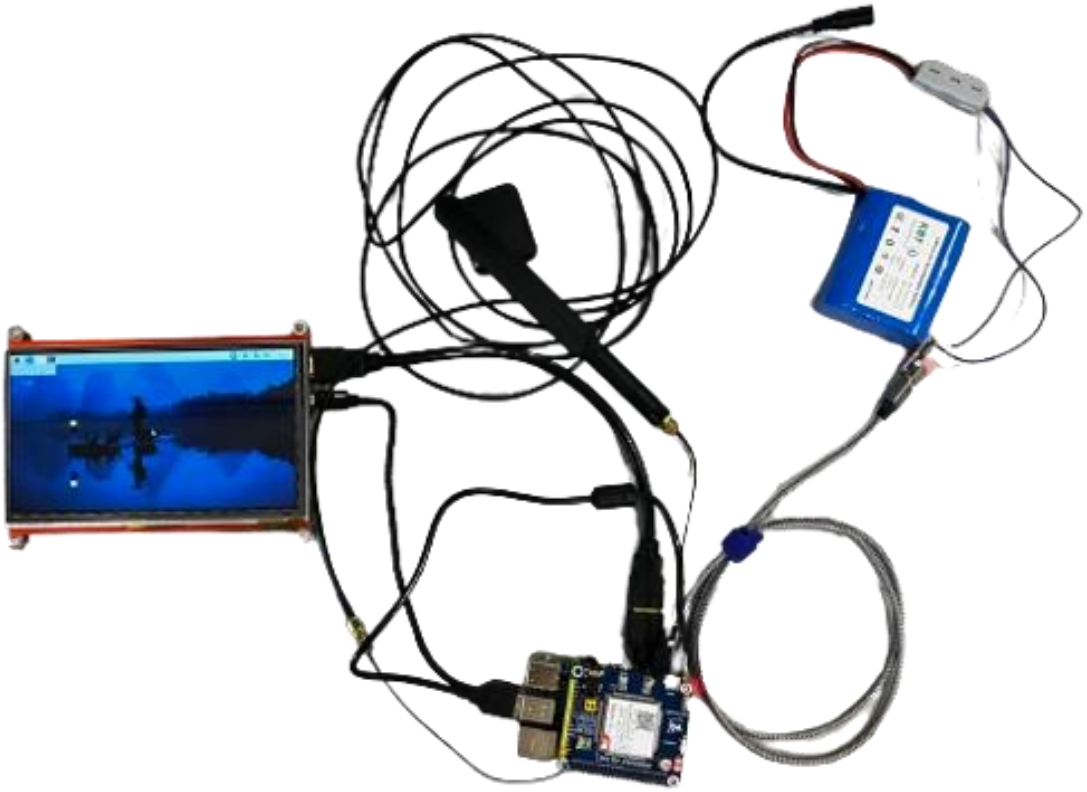


Figure 2: Assembled Power & Portability (Demonstration)

2.3 4G Connectivity

The connectivity infrastructure is a pivotal aspect that enables real-time data transmission and precise location tracking, essential for the project's functionality. At the core of this setup is an augmented 4G HAT module connected via GPIO pins of the Raspberry Pi. This module endows the system with cellular connectivity using a preordered T-Mobile SIM card, enabling access to the internet and data transmission.

A critical component of the connectivity setup is the SIM7600A-H 4G/GPS module. This module supports 4G LTE connectivity, offering fast and reliable internet access, while also utilizing GPS functionality for accurate location tracking. The combination of these features ensures that the system can locate Bluetooth beacons and upload their coordinates to Google Maps in real-time.

The integration of the 4G HAT module and the SIM7600A-H 4G/GPS module transforms the Raspberry Pi into a powerful, connected device capable of handling multiple communication protocols. The cellular connectivity provided by the 4G LTE network ensures that the system can operate independently of local Wi-Fi networks. This independence is crucial for outdoor and mobile use, where Wi-Fi connectivity might be unreliable or unavailable.

Overall, the 4G connectivity provided by the SIM7600A-H module is a critical component of the project, enabling reliable and efficient data communication. This ensures that the system can operate effectively in real-time, providing users with an accurate and engaging interactive experience.

2.4 Display & Interaction

The display and interaction capabilities are important to providing a user-friendly and engaging experience. The project uses the functionality of a HS-007 touch display, manufactured by Head Sun, connected to the Raspberry Pi 4 Model B via micro-USB and Nano-HDMI. This setup allows for communication between the display and the microcontroller, ensuring that users receive real-time feedback and interact with the GUI system.

The touch display serves multiple purposes within the project, primarily as the user interface that presents visual information and controls to facilitate interaction with the system. Users can view the status of the system and monitor the location of Bluetooth beacons on a map. This visual feedback is for navigating the “Treasure X” game, as it guides users to the beacons and ultimately to reveal the hidden treasure.

Users can directly interact with the display to input commands, navigate through GUI menus, and access various functions of the system. This intuitive method of interaction reduces

the learning curve and makes the system accessible to a broader audience, including those who may not be familiar with on-screen input methods without physical keyboards or mice.

Figure 3: Booted Monitor w/ Raspberry OS

Moreover, the touch display's connectivity via micro-USB and Nano-HDMI ensures these connections have direct input/output ease of use, meaning the display receives clear signal transmission directly from the Raspberry Pi. The combination of these features allows the display to function smoothly as a touchscreen while providing a consistent and responsive experience.

2.5 Enclosure & Portability

The enclosure and portability aspects are essential to ensuring the system is durable, user-friendly, and suitable for a variety of mobile environments. The entire hardware setup is housed in a custom-designed 3D-printed enclosure, which has been modeled to accommodate all components securely and generously.

The custom 3D-printed enclosure includes a dedicated case for the Raspberry Pi 4 Model B and other components, providing a stable and protective environment for the central microcontroller and its modules. Additionally, the enclosure features a shelf specifically designed to hold the rechargeable battery and cables so that they remain in place during operation and transportation. The design also incorporates a secure platform for the touch display accessibility.

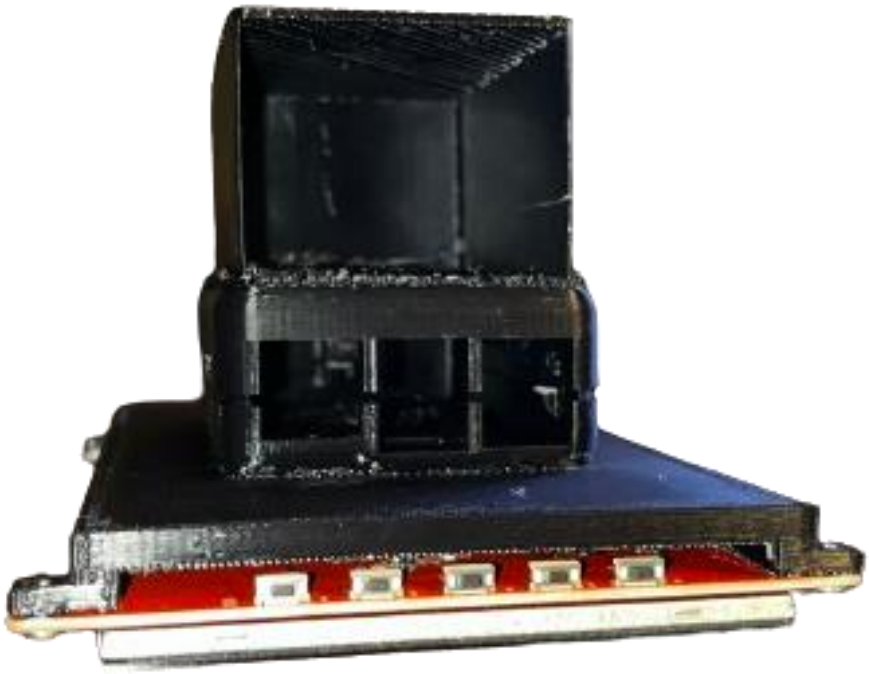


Figure 4: Assembled Top & Bottom Test Print (Demonstration)

The 3D-printed enclosure is not only durable but also lightweight, making it easy to carry and use in various outdoor and indoor settings. This portability is essential for the project's goal of creating an interactive and engaging educational tool that can be used in different environments, from classrooms to outdoor adventure scenarios.

The enclosure's design also prioritizes user convenience with carefully placed openings and slots for cables and connectors, allowing for easy storage and maintenance. Additionally, a dedicated hole is designed specifically for the antenna to slide into, intended to minimize the risk of damage and maximize a secure connection.

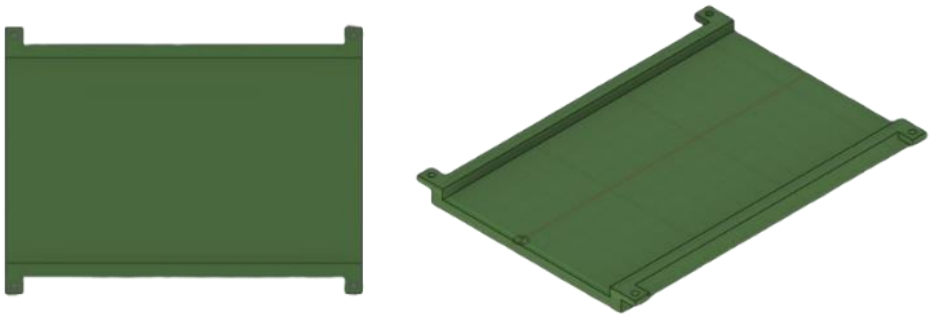


Figure 5: Final Fusion360 Top Print for Hardware Portability

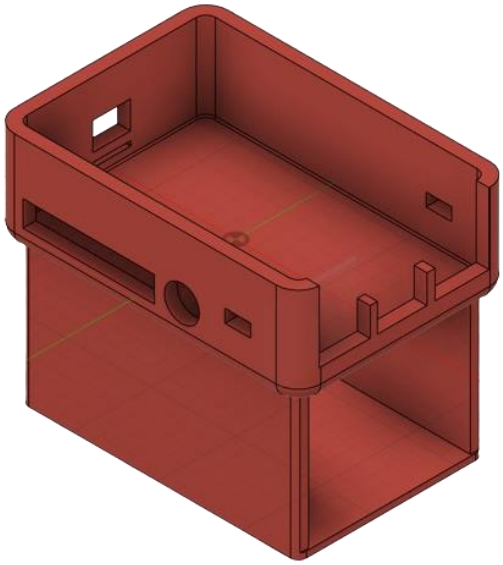


Figure 6: Final Fusion360 Bottom Print for Hardware Portability

2.6 Power Management

The LTC4056 module is tasked with converting 5V and 3A from the power source and output via USB for safe cable connection to power the Raspberry Pi, providing a stable and efficient power supply. This conversion is essential to protect and provide the microcontroller and the display as it operates without experiencing power fluctuations or drops. The module's ability to provide a stable output voltage is to maintain the reliability of the system’s performance, particularly during extended periods of use. Additionally, the power management system incorporates a rechargeable battery as the power source which offers approximately 6-9 hours of continuous operation.

