Sheridan College

Smart Home Delivery System with GeoFencing Integration

Ian Paolo Masesar [1], Ryoh Kuramoto [2]* and Atena Amanati Shahri [3]

[1] Internet of Things and Machine Intelligence Program, Sheridan College
[2] Internet of Things and Machine Intelligence Program, Sheridan College
[3] Professor at Sheridan College
* Correspondence: ryoh.kuramoto@rknewtech.com

Abstract: The rapid growth of online shopping has transformed the retail industry but has introduced challenges in last-mile delivery, particularly concerning security and efficiency. Traditional delivery methods often expose packages to theft or damage when left unattended. This report investigates innovative solutions integrating smart technologies into home delivery systems to address these issues. Specifically, it explores the use of ESP32 microcontrollers, Blynk cloud services, and geofencing technologies to develop a smart delivery system that secures package storage, provides real-time notifications, and improves delivery success rates. The system incorporates various hardware components, including ultrasonic sensors and servo motors, managed via a smartphone application, to enhance the reliability and convenience of package delivery. Testing and observations highlight potential improvements in hardware stability, software scalability, and system reliability. Future enhancements such as emergency unlock mechanisms, advanced location tracking, and data management are recommended to optimize the system's performance and expand its market applicability.

Keywords: ESP32; Blynk; GPS; Geofencing

1. Introduction

The rapid growth of online shopping has revolutionized the retail industry, but it has also introduced challenges in the last-mile delivery process, particularly regarding security and efficiency. Traditional delivery methods often involve packages being left unattended at doorsteps, which exposes them to risks such as theft or damage. This report explores various innovative solutions designed to address these challenges, with a focus on integrating smart technologies into home delivery systems.

Recent research has highlighted several approaches to improving last-mile delivery. For example, the ParcEMon platform leverages IoT technology to monitor greenhouse gas emissions from delivery vehicles, aiming to optimize routes and reduce environmental impact [1]. By equipping vans with sensors, the platform collects real-time data on emissions influenced by vehicle type, road conditions, and driving behavior, which is then processed through edge computing and analyzed in the cloud. This initiative aims to enhance delivery efficiency while promoting sustainability.

Another significant development is the application of geofencing technology in various fields. As demonstrated in the study by Shevchenko et al., geofencing can create virtual boundaries that trigger specific actions, such as conducting surveys when participants enter or exit defined areas [2]. This technology not only conserves battery life by avoiding continuous location tracking but also enhances user privacy. The study provides insights into the effectiveness of geofencing under different conditions, offering recommendations for its broader application.

In the realm of parcel management, Ooi et al. investigated a modular parcel locker system designed to streamline last-mile deliveries [3]. This system provides centralized drop-off points where multiple logistics companies can deposit parcels, allowing

recipients to collect their packages at their convenience. The system's modularity and scalability are key advantages, accommodating the growing demand for delivery endpoints while reducing logistical costs.

Additionally, the ParcelRestBox system was developed in response to the surge in online shopping during the COVID-19 pandemic in Malaysia [4]. This system enables users to receive parcels securely even when they are not at home, notifying them upon delivery. The prototype, built using a NodeMCU V3 ESP8266 microcontroller and infrared sensors, integrates with a mobile app to provide real-time updates and enhance the overall delivery experience.

Collectively, these studies present innovative strategies for addressing the challenges of last-mile delivery and parcel management. They range from leveraging high-tech sensors for environmental monitoring to utilizing geofencing for behavior tracking, and from modular locker systems to smart home delivery solutions. The primary objective of this project is to design and implement a smart delivery system that improves the security and efficiency of package deliveries. The system aims to secure package storage, provide real-time notifications, and enhance delivery success rates, ultimately reducing package theft and ensuring a more reliable delivery process for both homeowners and delivery personnel.

2. Materials and Methods

This section details the hardware, software, and overall system architecture utilized in the project.

2.1. Hardware Components

The hardware setup for the smart delivery system includes the following components, each selected for its specific functionality and integration within the system. The detailed specifications of these components are listed in Table 1.

- ESP32-DevKitC V4: Serves as the central control unit of the system. It manages connections between various components, including the courier's phone, ultrasonic sensor, camera, servo motor, and the cloud server.
- Ultrasonic Sensor HC-SR04: Detects the proximity of the delivery person to the delivery box, triggering subsequent actions.
- Servo Motor SG90 9g: Operates the mechanism for locking and unlocking the delivery box based on signals received from the ESP32.
- LCD 1602A Module: Provides visual feedback by displaying the status of the package within the parcel storage.
- Camera Module for ESP32: Captures images of the delivery process when the delivery person approaches the parcel storage, enhancing security.
- Courier Phone: Used by the delivery person to verify their location and interact with the system.
- Owner Phone: Allows the homeowner to monitor delivery history, manage notifications, and adjust system settings via a mobile app.
- Delivery Box: A secure box where packages are stored. The box's locking mechanism is controlled by the servo motor.