The design of the power management system also includes considerations for energy efficiency. By using components that have low power consumption and integrating them into a well-regulated power supply, the system maximizes battery life and minimizes energy waste. This efficiency is crucial to allow users to rely on the system for extended periods without the need for frequent recharging.

Figure 7: LTC4056–to–Battery Connection (Demonstration)

Moreover, the power management system supports all power-dependent components in active operation with the necessary voltage and current. This includes the Raspberry Pi, the touch display, the 4G/GPS module, and other peripherals. By providing a consistent and reliable power supply, the system avoids interruptions and maintains its functionality throughout its use.

In conclusion, the power management system is designed to provide efficient, reliable, and sustainable power to all components. The use of the LTC4056 module, combined with a rechargeable battery, is purposed for stable voltage levels and extended operation times. This comprehensive approach to power management is fundamental to the project's success, supporting its goals of portability, reliability, and user-friendly operation.

3. Software

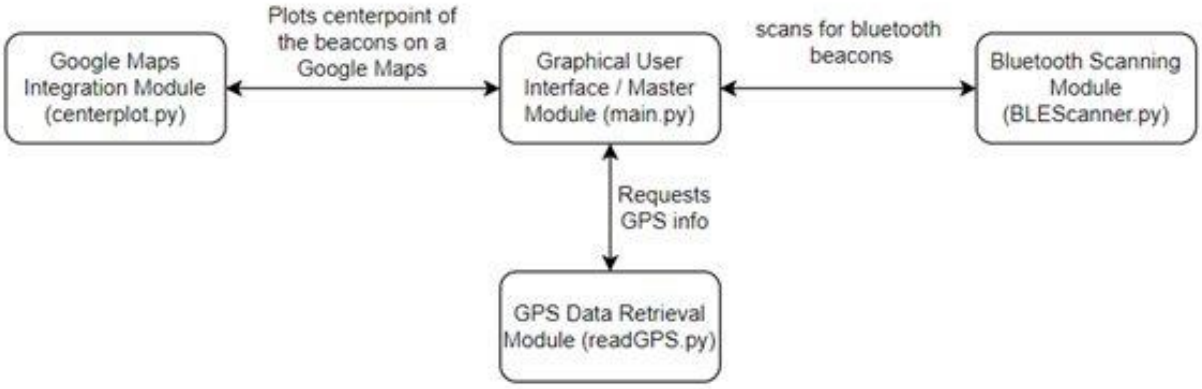


Figure 8: Software Module Flowchart

3.1 Software Overview

The software component of the "Treasure X" project is thoughtfully designed to smoothly integrate with the hardware, providing a user-friendly and interactive experience. This integration relies on several custom-developed Python scripts and modules that facilitate Bluetooth scanning, GPS data retrieval, and mapping on Google Maps.

The Bluetooth scanning module utilizes the 'bluepy' library to continuously scan for specific and nearby Bluetooth Low Energy (BLE) beacons. It measures the Received Signal Strength Indicator (RSSI) to provide accurate detection based on unique MAC addresses and RSSI values. This module is vital for identifying and locating the beacons required for the project.

GPS functionality is managed through a script that interfaces with the SIM7600A-H 4G/GPS module using AT commands. This script retrieves GPS coordinates and converts them from Degrees Minutes Seconds (DMS) format to Decimal Degrees (DD) format. The coordinates are then used to pinpoint the locations of the beacons.

A custom Python script integrates with the Google Maps API to generate a map displaying the found beacon locations. The terminal provides real-time feedback as each beacon is detected

and its coordinates are pinned on the map. Once all 8 beacons are found, the map displays all beacon locations along with the calculated center point, marked as the “treasure.”

The graphical user interface (GUI), designed with Python's ‘Tkinter’ library, offers an intuitive platform for interacting with the system. It features buttons to start the beacon search, display the trivia questions, and view progress and information regarding the game. The GUI incorporates an engaging and welcoming element to the beacon-hunting process.

Data communication is facilitated over a preordered T-Mobile 4G network using the SIM7600A-H module. This setup enables real-time data transmission, remote access, and monitoring, ensuring continuous operation without the need for physical presence.

Overall, the combination of BLE scanning, GPS data retrieval, mapping, and an intuitive GUI ensures a captivating user experience. The project’s software integration with the hardware creates a reliable tool in interactive learning.

3.2 Bluetooth Scanning & RSSI Measurement

The Bluetooth Scanning and RSSI Measurement module is designed to detect and locate Bluetooth Low Energy (BLE) beacons accurately. This functionality is implemented using the ‘bluepy’ library in the *BLEScanner.py* script, found in Appendix D. The code is structured to continuously scan for nearby BLE devices, measure the Received Signal Strength Indicator (RSSI), and provide feedback on the proximity of the detected beacons.

The scanning process begins with the initialization of the ‘Scanner’ object from the ‘bluepy.btle’ library to start scanning for BLE devices. A list of known beacons' MAC addresses is maintained to filter and identify relevant devices while the script enters a loop where it continuously scans for BLE devices every 0.5 seconds. As each device is detected, its MAC

address and RSSI value are recorded. The software then filters these results to identify the beacons that are part of the game.

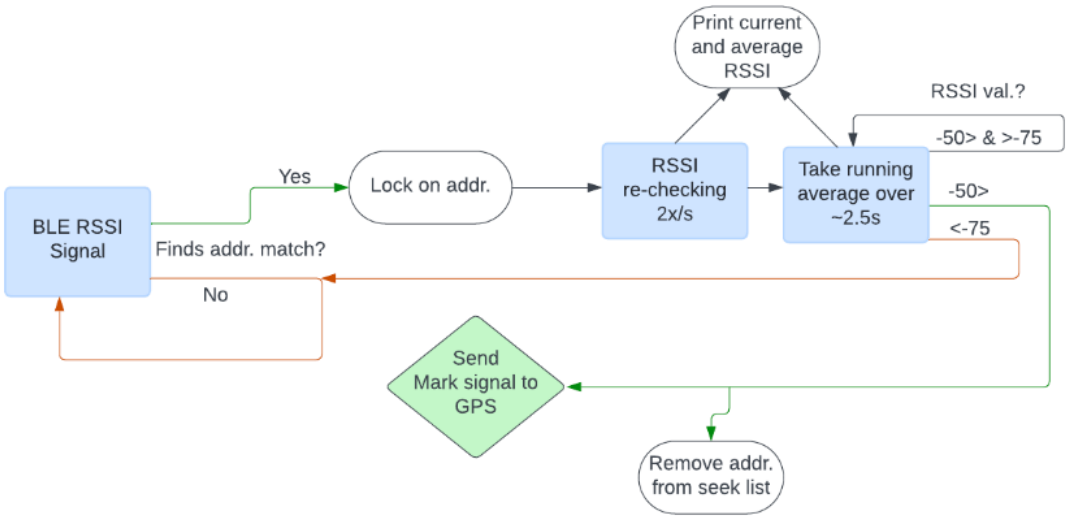


Figure 9: Complete BLE Scanner Flowchart

When a beacon is detected with an RSSI value greater than -75, the script “locks” onto the beacon. This involves tracking the beacon's address and calculating the average RSSI over a series of scans to determine the user's proximity to the beacon. The scanning script is designed to filter out irrelevant devices and focus on the beacons of interest based on their RSSI values and MAC addresses.

The terminal provides real-time feedback to the user based on the average RSSI values. If the average RSSI increases, the user is moving closer to the beacon; if it decreases, the user is moving farther away. This interactive feedback enhances the user experience and ensures accurate tracking and location of the beacons. The feedback is also given through the ‘gui_callback’ function, which updates the graphical user interface (GUI) with messages indicating the user's proximity to the beacon.

```
SIM7600X is starting:
SIM7600X is ready
DEV = c2:7f:0a:ba:09:1d RSSI = -82
DEV = c2:7f:0a:ba:09:1d RSSI = -84
DEV = c2:7f:0a:ba:09:1d RSSI = -85
DEV = c2:7f:0a:ba:09:1d RSSI = -85
DEV = c2:7f:0a:ba:09:1d RSSI = -85
DEV = c2:7f:0a:ba:09:1d RSSI = -82
DEV = c2:7f:0a:ba:09:1d RSSI = -85
DEV = c2:7f:0a:ba:09:1d RSSI = -79
DEV = c2:7f:0a:ba:09:1d RSSI = -79
DEV = c2:7f:0a:ba:09:1d RSSI = -86
DEV = c2:7f:0a:ba:09:1d RSSI = -80
DEV = c2:7f:0a:ba:09:1d RSSI = -76
DEV = c2:7f:0a:ba:09:1d RSSI = -70
Locking on to beacon with address c2:7f:0a:ba:09:1d
DEV = c2:7f:0a:ba:09:1d RSSI = -67
DEV = c2:7f:0a:ba:09:1d RSSI = -67
DEV = c2:7f:0a:ba:09:1d RSSI = -60
DEV = c2:7f:0a:ba:09:1d RSSI = -54
DEV = c2:7f:0a:ba:09:1d RSSI = -48
The average RSSI is -59.2
```

Figure 10: Targeted RSSI Variance & Average Test Results w/ Beacons

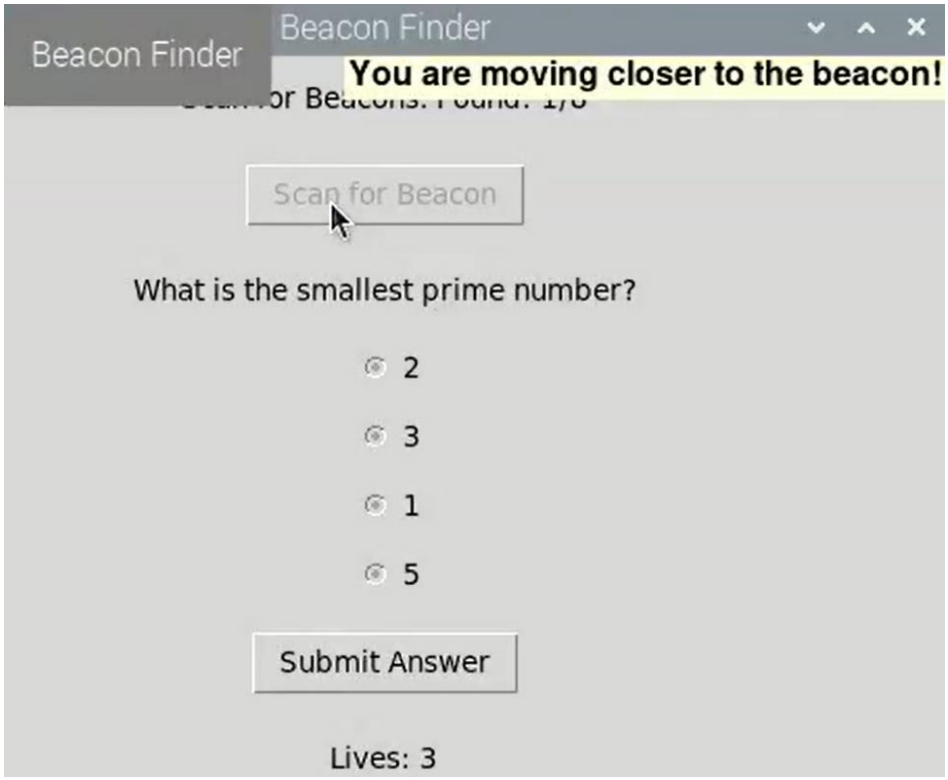


Figure 11: System Prompt for Imminent Beacon (Demonstration)

Once the average RSSI indicates a proximity greater than -50, the beacon is considered found. The script then removes the beacon from the list of active beacons, resets tracking variables, and continues scanning for the remaining beacons. This iterative process ensures that all beacons are accurately detected, and that their locations are recorded. Additionally, if the average RSSI drops below -75, the script disengages the lock on the beacon, indicating that the beacon is too far to track accurately. This feature helps maintain the accuracy of the scanning process so that only relevant beacons are tracked.

The main function of the script initiates the beacon scanning process and provides a callback function to print messages. By implementing these steps, the *BLEScanner.py* script efficiently scans for BLE beacons, measures their proximity using RSSI values, and provides real-time feedback to the user. This module is important to the intention of creating an interactive and engaging educational experience.

3.3 GPS Data Retrieval

The GPS Data Retrieval module is a fundamental component, responsible for obtaining accurate geographical coordinates of the detected beacons. This functionality is implemented using a script that interfaces with the SIM7600A-H 4G/GPS module via AT commands. The *readGPS.py* script, found in Appendix D, handles the GPS data retrieval and reassures that the coordinates are correctly processed and utilized within the project.

The process begins with the GPIO and power-on connection of the GPS module. The script sends specific AT commands to the SIM7600A-H module to activate the GPS functionality and retrieve the current GPS coordinates. The coordinates are initially received in Degrees Minutes Seconds (DMS) format, which is then converted to Decimal Degrees (DD) format for easier manipulation in the mapping process.

Once the GPS module is functioning, the script continuously retrieves GPS data. This involves sending commands to the SIM7600A-H module to obtain the current latitude and longitude. The retrieved data is parsed and converted from DMS to DD format. The retrieved GPS coordinates are then used to pinpoint the locations of the detected beacons. These coordinates are stored and later visualized on the generated map. The accuracy of the GPS data determines the precise locations of the beacons in blue and the calculated center point in red, which represents the hidden “treasure”.

Figure 12: GPS-to-Maps Beacon Interface (Demonstration)

The GPS data retrieval process is further enhanced by utilizing error handling and validation mechanisms. The script checks the validity of the retrieved coordinates and ensures that the data is within acceptable ranges. This helps prevent errors and inaccuracies that could arise from faulty GPS readings or signal issues.

The main function of the *readGPS.py* script initializes the GPS module, retrieves and processes the GPS data, and provides the coordinates are accurately logged and used within the project. By implementing the GPS Data Retrieval module, the project efficiently obtains and processes geographical coordinates, providing the necessary data for the project's mapping and navigation functionalities.

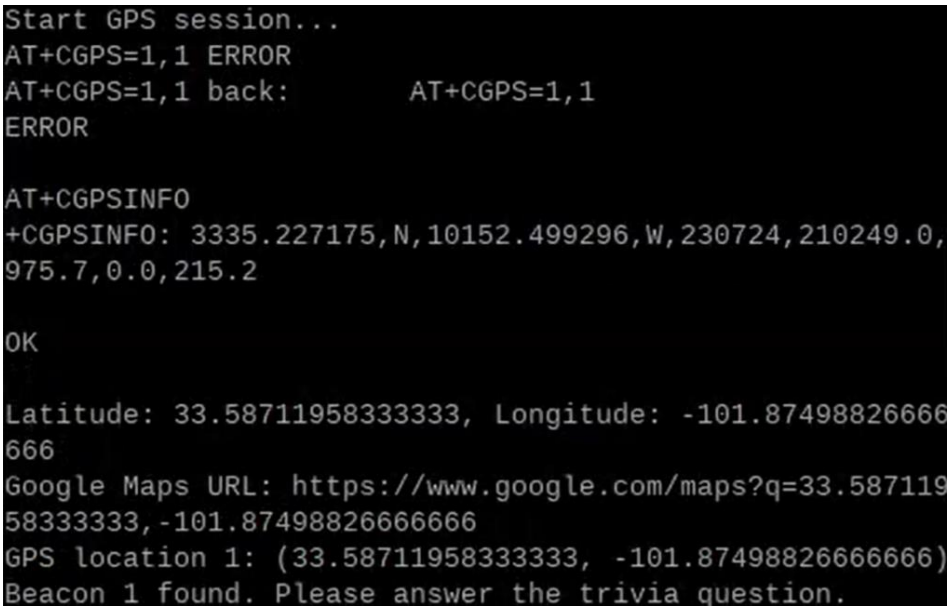


Figure 13: Active GPS Session Preview

3.4 Google Maps Integration

The Google Maps Integration module enables the visualization of the detected beacons' locations on a map. This functionality is implemented using a custom Python script that interacts with the Google Maps API, ensuring that the coordinates retrieved from the GPS module are accurately plotted and displayed. The *centerplot.py* script, found in Appendix D, provides a clear and interactive way for users to view the beacon locations and the calculated center point, which represents the hidden "treasure" after a completed game.

The process begins with the retrieval of GPS coordinates for each detected beacon, as managed by the *readGPS.py* script. These coordinates are stored and later used by the *centerplot.py*

script, which involves calculating the geometric median of the beacon coordinates to determine the central location.

Using the Google Maps API, the script generates a map centered around the calculated median. The coordinates of the detected beacons are plotted as colorized markers with the central point highlighted to indicate the treasure location. The map is designed to be interactive, allowing users to zoom in, zoom out, and pan across the map to explore the beacon locations and navigate towards the treasure. The generated map is saved as an HTML file, which can be viewed in a web browser.

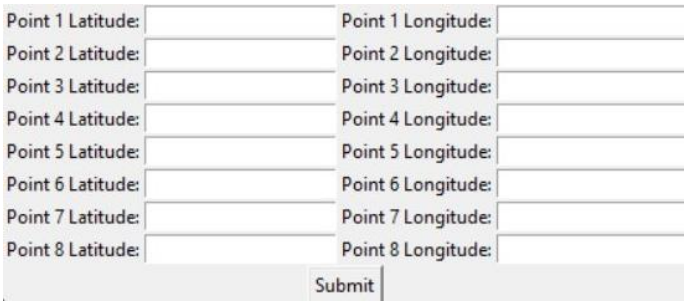


Figure 14: Google Maps Coordinate Window (Demonstration)

The *centerplot.py* script ensures that the GPS data to a completed game is accurately processed and visualized. By integrating with the Google Maps API, the project can provide a clear and interactive map. This integration allows for real-time visualization from the terminal once beacons are detected and their locations are recorded. Once all eight beacons are located, the map provides a final view of all beacon locations with the calculated center point marked as the "treasure" emphasized.

```
Latitude: 33.58667536666667, Longitude: -101.87513616666
666
Google Maps URL: https://www.google.com/maps?q=33.586675
36666667,-101.87513616666666
GPS location 3: (33.58667536666667, -101.87513616666666)
Beacon 3 found. Please answer the trivia question.
Beacon 3 confirmed.
```

Figure 15: Multiple Beacon URL Archive (Demonstration)

3.5 Graphical User Interface (GUI)

The Graphical User Interface (GUI) is designed to provide an active game as a user-friendly and interactive experience. Implemented using Python's 'Tkinter' library, the GUI facilitates user interaction with the system, allowing users to initiate the beacon scanning process, view beacon progress, and navigate through the game. The *main.py* script, found in Appendix D, handles the GUI seamlessly with the hardware and other software components.

Figure 16: GUI Welcome Page

The GUI is initiated through the 'CoordinateApp' class, which sets up the main window and various interface elements. Upon starting the application, the GPS module is powered on to ensure that the system is ready to retrieve location data. The main window of the application is titled "Beacon Finder" and is designed to be intuitive and easy to navigate.

Key features of the GUI include:

1. **Beacon Search**: A prominent button labeled "Scan for Beacon" allows users to initiate the search for Bluetooth beacons. When pressed, the game begins, and the system starts scanning for nearby beacons while the GUI provides real-time feedback on the

scanning progress. The scanning process is managed by the ‘scan_for_beacon’ method, which utilizes the *BLEScanner.py* script to detect beacons and retrieve their RSSI values.

2. **Map Display**: The GUI includes a label that displays the number of beacons found out of the expected total. This label is updated in real-time as beacons are detected and their locations are recorded. The label element provides users with clear feedback on their progress in the game.
3. **Hint & Feedback System**: The GUI features a hint label that provides real-time feedback to users based on their proximity to the detected beacons. This feedback is generated by the ‘gui_callback’ function, which is called during the beacon scanning process to update the user on whether they are moving closer to or farther from a beacon.
4. **Trivia Questions**: The GUI presents users with multiple-choice trivia questions each time a beacon is detected. These questions are randomly selected from a predefined list, and users must answer them correctly to confirm the beacon's location. The trivia questions and answers are managed by the ‘load_trivia_question’ and ‘check_trivia_answer’ methods. The ‘trivia_label’ displays the question, while radio buttons are used for selecting answers. Users have three lives, and incorrect answers reduce their lives, with the game resetting if all lives are lost.
5. **Power Management**: The GUI handles the power state of the GPS module, which reassures it is powered down when the application is closed to conserve battery life. This is managed by the ‘on_closing’ method, which also smoothly shuts down the application.