Table 1. Hardware Specifications

Component Name	Specifications
ESP32-DevKitC V4	- CPU: Xtensa 32-bit LX6 dual-core - Clock Speed: Up to 240 MHz - Wireless: Wi-Fi 802.11 b/g/n, Bluetooth v4.2 BLE - Memory: 520 KB SRAM - Storage: 4 MB Flash - GPIO: 36 programmable GPIOs - Operating Voltage: 3.3V
Ultrasonic Sensor (HC-SR04)	- Power Supply: 5V DC - Measuring Range: 2cm to 400cm - Resolution: 0.3cm - Measuring Angle: 15 degrees
SG90 9g Micro Servo Motor	- Interface Type: Compatible with JR & FUTABA interface - No Load Running Speed: 0.09±0.01 sec/60° at 4.8V - Stall Torque (4.8V): 17.5oz/inch / 1kg/cm - Operating Voltage: 4.8V - 6.0V
LCD1602A Module	- Display: 2 lines x 16 characters - Backlight: Yellow-Green - Interface: Parallel (16 pins) - Operating Voltage: 5V - Character Size: 5.56 x 2.96 mm - Viewing Area: 64.5 x 16.4 mm

2.2. Software Components

The software architecture of the system comprises the following components:

- Cloud Server: Utilizes Blynk cloud services for storing and managing image data captured by the ESP32. It provides real-time interaction and data management for both the delivery personnel and the homeowner.
- Smart Home Delivery Platform: A smartphone application and online platform developed using Blynk, offering user interaction and system management functionalities for both delivery personnel and homeowners.
- Arduino Program: Manages the integration between hardware and software components. It connects to the smart home delivery platform via Wi-Fi, facilitating communication and control across the system.

2.3. System Architecture

The overall system architecture is depicted in Figure 1. Figure 2 provides a detailed overview of how the various hardware components are interconnected and communicate with each other within the system.