6. **Lives Display**: The GUI includes a label that showcases the number of lives remaining. This label is updated dynamically as users answer trivia questions correctly or incorrectly. From the 'lives_label' element, users are aware of their remaining chances in the game.

The 'CoordinateApp' class initializes the elements listed above, defining the layout of the GUI, appointed so that users can easily access and interact with the various features of the system. The GUI's utilization with the hardware and other software components provides a cohesive and engaging user experience, contributing to the goal of creating an interactive and educational tool.

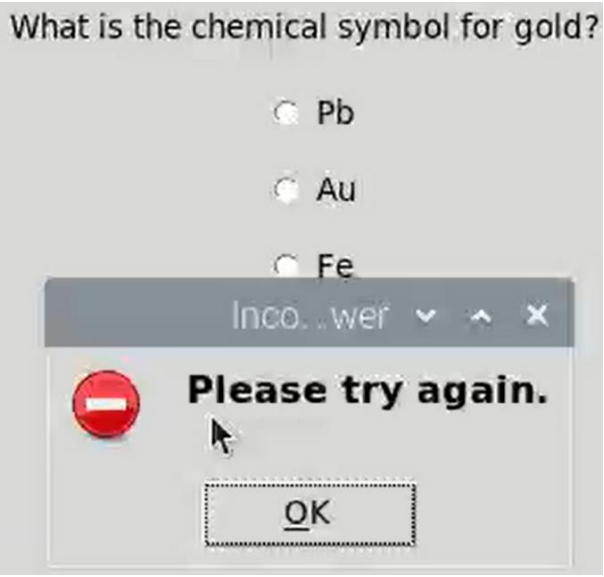


Figure 17: System Prompt for Incorrect Answer (Demonstration)

3.6 Data Communication & 4G Connectivity

The connectivity infrastructure begins with the 4G HAT module, equipped with a preordered T-Mobile SIM card, providing the system with cellular connectivity and enabling it to access the internet for data transmission. The SIM7600A-H module supports 4G LTE connectivity, ensuring fast and reliable internet access, which is critical for real-time data transmission and GPS functionality.

The core functionality of the module is managed by the *readGPS.py* script, which interfaces with the SIM7600A-H module using AT commands. The script initializes the module and sends commands to retrieve GPS coordinates. These coordinates are then processed and used to pinpoint the locations of detected beacons. The continuous interaction with the GPS module ensures that the system receives accurate location data.

The SIM7600A-H module's ability to support both 4G LTE and GPS functionalities allows the system to achieve real-time data communication. Moreover, the 4G connectivity provided by the module allows for remote access and monitoring of the system. This capability is particularly useful for outdoor and mobile use, where local Wi-Fi networks may be unreliable or unavailable. The cellular connectivity concluded for the project allows the system to operate independently, making it highly portable and versatile.

4. Results

The “Treasure X” project concluded in a fully portable, functional, and interactive system that combines both hardware and software components to create an engaging treasure hunt experience. The game is designed with specific rules and objectives to guide users through the process of locating Bluetooth beacons and ultimately discovering the hidden treasure.

4.1 Game Rules & Completion

Users begin the game by powering on the system and accessing the main interface on the touch display. The interface provides a “Scan for Beacon” button that initiates the search for nearby Bluetooth beacons. The system continuously scans for Bluetooth beacons using the *BLEScanner.py* module, which measures the Received Signal Strength Indicator (RSSI) to determine the proximity of each beacon. Real-time feedback is provided through the GUI, indicating whether users are moving closer to or farther from a detected beacon.

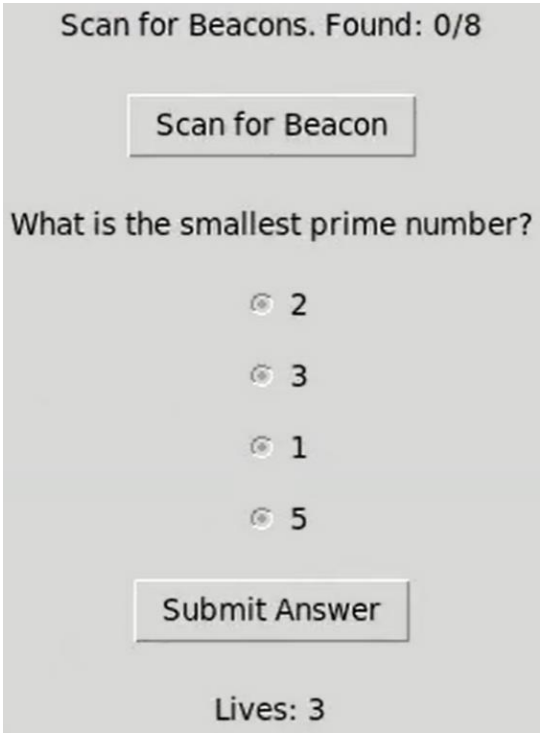


Figure 18: GUI Display (Demonstration)

Upon detecting a beacon, the system presents users with a multiple-choice trivia question related to general knowledge. Correctly answering the trivia question confirms the beacon's location. Each user is given three lives, with incorrect answers reducing the number of lives. If all lives are lost, the game resets. Once a beacon is confirmed, the system retrieves and records its GPS coordinates using the *readGPS.py* module, which interfaces with the SIM7600A-H 4G/GPS module to obtain accurate location data.

This process is repeated until the successful completion of a game is marked by the location and confirmation of all eight Bluetooth beacons. After the eighth beacon is detected and confirmed, the system calculates the geometric median of the recorded GPS coordinates using the *centerplot.py* module. This median represents the location of the hidden treasure. The final map, displaying all beacon locations and the treasure's location, is generated and presented to the user. The map is created using the Google Maps API, ensuring accurate and interactive visualization of the game results.

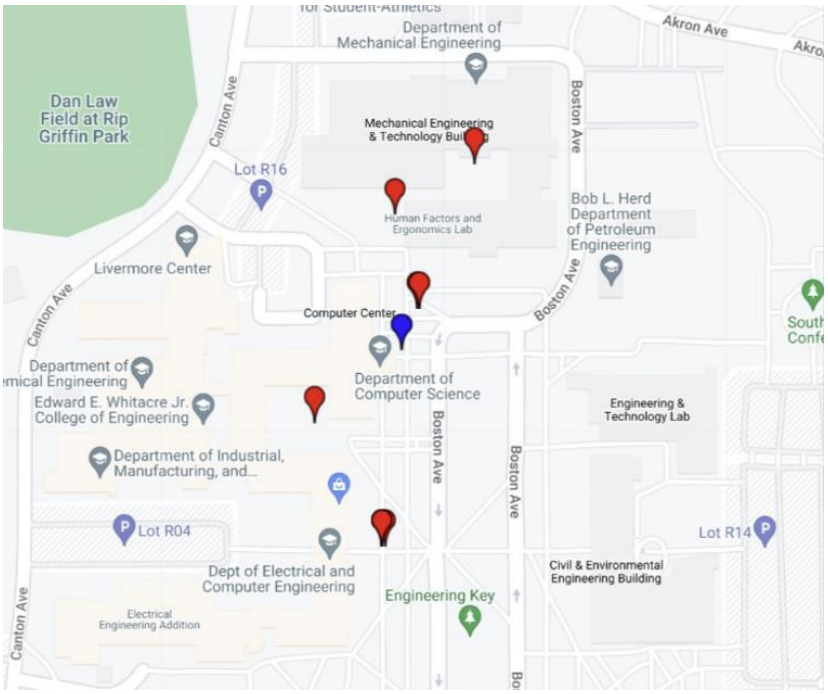


Figure 19: Completed Game w/ Pinned Coordinates (Demonstration)

4.2 Hardware & Software Design

The “Treasure X” project integrates hardware and software components to deliver a seamless treasure hunt experience. The core of the hardware architecture is the Raspberry Pi 4 Model B, powered safely by a HiLetgo buck circuit using an LTC4056 module, supported by a rechargeable battery providing 6-9 hours of operation.

A 4G HAT module, connected via GPIO pins, enables real-time data transmission and GPS functionality through the SIM7600A-H 4G/GPS module, providing fast internet access and precise location tracking. User interaction is facilitated through an HS-007 touch display, connected via micro-USB and Nano-HDMI adapters, ensuring real-time feedback and easy system interaction.

All components are housed in a custom-designed 3D-printed enclosure with dedicated compartments for the Raspberry Pi, battery, and touch display. The enclosure is durable, lightweight, and includes openings for cables and connectors, as well as a slot for the antenna.

Custom Python scripts handle Bluetooth scanning, GPS data retrieval, and mapping. The *BLEScanner.py* module uses the ‘pybluez’ library for accurate beacon detection while the *readGPS.py* script interfaces with the SIM7600A-H 4G/GPS module to retrieve and convert GPS coordinates, which are then mapped using the Google Maps API.

The GUI, designed with Python's ‘Tkinter’ library, provides an intuitive interface with features for starting beacon searches, displaying maps, and viewing beacon information. Data communication over the 4G network ensures real-time data transmission and remote access, keeping the system's live data.

4.3 System Performance

The system's hardware and software components worked together efficiently, ensuring reliable operation and user satisfaction.

The hardware components, including the Raspberry Pi 4 Model B, 4G HAT module, and SIM7600A-H 4G/GPS module, performed reliably in various environments. The power management system, with the HiLetgo buck module and LTC4056 module, provided stable power supply, allowing the system to operate continuously for 6-9 hours on a single battery charge. The custom 3D-printed enclosure proved durable and portable, protecting the internal components while maintaining lightweight.

The software components operated seamlessly with the hardware, delivering accurate and timely data. The Bluetooth scanning module effectively detected beacons, with the RSSI measurements providing precise proximity information. The GPS data retrieval script consistently obtained accurate coordinates, which were correctly mapped using the Google Maps API. The GUI provided an intuitive user interface, facilitating easy interaction and real-time feedback.

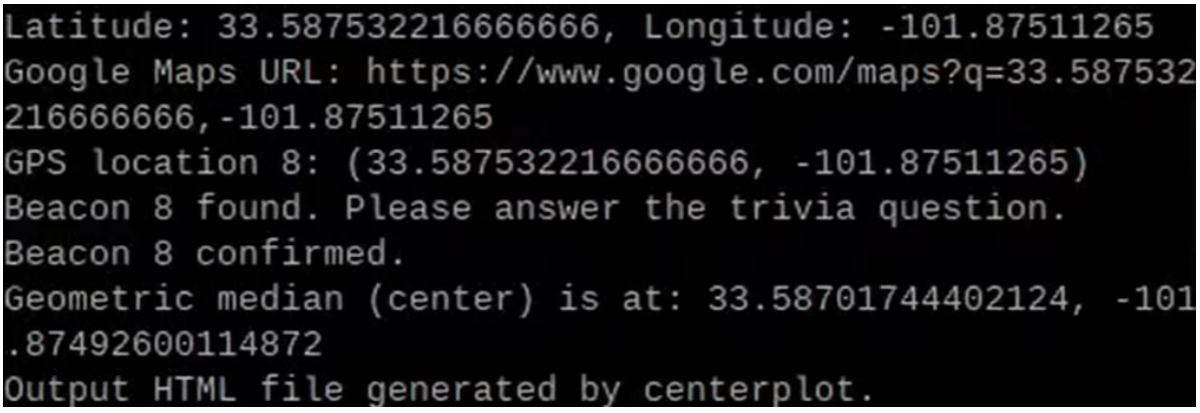


Figure 20: Completed Game w/ HTML File Output Receipt

The system successfully located and confirmed all eight Bluetooth beacons in various indoor and outdoor test scenarios. The trivia questions added an engaging element to the game, with the three-life mechanism enhancing user involvement. The final map, generated upon finding all beacons, accurately displayed the beacon locations and the calculated center point, identified and pinned as the treasure's location.

Overall, the “Treasure X” project showcased excellent performance in both hardware and software aspects. The integration of these components resulted in a reliable, portable, and engaging system, meeting the project's intentions and providing a compelling demonstration of microcontroller-based interactive learning tools.

Figure 21: Assembled Hardware & Software Attestation

5. Summary & Conclusions

The “Treasure X” project set out to develop an innovative and interactive educational tool by utilizing modern microcontroller and tracking technologies. Throughout the project, we successfully designed and implemented a system using a Raspberry Pi 4 Model B to locate Bluetooth beacons, record their GPS coordinates, and map them on Google Maps to reveal a hidden “treasure” at the center. This achievement not only met the project's objectives but also highlighted the potential for utilizing advanced technologies into educational settings to enhance or specifically accommodate learning experiences.

We began by thoroughly exploring and selecting the necessary hardware components, including the Raspberry Pi, LTC4056 power module, 4G/GPS module, and touch display. Each component was carefully chosen for its reliability and efficiency, and they were appropriately utilized to ensure smooth operation. The custom-designed 3D-printed enclosure provided durable and portable housing for the system, making it suitable for use in various indoor and outdoor environments.

A key focus of the project was ensuring connectivity and real-time data transmission. The SIM7600A-H 4G/GPS module enabled seamless cellular connectivity and precise location tracking. The power management system, centered around the LTC4056 buck module, ensured safe, stable, and continuous operation, making the system practical for extended use.

In conclusion, the project successfully demonstrated how modern technology can be leveraged to create interactive and educational tools. By providing a range of advanced components and focusing on user-friendly design, the project completion has developed a system that not only succeeds in its educational purpose but also showcases potential applications in educational fields.

6. Acknowledgements

We would like to extend sincere gratitude to Texas Tech University's Electrical and Computer Engineering Department for providing the resources and facilities necessary for the research and development. The support from the department enabled us to access the tools and equipment essential for the project. We would also like to stretch this gratitude out to our instructor, Mark Haustein, for the fortuitous guidance and encouragement throughout the project. The insights were instrumental in navigating the challenges and achieving the project goals.

Additionally, we are grateful to the online communities and forums, found and appropriated in References, for providing platforms to seek advice and share knowledge. The insights and solutions offered by these communities were treasured in addressing technical issues and enhancing our understanding within the project.

Furthermore, we would like to express thanks to the Texas Tech University Library's Makers Team for printing the 3D housing for this project. Their expertise and assistance were crucial in creating the durable and portable enclosure modeled for the complete system.

Finally, we would like to thank our families and friends for their sedulous support and endearment. Their belief in our capabilities motivated us to persevere and conclude the project successfully.

7. Safety, Public Health, & Welfare Considerations

The development and deployment of the “Treasure X” project, involving the use of portable microcontrollers, GPS, and Bluetooth technologies, necessitates a comprehensive evaluation of its implications on safety, public health, and welfare. In conclusion, while the project represents a significant advancement in interactive educational tools, it also brings forth responsibilities. Ensuring safety, public health, and welfare in its design and operation ensures a balanced approach, aligning technological advancements with human-centric considerations.

7.1 Safety Considerations

Physical Safety: The portable nature of the system means it could be used in various environments, including outdoor areas. The 3D-printed enclosure provides structural integrity, protecting the internal components from physical damage during use and transportation. Additionally, the enclosure includes ventilation openings to prevent overheating of the electronics, ensuring safe operation over extended periods.

Electrical Safety: The power management system, centered around the LTC4056 module and HiLetgo buck converter, ensures safe, stable voltage and current supply to the Raspberry Pi and its attached components. This setup prevents electrical faults such as overvoltage or overcurrent, which could damage the hardware or pose a safety risk to users. All electrical connections are validated to prevent accidental overcurrent for circuits.

Operational Safety: The software includes error handling mechanisms to ensure the system can operate safely, even in the event of unexpected conditions. For instance, the Bluetooth scanning module and GPS data retrieval scripts have built-in checks to handle data inconsistencies or signal losses, providing feedback to users through the GUI. This ensures that users are informed of any issues and can take appropriate actions necessary.

User Interaction Safety: The touch display interface is provided as user-friendly and intuitive during operation. Clear instructions and real-time feedback guide users through the treasure hunt process, reducing the likelihood of confusion or incorrect usage. The trivia question feature also adds engagement, promoting thoughtful interaction with the system.

7.2 Public Health Considerations

Air Quality Health: The system is designed to be environmentally friendly, with a rechargeable battery that minimizes the need for disposable batteries and reduces waste. Additionally, the electronics are housed in a 3D-printed enclosure made from eco-friendly filament. The system's low power consumption also contributes to reduced energy use, which can help lower the overall carbon footprint.

Noise Pollution Health: The operation of the “Treasure X” system is effectually silent, with no fans or moving mechanical parts that generate noise. This makes it suitable for use in quiet environments, such as classrooms and libraries, without causing any disturbance. The low noise level also ensures that the system does not contribute to noise pollution in outdoor settings.

Ergonomics & User Comfort Health: The touch display interface is designed with user ergonomics in mind, providing an intuitive way to interact with the system. The interface layout is clear and lightweight to navigate, reducing the strain on users' eyes and hands. Additionally, the system's portability allows users to position it at a comfortable height and angle, promoting better posture and reducing physical discomfort during sit or standing use.

Educational Health:

By incorporating trivia questions and interactive elements, the project promotes cognitive engagement. This approach not only makes learning more enjoyable but also encourages thinking

and problem-solving skills. The system's educational focus aligns with public health goals by fostering intellectual development and learning habits.

7.3 Welfare Considerations

Job Implication Welfare: While the project automates certain tasks, it is designed to complement and enhance human activities rather than replace them. By serving as an educational tool, this opens opportunities for educators to incorporate technology into their lectures and methods for learners to gain intended skills. This positive impact on education and skill development supports the welfare of the workforce.

Educational Welfare: The intention of the project is to create an educational tool that enhances learning experiences. By incorporating interactive elements such as Bluetooth beacon tracking, GPS data retrieval, and trivia questions, the system promotes active learning and helps users develop critical thinking and problem-solving skills. This educational approach contributes to the long-term welfare of students and learners.

Community Welfare: The project has the potential to foster community engagement by encouraging group activities and collaborative learning. The treasure hunt aspect of the system can be used in team-building exercises, educational events, and community programs, promoting social interaction and cooperation among participants. This community-focused approach enhances social welfare, bringing people together and creating opportunities for a shared attempt.

Sustainability Welfare: The project's emphasis on sustainability using rechargeable batteries and eco-friendly materials aligns with broader welfare goals of environmental stewardship. By reducing waste and promoting responsible use of resources, the project contributes to the welfare of the environment and community.

8. Global, Environmental, & Economic Factors

The “Treasure X” project stands at the confluence of global considerations, environmental impacts, and economic ramifications. This section examines these interconnected factors, highlighting the broader implications of the project.

In conclusion, the “Treasure X” project not only provides a cutting-edge educational tool but also demonstrates a thoughtful approach to global, environmental, and economic considerations. By addressing these factors, the project sets a positive example of how technology can be integrated responsibly and sustainably into educational settings.

8.1 Global Factors

Global Interoperability: By utilizing widely recognized and standardized technologies such as the Raspberry Pi, Bluetooth Low Energy (BLE), and GPS modules, the project ensures compatibility and interoperability with other systems and devices. This adherence to global standards facilitates the integration of “Treasure X” into various educational and technological ecosystems worldwide, promoting a uniform approach to interactive learning tools.

Global Education & Cross-Cultural Equity: The project is designed to be a versatile educational tool that can be adapted to various cultural contexts. The trivia questions and interactive elements can be customized to reflect different cultural knowledge and local educational standards, making the system relevant for users worldwide. This adaptability enhances the potential impact on global education.

Global Supply Chain & Component Sourcing: The components used in the project are sourced from global suppliers. This reliance on an international supply chain highlights the interconnected nature of modern technology development and the importance of maintaining reliable supply chains.

Global Collaboration: The development of “Treasure X” involved leveraging knowledge and resources from global online communities, including forums and collaborative platforms like Stack Exchange and the Raspberry Pi forums. These communities provided valuable insights and support, highlighting the importance of global collaboration in technological innovation. This collaborative approach underscores the international network of developers and educators.

8.2 Environmental Factors

Environmental Energy-Consumption: The system is designed to operate efficiently, with an emphasis on minimizing energy consumption. The use of rechargeable batteries and low-power components helps reduce the system's carbon footprint. Exploring the integration of renewable energy sources, such as solar panels, may enhance the project's environmental sustainability.

Environmental Sustainability: The 3D-printed enclosure for the system is made from eco-friendly materials, such as biodegradable PLA (polylactic acid) filament. This choice of material reduces the environmental impact of the project by minimizing plastic waste and promoting the use of sustainable resources. The custom-designed enclosure not only provides durability and protection for the hardware components but also aligns with environmental sustainability.