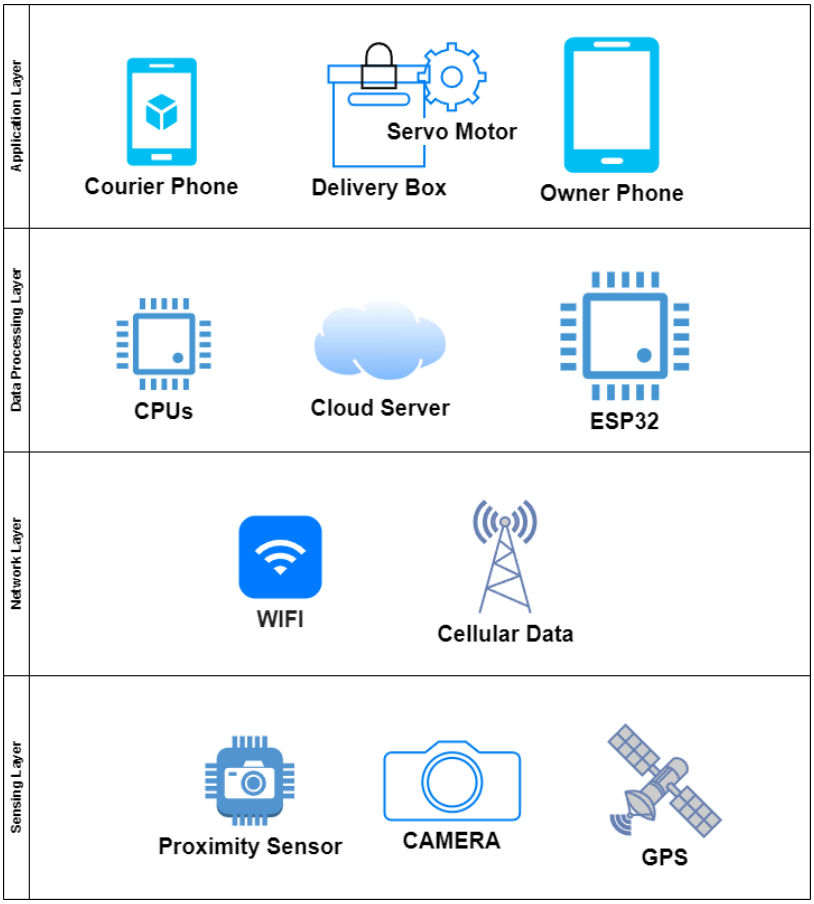

Figure 1. System Architecture

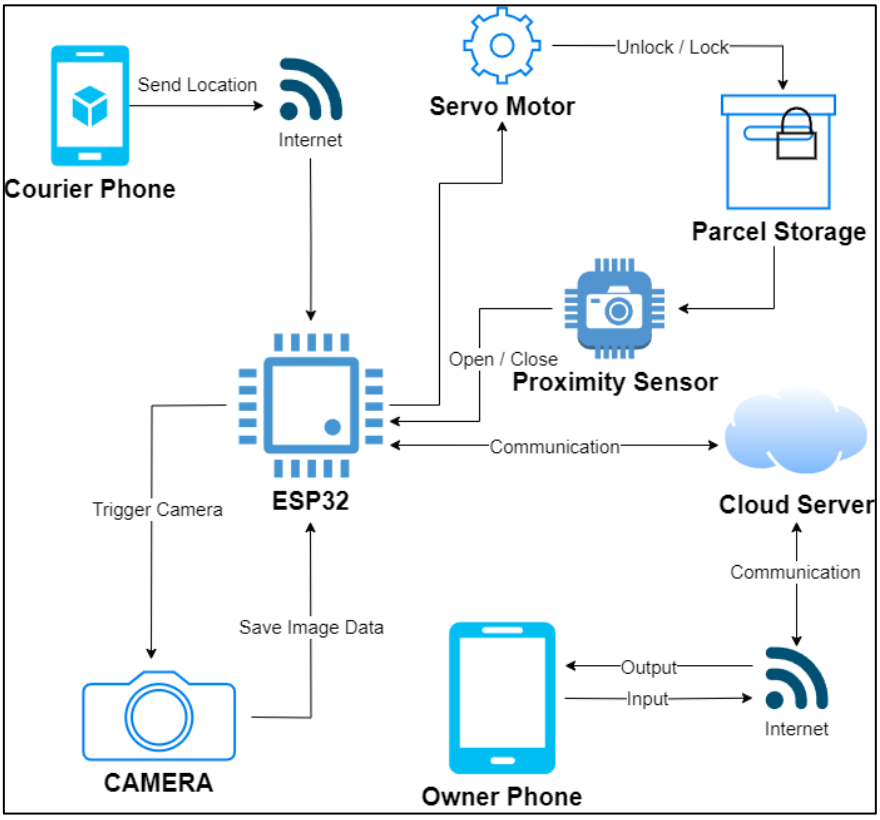

Figure 2. System Components Diagram

The physical layout of the hardware components, including the wiring and connections specific to the ESP32 module, is shown in Figure 3. Pin configurations may vary depending on the specific ESP32 model used.

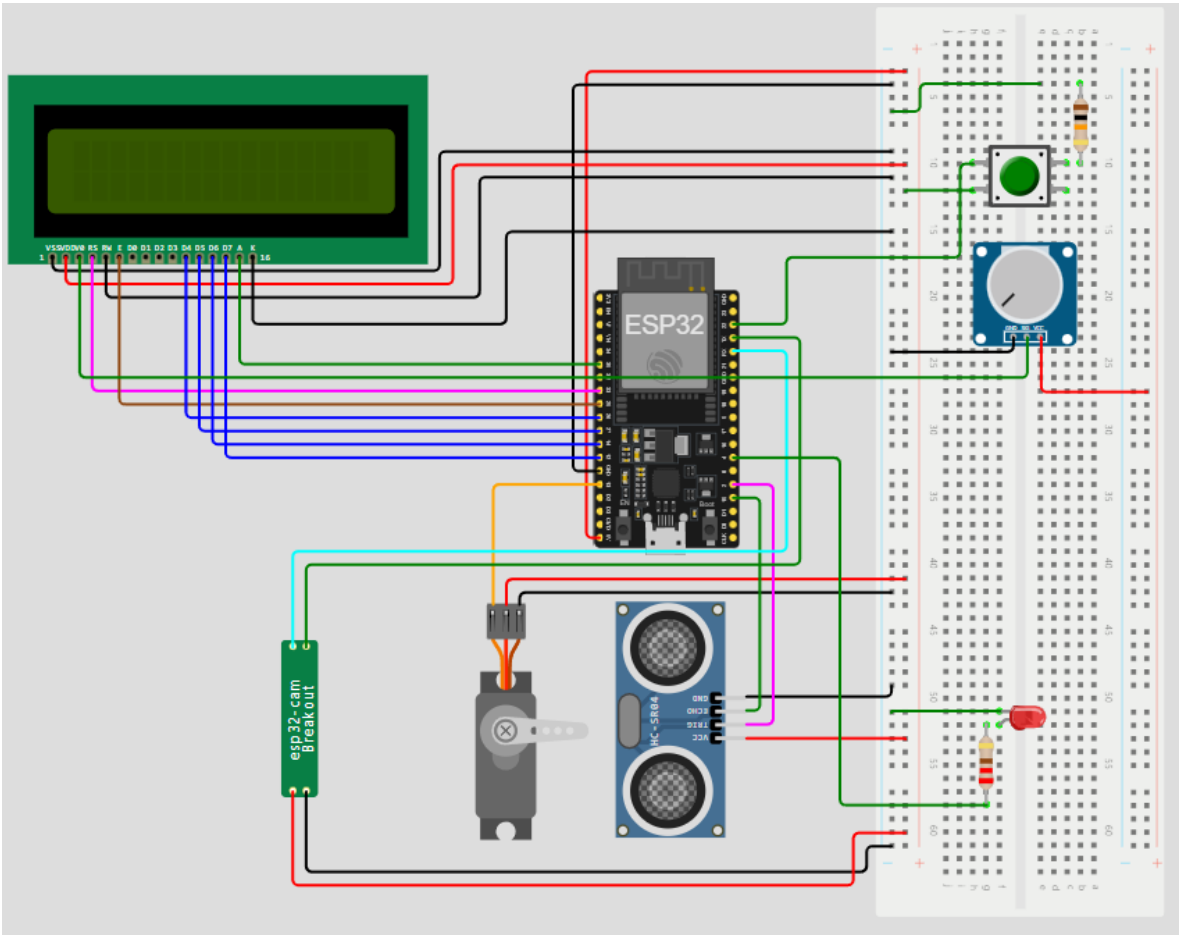

Figure 3. Hardware Layout

The sequence of operations within the system is detailed in Figure 4, highlighting the interaction between the user, delivery personnel, and the hardware components.