Environmental Waste Management: The project emphasizes the recycling of electronic components. By using a rechargeable battery, the system reduces the need for disposable batteries, which are a significant source of electronic waste. Additionally, the project promotes the responsible disposal of electronic components at the end of their lifecycle.

Environmental Awareness: Through its educational and interactive design, the project can be used to teach students about environmental sustainability and the importance of responsible technology use. The inclusion of trivia questions and interactive elements related to environmental topics can raise awareness and encourage environmentally friendly practices among users.

8.3 Economic Factors

Economic Expenditure: Tracking economic factors is pivotal for the success and feasibility of the project. As noted, economic factors were diligently tracked daily into an excel budget sheet, as seen in Appendix A.

Economic Training: The project provides opportunities for educators and students to develop valuable technical skills, such as programming, hardware integration, and data analysis. By enhancing the technical capabilities of its users, the project contributes to workforce development and economic growth. Educators trained in using the system can also share their knowledge with others, further amplifying the project's economic benefits.

Long-Term Economic Benefits: The knowledge and skills gained with the system can have long-term economic benefits. Students who are exposed to advanced technologies and interactive learning methods are better prepared for careers in science, technology, engineering, and mathematics (STEM) fields. This preparation can lead to higher earning potential and contribute to the overall economic prosperity of the community.

Economic Scalability & Replicability: The design and implementation of the project are scalable and replicable, allowing for widespread adoption in different educational settings. The system can be easily reproduced and adapted to meet the specific needs of various schools and institutions for economic efficiency.

References

1. Autodesk. "Raspberry Pi 4 Enclosure Design," Available at: https://www.autodesk.com/community/gallery/project/150284/raspberry-pi-4-enclosure-design
2. Raspberry Pi Stack Exchange. "SIM7600 as RNDIS Interface for Raspberry Pi," Available at: https://raspberrypi.stackexchange.com/questions/121873/sim7600-as-rndis-interface-for-raspberry-pi
3. Wikipedia. "RNDIS," Available at: https://en.wikipedia.org/wiki/RNDIS
4. Wikipedia. "Geometric Median," Available at: https://en.wikipedia.org/wiki/Geometric_median
5. Raspberry Pi Forums. "Forum Discussion on Raspberry Pi," Available at: https://forums.raspberrypi.com/viewtopic.php?t=250041
6. Raspberry Pi Forums. "Forum Discussion on Raspberry Pi," Available at: https://forums.raspberrypi.com/viewtopic.php?t=224355
7. EMAC Wiki. "Getting Started With Minicom," Available at: https://wiki.emacinc.com/wiki/Getting_Started_With_Minicom
8. Twilio. "Introduction to Modem AT Commands," Available at: https://www.twilio.com/docs/iot/supersim/introduction-to-modem-at-command
9. Waveshare. "SIM7500_SIM7600 Series AT Command Manual V3.00," Available at:https://www.waveshare.com/w/upload/a/af/SIM7500_SIM7600_Series_AT_Command_Manual_V3.00.pdf

10. ResearchGate. "Distance vs RSSI Curve Fit by Plotting Various Points," Available at: https://www.researchgate.net/figure/Distance-vs-RSSI-Curve-Fit-by-Plotting-Various-Points_fig1_322877438
11. Mouser. "LM2596T-5.0," Available at: https://www.mouser.com/ProductDetail/Texas-Instruments/LM2596T-5.0?qs=X1J7HmVL2ZHGhhjJIS7EQw%3D%3D
12. SIMCom. "SIM7600X," Available at: https://www.simcom.com/product/SIM7600X.html%E2%80%8B
13. Core Electronics. "Raspberry Pi 4G GPS HAT," Available at: https://core-electronics.com.au/guides/raspberry-pi/raspberry-pi-4g-gps-hat/
14. Maker Pro. "How to Read GPS Data with Python on a Raspberry Pi," Available at: https://maker.pro/raspberry-pi/tutorial/how-to-read-gps-data-with-python-on-a-raspberry-pi
15. Hackaday. "All About USB-C Resistors and Emarkers," Available at: https://hackaday.com/2023/01/04/all-about-usb-c-resistors-and-emarkers/

Appendix A

Excel Budget

This section provides an overview of the comprehensive budgeting process employed from the project's initiation on May 30, 2024, to its conclusion on July 25, 2024.

The total of TDL, TCL, TDM, and TRM, incorporating business overhead costs, resulted in the Managed Total Cost. This figure, totaling $18,481.48, encompasses all direct and indirect expenses associated with the project. This managed total was compared against the initial estimate of $18,904.48. The deviation underscores the project's efficiency in cost management, resulting in exactly $423 of expenditure left when compared to the estimated projection cost.

Project Lab B - ECE 3332-301	***Managed Total***			***Total Estimate***			**Start Date**	5/30/2024	
Direct Labor:							**Today**	7/25/2024	
Category:	*Rate/Hr*	*Hrs*		*Rate/Hr*	*Hrs*				
Tarek Darwiche	15	95	$1,425.00	15	96	$1,440.00	**End Date**	7/25/2024	
Dylan Pitcher	15	95	$1,425.00	15	96	$1,440.00			
Christian Maldonado	15	95	$1,425.00	15	96	$1,440.00			
Jeffrey Saied	15	96	$1,440.00	15	96	$1,440.00			
DL Subtotal (DL)		Subtotal:			Subtotal:				
Labor Overhead	*rate:*	100%	$5,715.00	*rate:*	100%	$5,760.00			
Total Direct Labor (TDL)			$11,430.00			$11,520.00			
Contact Labor:									
Student(s)	15	0	$0.00	15	16	$240.00			
Total Contract Labor (TCL)			$0.00			$240.00			
Direct Material Costs:			$194.53			$100.00			
(from Material Cost worksheet)									
Total Direct Material Cost: (TDM)			$194.53			$100.00			
Equipment Rental Cost:	Value	Rental Rate		Value	Rental Rate		Date begin	Date end/today	Total rental days
Raspberry Pi 4 Model B	$45.00	0.20%	$4.68	$45.00	0.20%	$4.68	6/3/2024	7/25/2024	52
Waveform Generator	$1,112.00	0.20%	$115.65	$1,112.00	0.20%	$115.65	6/3/2024	7/25/2024	52
Digital Storage Oscilloscope	$999.00	0.20%	$103.90	$999.00	0.20%	$103.90	6/3/2024	7/25/2024	52
Multimeter	$844.00	0.20%	$87.78	$844.00	0.20%	$87.78	6/3/2024	7/25/2024	52
SIM7600A-H 4G HAT	$53.99	0.20%	$4.65	$53.99	0.20%	$4.65	6/12/2024	7/25/2024	43
iBKS 105 BT Beacons	$179.70	0.20%	$17.26	$179.70	0.20%	$17.26	6/7/2024	7/25/2024	48
HDMI + USB Touch Display	$35.99	0.20%	$2.52	$35.99	0.20%	$2.52	6/20/2024	7/25/2024	35
Total Rental Costs: (TRM)			$336.44			$336.44			
Total TDL+TCL+TDM+TRM			$11,960.97			$12,196.44			
Business overhead		55%	$6,578.54		55%	$6,708.04			
Total Cost:		**Current**	$18,539.51		**Estimate**	$18,904.48			

SUMMARY:	
Labor + OH	$11,430.00
Contract Labor	$0.00
Materials & Equip Rental	$530.97
Overhead	$6,578.54
TOTAL	$18,539.51

Appendix B

Excel Bill of Materials

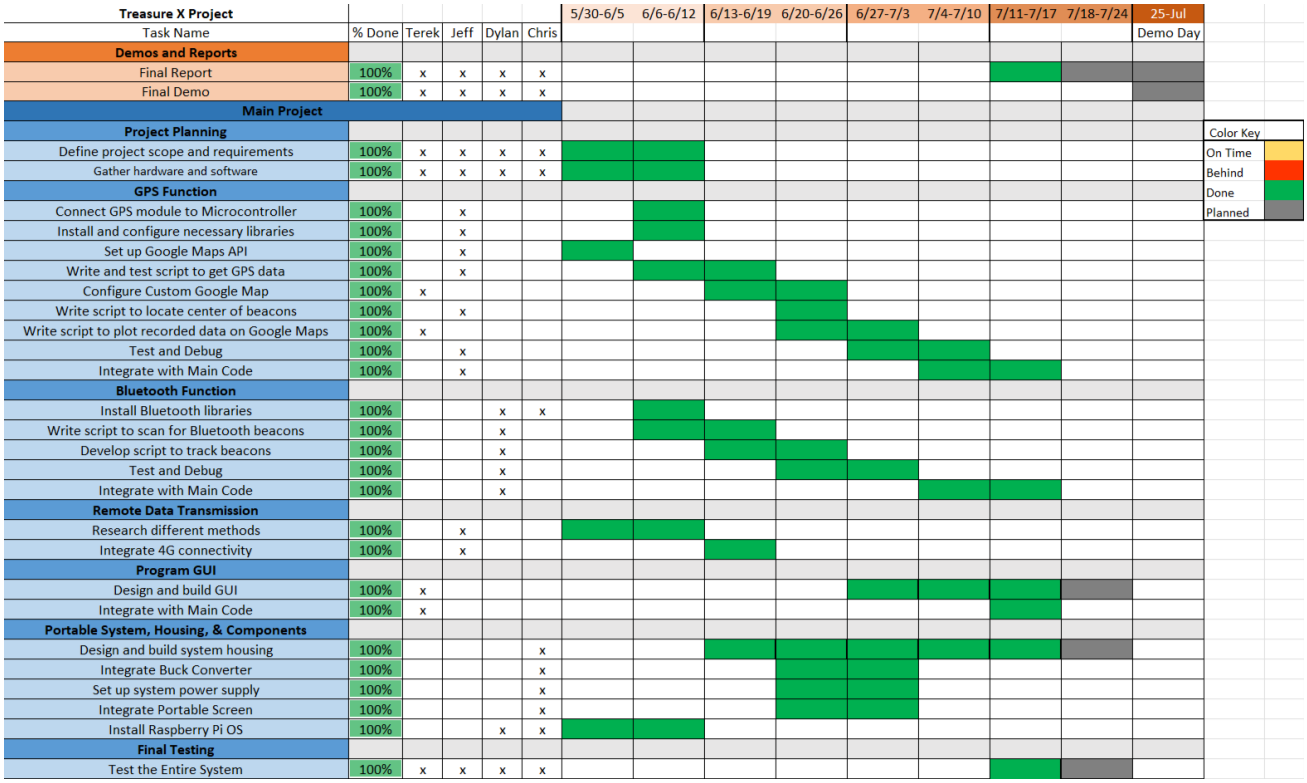
The Bill of Materials (BOM) reflects a detailed inventory of components necessary for the successful execution of the project. Notably, the BOM total expense stands at less than $100, excluding the beacons, a figure that aligns with the meticulous planning and procurement strategies implemented. The careful selection of parts and suppliers has ensured that the project remains within the economic scope, demonstrating prudence and resourcefulness in the allocation of resources. This BOM is integral to the Managed Total Cost, contributing directly to the project's overarching financial framework.

Name	Amount
Raspberry Pi 4 Model B (4GB)	1
32GB MicroSD Card (Preinstalled RPi OS)	1
HDMI + USB Touch Display (Bookworked OSK)	1
NanoHDMI to HDMI Adapter	1
HDMI Cable	1
USB to USB-C Cable	1
MicroUSB to USB Cable	1
SIM7600A-H 4G HAT Module (Total Components)	1
SIM Card (Preordered 4G Network)	1
LTC4056 Step Down Power Module (Total Components)	1
12V 5.2A Rechargable Battery + Cable I/O Connector	1
iBKS 105 Bluetooth Beacon	8
CR2477 3V Li-Ion Battery	8
3D Printed Top Model Casing	1
3D Printed Bottom Model Casing	1

Appendix C

Gantt Chart

The dynamic Gantt chart, initiated on May 30, 2024, and set to conclude on July 25, 2024, proved invaluable for flexible project management. This continuously updated chart allowed real-time progress tracking, efficient resource allocation, and bottleneck mitigation.



Treasure X Project						5/30-6/5	6/6-6/12	6/13-6/19	6/20-6/26	6/27-7/3	7/4-7/10	7/11-7/17	7/18-7/24	25-Jul
Task Name	% Done	Terek	Jeff	Dylan	Chris									Demo Day
Demos and Reports														
Final Report	100%	x	x	x	x									
Final Demo	100%	x	x	x	x									
Main Project														
Project Planning														
Define project scope and requirements	100%	x	x	x	x									
Gather hardware and software	100%	x	x	x	x									
GPS Function														
Connect GPS module to Microcontroller	100%		x											
Install and configure necessary libraries	100%		x											
Set up Google Maps API	100%		x											
Write and test script to get GPS data	100%		x											
Configure Custom Google Map	100%	x												
Write script to locate center of beacons	100%		x											
Write script to plot recorded data on Google Maps	100%	x												
Test and Debug	100%		x											
Integrate with Main Code	100%		x											
Bluetooth Function														
Install Bluetooth libraries	100%			x	x									
Write script to scan for Bluetooth beacons	100%			x										
Develop script to track beacons	100%			x										
Test and Debug	100%			x										
Integrate with Main Code	100%			x										
Remote Data Transmission														
Research different methods	100%		x											
Integrate 4G connectivity	100%		x											
Program GUI														
Design and build GUI	100%	x												
Integrate with Main Code	100%	x												
Portable System, Housing, & Components														
Design and build system housing	100%				x									
Integrate Buck Converter	100%				x									
Set up system power supply	100%				x									
Integrate Portable Screen	100%				x									
Install Raspberry Pi OS	100%			x	x									
Final Testing														
Test the Entire System	100%	x	x	x	x									

Color Key
On Time
Behind
Done
Planned

Appendix D

Code

- *main.py*

```
# CoordinateApp Code
class CoordinateApp:import tkinter as tk
from tkinter import messagebox
import random
import time
from readGPS import get_gps_position, power_on, power_down
from centerplot import calculate_center
from BLEScanner import scan_for_beacons
# Assuming power_key is defined in readGPS.py
power_key = 6
# List of multiple-choice trivia questions and their options
trivia_questions = [
    ("What is the capital of France?", ["Paris", "Berlin", "Madrid", "Rome"], "Paris"),
    ("What is the largest planet in our solar system?", ["Earth", "Mars", "Jupiter", "Saturn"],
"Jupiter"),
    ("Who wrote 'To Kill a Mockingbird'?", ["Mark Twain", "Harper Lee", "Ernest Hemingway", "F.
Scott Fitzgerald"], "Harper Lee"),
    ("What is the smallest prime number?", ["1", "2", "3", "5"], "2"),
    ("What year did the Titanic sink?", ["1912", "1915", "1920", "1908"], "1912"),
    ("What is the chemical symbol for gold?", ["Ag", "Au", "Pb", "Fe"], "Au"),
    ("Who painted the Mona Lisa?", ["Vincent van Gogh", "Pablo Picasso", "Claude Monet",
"Leonardo da Vinci"], "Leonardo da Vinci"),
    ("What is the tallest mountain in the world?", ["K2", "Kangchenjunga", "Lhotse", "Mount
Everest"], "Mount Everest"),
    ("Who developed the theory of relativity?", ["Isaac Newton", "Albert Einstein", "Niels
Bohr", "Galileo Galilei"], "Albert Einstein"),
    ("What is the hardest natural substance on Earth?", ["Gold", "Iron", "Diamond",
"Platinum"], "Diamond"),
    ("Who was the first person to walk on the moon?", ["Buzz Aldrin", "Michael Collins", "Yuri
Gagarin", "Neil Armstrong"], "Neil Armstrong"),
    ("What is the longest river in the world?", ["Amazon", "Nile", "Yangtze", "Mississippi"],
"Nile"),
    ("What is the main ingredient in guacamole?", ["Tomato", "Onion", "Lime", "Avocado"],
"Avocado"),
    ("What is the capital of Japan?", ["Beijing", "Seoul", "Tokyo", "Bangkok"], "Tokyo"),
    ("What is the largest ocean on Earth?", ["Atlantic", "Indian", "Arctic", "Pacific"],
"Pacific")
]
class CoordinateApp:
    def __init__(self, root):
        self.root = root
        self.gps_coordinates = []
        self.beacons_found = 0
        self.lives = 3
        self.expected_beacons = 2  # Adjust this to the number of beacons you expect
        self.last_rssi = None
        self.root.title("Beacon Finder")
        self.root.geometry("800x600")
        # Power on the GPS module at the start
        power_on(power_key)
        self.setup_ui()
        # Ensure GPS module is powered down when the app is closed
        self.root.protocol("WM_DELETE_WINDOW", self.on_closing)
    def setup_ui(self):
        self.label = tk.Label(self.root, text=f"Scan for Beacons. Found:
{self.beacons_found}/{self.expected_beacons}")
```

```
 54.         self.label.pack(pady=10)
 56.         self.scan_button = tk.Button(self.root, text="Scan for Beacon",
command=self.scan_for_beacon)
 57.         self.scan_button.pack(pady=10)
 59.         self.hint_label = tk.Label(self.root, text="", bg="lightyellow", font=("Helvetica", 12,
"bold"))
 60.         self.hint_label.place(relx=1.0, rely=0.0, anchor="ne", padx=10, pady=10)
 62.         self.trivia_label = tk.Label(self.root, text="", wraplength=500)
 63.         self.trivia_label.pack(pady=10)
 65.         self.answer_var = tk.StringVar()
 66.         self.option_buttons = []
 67.         for i in range(4):
 68.             rb = tk.Radiobutton(self.root, variable=self.answer_var, value="", text="")
 69.             rb.pack(pady=5)
 70.             self.option_buttons.append(rb)
 72.         self.trivia_submit_button = tk.Button(self.root, text="Submit Answer",
command=self.check_trivia_answer)
 73.         self.trivia_submit_button.pack(pady=10)
 75.         self.lives_label = tk.Label(self.root, text=f"Lives: {self.lives}")
 76.         self.lives_label.pack(pady=10)
 78.         self.load_trivia_question()
 80.     def load_trivia_question(self):
 81.         question, options, self.current_answer = random.choice(trivia_questions)
 82.         self.trivia_label.config(text=question)
 83.         random.shuffle(options)
 84.         self.answer_var.set("")
 85.         for rb, option in zip(self.option_buttons, options):
 86.             rb.config(text=option, value=option)
 87.         self.trivia_submit_button.config(state=tk.NORMAL)
 89.     def scan_for_beacon(self):
 90.         def gui_callback(message):
 91.             self.hint_label.config(text=message)  # Update the hint label
 93.         try:
 94.             avg_rssi = scan_for_beacons(gui_callback)
 95.             if avg_rssi is not None:
 96.                 if self.last_rssi is not None:
 97.                     if avg_rssi < self.last_rssi:
 98.                         self.hint_label.config(text="Getting colder...")
 99.                     else:
100.                         self.hint_label.config(text="Getting hotter...")
101.                 self.last_rssi = avg_rssi
103.                 gps_data = get_gps_position()
104.                 if gps_data:
105.                     self.gps_coordinates.append(gps_data)
106.                     self.beacons_found += 1
107.                     self.label.config(text=f"Scan for Beacons. Found:
{self.beacons_found}/{self.expected_beacons}")
108.                     print(f"GPS location {len(self.gps_coordinates)}: {gps_data}")
109.                     self.scan_button.config(state=tk.DISABLED)
110.                     print(f"Beacon {self.beacons_found} found. Please answer the trivia
question.")
111.                 else:
112.                     messagebox.showerror("GPS Error", "Failed to get GPS position.")
113.             else:
114.                 messagebox.showerror("Beacon Error", "No beacon found.")
115.         except Exception as e:
116.             messagebox.showerror("Error", f"An error occurred: {e}")
117.     def check_trivia_answer(self):
118.         if self.answer_var.get() == self.current_answer:
119.             print(f"Beacon {self.beacons_found} confirmed.")
120.             if self.beacons_found < self.expected_beacons:
121.                 self.scan_button.config(state=tk.NORMAL)
122.                 self.load_trivia_question()
123.             else:
124.                 center_location = calculate_center(self.gps_coordinates)
```

```
                messagebox.showinfo("Center Location", f"Center location (treasure):
{center_location}")
                print("Output HTML file generated by centerplot.")
        else:
            self.lives -= 1
            self.lives_label.config(text=f"Lives: {self.lives}")
            if self.lives == 0:
                messagebox.showerror("Game Over", "You have no lives left. The game will
reset.")
                self.reset_game()
            else:
                messagebox.showerror("Incorrect Answer", "Please try again.")
    def reset_game(self):
        self.gps_coordinates = []
        self.beacons_found = 0
        self.lives = 3
        self.label.config(text=f"Scan for Beacons. Found:
{self.beacons_found}/{self.expected_beacons}")
        self.lives_label.config(text=f"Lives: {self.lives}")
        self.scan_button.config(state=tk.NORMAL)
        self.load_trivia_question()

    def on_closing(self):
        power_down(power_key)
        self.root.destroy()
def main_screen():
    root = tk.Tk()
    app = CoordinateApp(root)
    root.mainloop()
def about_screen(welcome_root):
    about_root = tk.Toplevel(welcome_root)
    about_root.title("About")
    about_root.geometry("800x600")  # Set the size of the window to 800x600
    # Add a wallpaper background
    about_wallpaper = tk.PhotoImage(file="about_background.png")
    about_background_label = tk.Label(about_root, image=about_wallpaper)
    about_background_label.place(relwidth=1, relheight=1)
    # About screen layout
    about_text = """
    This is an app for finding beacons and calculating the center of their GPS locations.
    You need to find 2 beacons (adjustable) and answer trivia questions to confirm their
locations.
    """
    about_label = tk.Label(about_root, text=about_text, bg="white", font=("Helvetica", 12),
justify="left")
    about_label.pack(pady=100)

    back_button = tk.Button(about_root, text="Back", command=about_root.destroy)
    back_button.pack(pady=20)
    # Keep the image reference to avoid garbage collection
    about_background_label.image = about_wallpaper
    about_root.mainloop()
def welcome_screen():
    welcome_root = tk.Tk()
    welcome_root.title("Welcome")
    welcome_root.geometry("800x600")  # Set the size of the window to 800x600
    welcome_root.eval('tk::PlaceWindow . center')  # Center the window
    # Add a wallpaper background
    wallpaper = tk.PhotoImage(file="background.png")
    background_label = tk.Label(welcome_root, image=wallpaper)
    background_label.place(relwidth=1, relheight=1)
    # Welcome screen layout
    welcome_label = tk.Label(welcome_root, text="Welcome to the Beacon Finder App", bg="white",
font=("Helvetica", 16))
    welcome_label.pack(pady=20)
```

```
    start_button = tk.Button(welcome_root, text="Start", command=lambda:
[welcome_root.destroy(), main_screen()])
    start_button.pack(pady=10)
    about_button = tk.Button(welcome_root, text="About", command=lambda:
about_screen(welcome_root))
    about_button.pack(pady=10)
    # Keep the image reference to avoid garbage collection
    background_label.image = wallpaper
    welcome_root.mainloop()
if __name__ == "__main__":
    welcome_screen()
```