1. Start: Energize the ESP32. The Arduino code automatically runs.
2. Wi-Fi Connection: Check if the ESP32 is connected to Wi-Fi.
3. Initialize Homeowner App Settings: During the initial setup, all buttons in the homeowner app are set to ON by default. The user can adjust the button states as needed.
4. Delivery Person Detection: The electronic components are activated when the courier enters the geofenced area. The homeowner's location should be pre-configured in the Arduino code. The camera connects to the server and becomes available for live streaming to the user.
5. Delivery Box Unlocking: The storage box unlocks when the ultrasonic sensor detects the courier. The servo motor rotates 180 degrees to unlock the box. When rotated, it takes off the hook from the ring to allow the box to open. The status of both "Lock" and "Open" is displayed on the LCD module.
6. User Notification: Notifications are sent to the homeowner via email when the delivery box is unlocked, provided the notification button in the user app is turned on.
7. Storage Box Locking: The storage box automatically locks 10 seconds after being closed. The servo motor rotates 180 degrees towards a reverse angle to lock the box. When rotated, the hook is fixed by the ring to prevent the box from opening.

8. System Continuation: The system continues to operate until the ESP32 is de-energized.

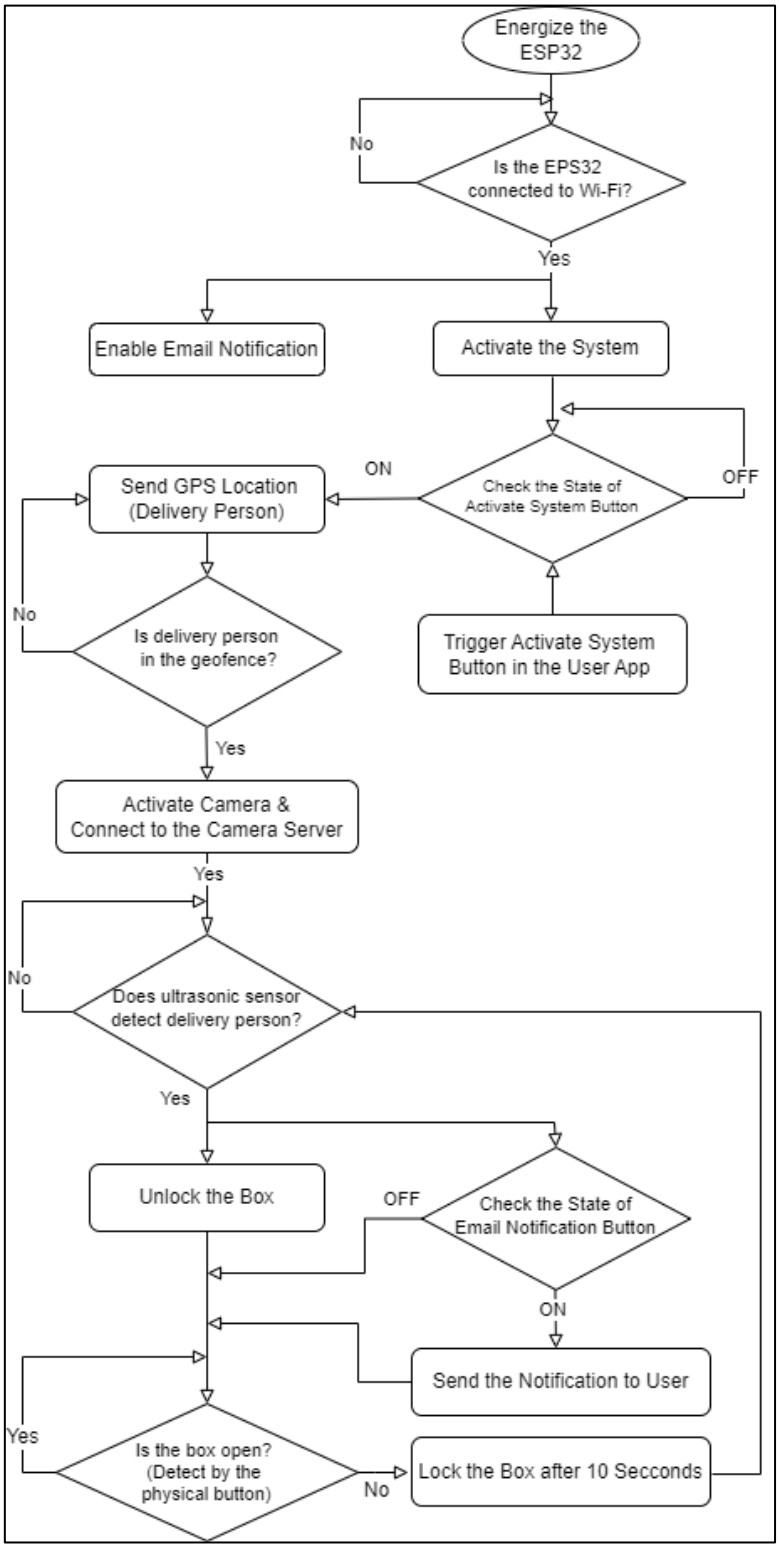

Figure 4. System Flow Diagram

3. Results

This section outlines the outcomes of the project's software and hardware development, as well as observations from system testing.

3.1. Software Development

3.1.1. User Application Development

The project involved the development of both web and smartphone applications using the Blynk platform. The web application was created using "Blynk.Console," and the smartphone application was built using the Blynk development app. Although the web and smartphone apps serve similar functions, there are minor differences in their interfaces due to platform-specific features. Key functions of the applications include:

- Uptime Label: Displays the total time the system has been in operation.
- Activate System Switch: Allows users to enable or disable the system.
- Email Notification Switch: Toggles the email notification feature on or off.

Figures 5 and 6 show the interfaces of the web and smartphone applications, respectively.

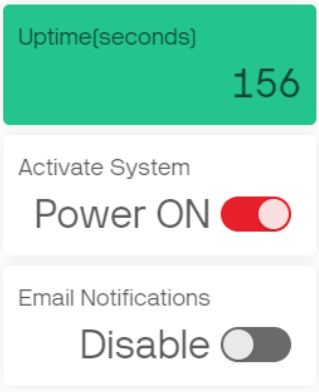

Figure 5. Web Application Interface

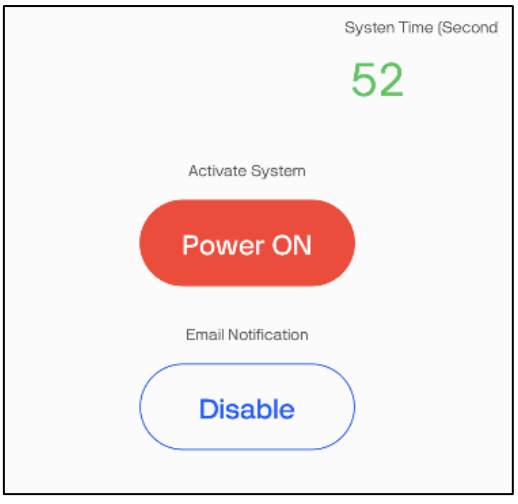

Figure 6. Smartphone Application Interface

3.1.2. Wi-Fi Provisioning

Wi-Fi provisioning [5] for the ESP32 device was implemented using Blynk's provisioning feature integrated into the Arduino code. During the initial setup, users can configure their Wi-Fi settings through the Blynk app by selecting the "Reconfigure" option and entering the necessary credentials. These settings are retained for future sessions. If a

user needs to reset the Wi-Fi settings, this can be done by pressing the reset button on the ESP32. The Wi-Fi provisioning interface is illustrated in Figure 7.

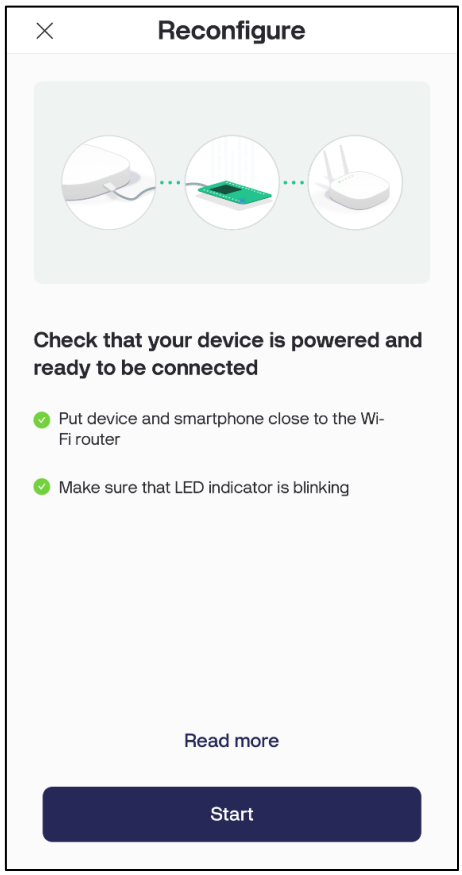

Figure 7. Wi-Fi Provisioning Interface

3.1.3. Geofencing Implementation

The geofencing function was implemented to activate system actions only when the delivery person enters the designated area around the homeowner's location. This feature helps conserve system resources by ensuring the delivery box operates only when necessary. For iPhone users, geofencing is managed through IFTTT (If This Then That), while Android users use Termux to send GPS information via HTTP requests to the Blynk platform. The specific request format is determined by Blynk and should be regularly checked for updates, as Blynk occasionally changes the format [6]. The standard format is: https://{server_address}/external/api/update?token={token}&{pin}={value}.

An example of setting up an IFTTT condition is shown in Figure 8, where pressing a button triggers a web request to update the system's status on Blynk. This button is created using the iPhone's widget function, as illustrated in Figure 9.

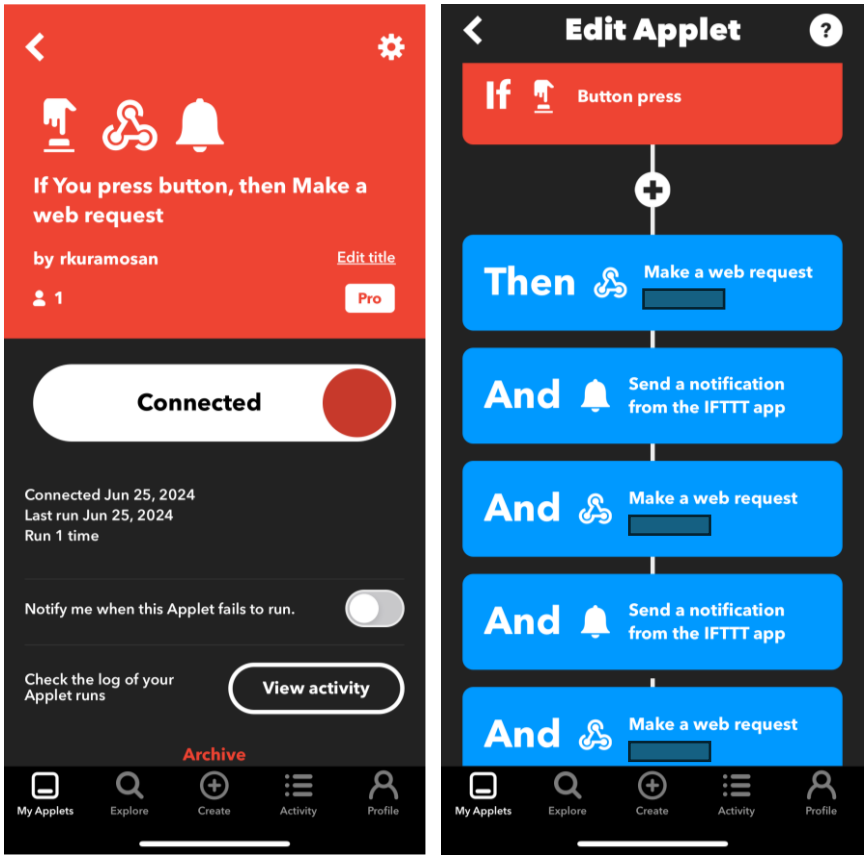

Figure 8. IFTTT Setup Interface

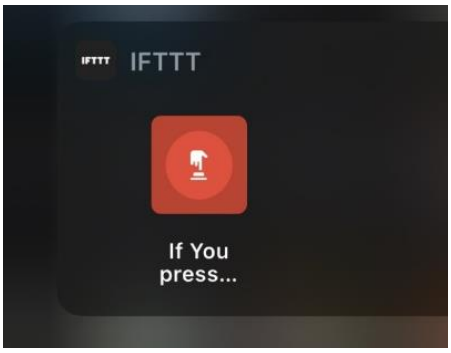

Figure 9. IFTTT Button Widget

3.1.4. E-mail Notifications

The email notification feature was developed using Blynk's notification tool, which sends emails to the homeowner based on specific events, such as the delivery box being unlocked. The system manufacturer must first create a user account on the Blynk platform and configure the email settings. The "When" and "Do This" rules in the Automation section enable automatic email notifications when predefined conditions are met. A test email sent to the user is demonstrated in Figure 10.

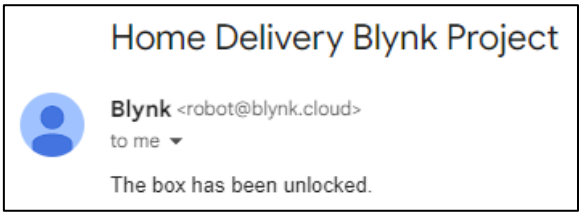

Figure 10. Test Email Screenshot

3.2. Hardware Development

The hardware development phase involved constructing a delivery box integrated with various electronic components, including an ultrasonic sensor, servo motor, camera, LCD module, and a breadboard. These components are strategically concealed under a partition within the box to maintain a clean and organized appearance while ensuring functionality.

The system is activated when the delivery person enters the geofenced area, which is determined using GPS data sent through IFTTT (If This Then That) integration. Upon entering the designated area, the LCD module displays "Lock State: Locked," signaling that the system is ready for interaction (Figures 11 and 12). Simultaneously, the camera module is activated and connected to the server, allowing real-time monitoring (Figure 13).

The camera functionality is managed using the "CameraWebServer" example code available in the Arduino IDE. This code allows for easy integration with the ESP32 and provides features such as live streaming, image capture, resolution adjustment, and face recognition [7]. Users can interact with the camera through a web server interface, ensuring enhanced security during the delivery process.

Figure 11. Sending GPS Location

Figure 12. System Locked State

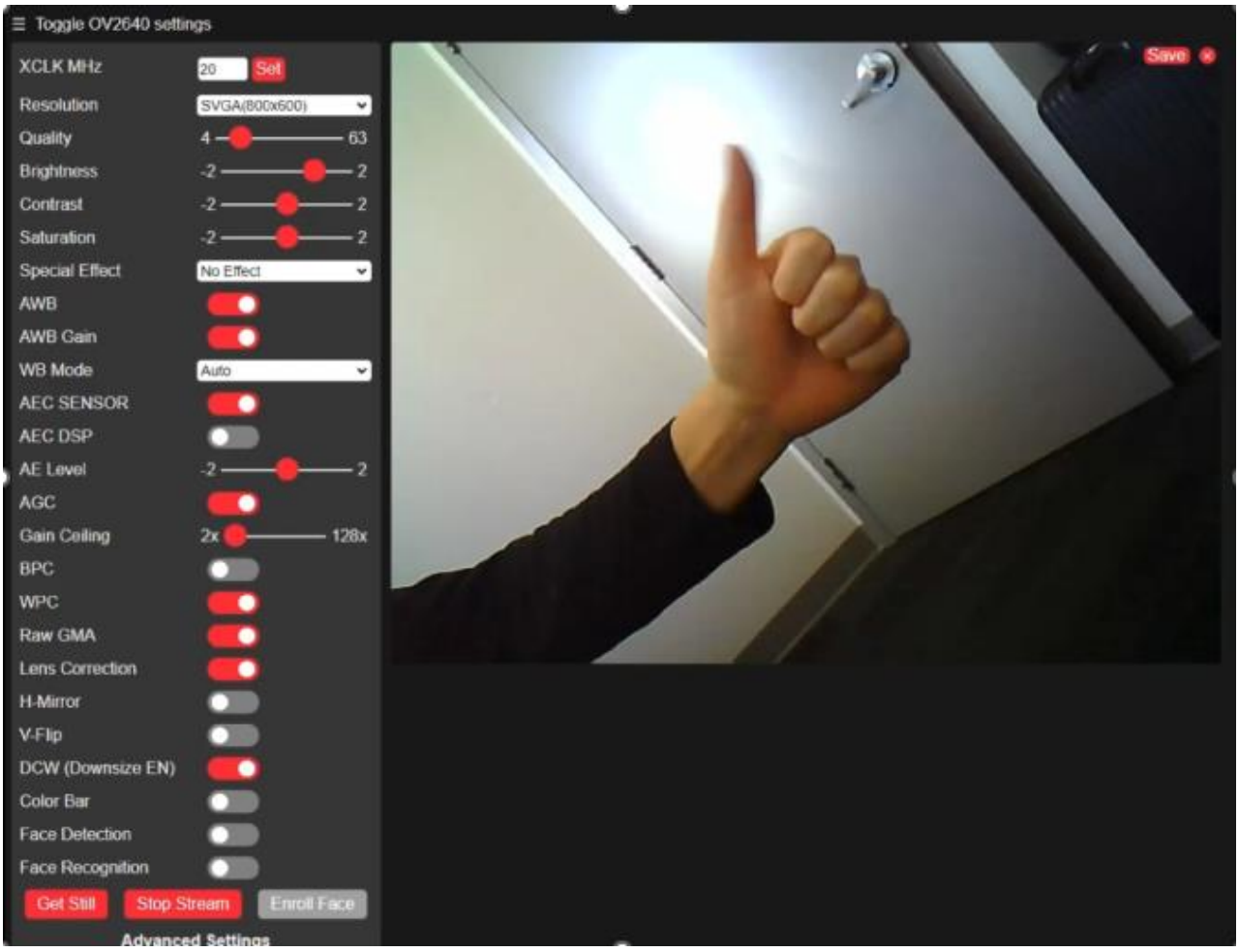

Figure 13. Camera Server Interface

Users can deactivate the system through the smartphone app by toggling the "Activation Button." This action disables the system, preventing any further interactions until reactivated by the user (Figures 14 and 15).

Figure 14. System Deactivation

Figure 15. System Activation

Upon detecting the delivery person through the ultrasonic sensor, the system automatically unlocks the delivery box by rotating the servo motor. The LCD module updates to display "Lock State: Unlocked," indicating that the box is ready for the package to be placed inside (Figures 16, 17, and 18). A physical button is used to detect whether the box is open or closed, ensuring the correct operation of the locking mechanism.

Once the item is placed inside (Figure 19), the delivery person closes the box. The system is programmed to automatically relock the box 10 seconds after closure, rotating the servo motor back to its locked position to secure the package (Figure 20).

Figure 16. Delivery Person Detection

Figure 17. Unlocking Delivery Box

Figure 18. Box Open State

Figure 19. Item Placement

Figure 20. Box Relocking

4. Discussion

The discussion section delves into the challenges, observations, and potential improvements associated with the development and deployment of the smart delivery system, categorized into hardware and software development, future enhancements, and business considerations.

4.1. Hardware Development

During the hardware development phase, several key issues were encountered that highlighted the need for robust design and implementation strategies:

- ESP32 Module Reusability Issues: The decision to reuse ESP32 modules revealed unpredictable performance issues, such as unreliable pings and connectivity problems. These issues underscore the necessity for using new or thoroughly tested components in complex systems where reliability is paramount. Troubleshooting such problems can be time-consuming and may require a systematic approach to isolate the root causes in hardware and software interactions.
- Power Supply Instability: The initial choice of a 6V battery setup, based on early voltage experiments, proved unstable in the final system configuration. This instability could lead to inconsistent operation, which is unacceptable for a product intended for regular consumer use. Future iterations of the system must include comprehensive power supply testing under all expected operational conditions to identify a more reliable power source that can consistently meet the system's demands.
- Connection Reliability: The use of long and excessive cables contributed to frequent disconnections and overall system instability. To enhance reliability, it is recommended to minimize cable lengths wherever possible and to solder connections rather than using connectors, which may become loose over time. This change could significantly reduce connection failures and improve the system's overall durability.

4.2. Software Development

As the software component of the project expanded, several challenges arose, reflecting the complexities of managing and scaling such systems:

- Codebase Management: The project's growth has led to a large and increasingly unmanageable codebase. Adopting modular coding practices and establishing a clear file management system are essential steps for maintaining and scaling the system effectively. This approach will make the code easier to troubleshoot, update, and expand in the future, which is critical for the system's long-term viability.
- GPS Accuracy Limitations: During experiments, smartphone GPS accuracy was observed to vary between 5 to 10 meters. This variability is significant when setting up

geofencing, as it could lead to incorrect triggering of the system. To mitigate this, the geofencing threshold value should be adjusted to account for GPS inaccuracies, potentially setting a buffer of 20 to 30 meters instead of the intended 10-meter radius.

- Camera Streaming Reliability: The web server camera platform used in the project requires a stable and high-speed Wi-Fi connection for smooth real-time streaming. Any delay or dropout in the connection can compromise the user experience and system reliability. Future enhancements should explore methods to optimize streaming performance, possibly by integrating buffering mechanisms or employing more efficient streaming protocols.

4.3. Future Enhancements

Looking forward, several potential enhancements could improve the system's functionality and resilience:

- Emergency Unlock Mechanism: The current system lacks a backup method for accessing the delivery box during Wi-Fi connectivity loss or power outages. Implementing an emergency unlock method, such as a physical key or a local Bluetooth connection, would ensure continued access to the box under adverse conditions, thereby enhancing the system's reliability and user confidence.
- Integration of Python for Advanced Features: Incorporating Python into the system could significantly increase its flexibility, allowing for more complex logic and broader functionality. This could include advanced data processing, more sophisticated user interfaces, and additional automation capabilities, making the system more adaptable to various user needs.
- Location Tracking for Enhanced Notifications: Implementing real-time location tracking for the delivery person could provide more accurate and timely notifications to the homeowner, thereby improving the overall delivery experience. This feature would be particularly useful in dynamic environments where the delivery times can vary.
- Image Storage and Review Capability: Enhancing the system to automatically store images captured by the camera would allow users to review footage later. This feature would provide an additional layer of security and convenience, ensuring that critical information is preserved even if the live feed is interrupted.

4.4. Business Considerations

In addition to technical enhancements, several business considerations could help commercialize the system effectively:

- Target Market: The primary market for this product includes delivery service companies looking to offer enhanced security and efficiency in their delivery processes. Positioning the system as a value-added service could attract businesses aiming to differentiate themselves in a competitive market.
- Custom PCB Design: Developing a custom PCB design could improve system stability and efficiency by embedding wiring directly into the board, reducing the risk of component failure due to loose connections or physical wear. This design shift could also streamline production and reduce assembly costs.
- Sophisticated Data Management: To handle user data more effectively, transitioning to a robust database system like MySQL could ensure secure and efficient data handling, particularly as the system scales and more users are onboarded.
- Custom Interface Development: Migrating from the Blynk platform to a custom-developed interface could enhance functionality and better meet the needs of business customers. A tailored platform could provide greater flexibility, more features, and tighter integration with existing business systems.

5. Conclusions

The development of a Smart Home Delivery System with GeoFencing Integration presents a significant step forward in addressing the challenges associated with last-mile delivery in the rapidly growing e-commerce sector. The integration of ESP32 microcontrollers, Blynk cloud services, and geofencing technologies has shown considerable potential in enhancing the security, efficiency, and convenience of package deliveries.

The system's ability to secure packages, provide real-time notifications, and improve delivery success rates has been validated through various tests. The implementation of hardware components such as ultrasonic sensors, servo motors, and cameras, managed via a smartphone application, has demonstrated the practical feasibility of a smart delivery solution that minimizes the risk of package theft and damage. However, challenges such as hardware instability, software complexity, and GPS accuracy limitations were identified, necessitating further refinement.

Future enhancements, including the introduction of an emergency unlock mechanism, advanced location tracking, and image storage capabilities, are recommended to improve the system's reliability and user experience. Additionally, considerations for business commercialization, such as custom PCB design and the development of a proprietary user interface, could enhance the system’s marketability and scalability.

In conclusion, the Smart Home Delivery System with GeoFencing Integration is a promising innovation that addresses key issues in last-mile delivery. With further development and optimization, it has the potential to significantly improve the security and efficiency of home deliveries, benefiting both consumers and delivery service providers.

References

1. Yavari, A.; Bagha, H.; Korala, H.; Mirza, I.; Dia, H.; Scifleet, P.; Sargent, J.; Shafiei, M. ParcEMon: IoT platform for real-time parcel level last-mile delivery greenhouse gas emissions reporting and management. *Sensors* **2019**, *22*, 7380. [CrossRef]
2. Shevchenko, Y.; Reips, U. Geofencing in location-based behavioral research: Methodology, challenges, and implementation. *Springer* **2023**. [CrossRef]
3. Ooi, J.; Tan, C. Smart Modular Parcel Locker System using Internet of Things (IoT). *IEEE 11th International Conference on System Engineering and Technology (ICSET)* **2021**, *66-71*. [CrossRef]
4. Mokhsin, M.; Ludin, M.; Suhaimi, A.; Zainol, M.; Som, M. H.; Halim, H. ParcelRestBox: IoT-based parcel receiving box system design for smart city in Malaysia. *IEEE International Conference on Computing (ICOCO)* **2021**, 180-185. [CrossRef]
5. Blynk.Documentation. Available online: https://docs.blynk.io/en/getting-started/activating-devices/blynk-edgent-wifi-provisioning (accessed on 18 December 2023).
6. Blynk.Documentation. Available online: https://docs.blynk.io/en/blynk.cloud/device-https-api/update-datastream-value (accessed on 28 March 2024).
7. Random Nerd Tutorials. Available online: https://randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition-arduino-ide/ (accessed on 21 March 2019).