- *BLEScanner.py*

```
from bluepy.btle import Scanner
# List of known beacons
beacons = ["e1:a4:47:95:63:70", "c2:7f:0a:ba:09:1d", "d6:ee:e4:3b:37:95", "f0:03:1c:3e:dc:ee",
"f9:67:58:3a:0f:6d", "f1:c2:51:76:6e:54", "de:0a:7c:3a:6b:0c", "d2:df:c1:4c:1c:79"]
def scan_for_beacons(gui_callback):
    scanner = Scanner()
    beacons_active = 2  # Number of active beacons to find
    lock = 0
    counter = 0
    rssi_tot = 0
    avg_rssi = 0
    prev_rssi = 0
    while beacons_active > 0:
        devices = scanner.scan(.5)
        for device in devices:
            if device.addr in beacons:
                print(f"DEV = {device.addr} RSSI = {device.rssi}")
                if device.rssi > -75:
                    tracked_addr = device.addr
                    print(f"Locking on to beacon with address {tracked_addr}")
                    lock = 1
                    counter = 0
                    rssi_tot = 0
                    while lock == 1:
                        devices = scanner.scan(.5)
                        for device in devices:
                            if device.addr == tracked_addr:
                                print(f"DEV = {device.addr} RSSI = {device.rssi}")
                                counter += 1
                                rssi_tot += device.rssi
                                if counter == 5:
                                    prev_rssi = avg_rssi
                                    avg_rssi = rssi_tot / 5
                                    counter = 0
                                    rssi_tot = 0
                                    print(f"The average RSSI is {avg_rssi}")
                                    if avg_rssi < prev_rssi and prev_rssi != 0:
                                        gui_callback("You are moving away from the beacon")
                                    elif avg_rssi >= prev_rssi and prev_rssi != 0:
                                        gui_callback("You are moving closer to the beacon!")
                                    if avg_rssi > -50 and avg_rssi != 0:
                                        print(f"Beacon with address {device.addr} has been
found, removing address...")
                                        if device.addr in beacons:
                                            beacons.remove(tracked_addr)
                                            tracked_addr = None
                                        counter = 0
                                        rssi_tot = 0
                                        lock = 0
                                        avg_rssi = 0
```

```
                                        beacons_active -= 1
                                        prev_rssi = 0
                                        print(f"Remaining beacons list {beacons}")
                                        return avg_rssi  # Return the average RSSI value
                                    if avg_rssi < -75 and avg_rssi != 0:
                                        print(f"Beacon with address {device.addr} is too far,
disengaging lock on")
                                        counter = 0
                                        rssi_tot = 0
                                        lock = 0
                                        avg_rssi = 0
                                        prev_rssi = 0
    return None
if __name__ == "__main__":
    found_beacon = scan_for_beacons(lambda msg: print(msg))
    if found_beacon:
        print(f"Beacon found: {found_beacon}")
    else:
        print("No beacons found.")
```

- *readGPS.py*

```
import numpy as np
from scipy.spatial.distance import cdist
import gmplot
import webbrowser
# Function to calculate geometric median
def geometric_median(X, eps=1e-5):
    y = np.mean(X, axis=0)
    while True:
        D = cdist(X, y[np.newaxis, :])
        nonzeros = (D != 0).flatten()
        D_flat = D.flatten()
        if np.sum(nonzeros) == 0:
            return y
        Dinv = 1 / D_flat[nonzeros]
        Dinvs = np.sum(Dinv)
        W = Dinv / Dinvs
        T = np.sum(W[:, np.newaxis] * X[nonzeros], axis=0)
        num_zeros = len(X) - np.sum(nonzeros)
        if num_zeros == 0:
            y1 = T
        elif num_zeros == len(X):
            return y
        else:
            R = (T - y) * Dinvs
            r = np.linalg.norm(R)
            rinv = 0 if r == 0 else num_zeros / r
            y1 = max(0, 1 - rinv) * T + min(1, rinv) * y
        if np.linalg.norm(y - y1) < eps:
            return y1
        y = y1
# Function to calculate the center of GPS coordinates and output an HTML file
def calculate_center(beacons):
    # Convert beacons to a NumPy array
    beacons_np = np.array(beacons)
    center = geometric_median(beacons_np)
    zoom_level = 18
    # Your Google Maps API key
    gmap = gmplot.GoogleMapPlotter(center[0], center[1], zoom_level,
apikey='AIzaSyArhiR79MPEGcSlfTNt0N4yZYwzukIWoNo')
    # Plot the beacons
    for lat, lon in beacons:
        gmap.marker(lat, lon, 'red')
```

```
    # Plot the center
    gmap.marker(center[0], center[1], 'blue')
    # Draw the map
    map_file = "map.html"
    gmap.draw(map_file)
    print(f"Geometric median (center) is at: {center[0]}, {center[1]}")
    # Open the HTML file in the default web browser
    webbrowser.open(map_file)
    return center
```

- *centerplot.py*

```
import numpy as np
from scipy.spatial.distance import cdist
import gmplot
# Function to calculate geometric median
def geometric_median(X, eps=1e-5):
    y = np.mean(X, axis=0)
    while True:
        D = cdist(X, y[np.newaxis, :])
        nonzeros = (D != 0).flatten()
        D_flat = D.flatten()
        if np.sum(nonzeros) == 0:
            return y
        Dinv = 1 / D_flat[nonzeros]
        Dinvs = np.sum(Dinv)
        W = Dinv / Dinvs
        T = np.sum(W[:, np.newaxis] * X[nonzeros], axis=0)
        num_zeros = len(X) - np.sum(nonzeros)
        if num_zeros == 0:
            y1 = T
        elif num_zeros == len(X):
            return y
        else:
            R = (T - y) * Dinvs
            r = np.linalg.norm(R)
            rinv = 0 if r == 0 else num_zeros / r
            y1 = max(0, 1 - rinv) * T + min(1, rinv) * y
        if np.linalg.norm(y - y1) < eps:
            return y1
        y = y1
# Function to calculate the center of GPS coordinates and output an HTML file
def calculate_center(beacons):
    # Convert beacons to a NumPy array
    beacons_np = np.array(beacons)
    center = geometric_median(beacons_np)
    zoom_level = 18
    # Your Google Maps API key
    gmap = gmplot.GoogleMapPlotter(center[0], center[1], zoom_level,
apikey='AIzaSyArhiR79MPEGcSlfTNt0N4yZYwzukIWoNo')
    # Plot the beacons
    for lat, lon in beacons:
        gmap.marker(lat, lon, 'red')
    # Plot the center
    gmap.marker(center[0], center[1], 'blue')
    # Draw the map
    gmap.draw("map.html")
    print(f"Geometric median (center) is at: {center[0]}, {center[1]}")
    return center
```

Appendix E

WRITTEN LAB REPORT EVALUATION FORM

Student Name:

Course Number:

Instructor:

Date:

Please score the student by circling one of the responses following each of the statements.

1) The student's writing style (clarity, directness, grammar, spelling, style, format, etc)

A B C D F Zero

2) The quality and level of technical content of the student's report

A B C D F Zero

3) The quality of results and conclusions

A B C D F Zero

4) Quality of measurements planned / taken

A B C D F Zero

5) Appropriate engineering standards employed

A B C D F Zero

6) Multiple realistic constraints considered

A B C D F Zero

7) Properly utilized knowledge and skills acquired in earlier course work

A B C D F Zero

Grade:

Appendix F

ORAL PRESENTATION EVALUATION FORM

Student Name:

Course Number:

Instructor:

Date:

Please score the student by circling one of the responses following each of the statements.

1) The student's apparent theoretical preparation

A B C D F

2) Quality of the student's specific design

A B C D F

3) The quality of the student's presentation

A B C D F

4) The student's ability to answer questions

A B C D F

5) The student's attitude toward the lab (initiative, ability to self-direct, team work, etc.)

A B C D F

Grade: