IOT Enabled Access Control Locker

[Redacted]

CMP408: IoT and Cloud Secure Development

2024/25

Note that Information contained in this document is for educational purposes.

.

Contents

.

1 Introduction

1.1 Background

IOT (internet of things) devices refer to devices that facilitate connectivity between the cloud or other devices on a network. These devices extend the functionality of systems by allowing them to quickly transfer data and interact with the world, often physically in the form of embedded devices like sensors (Sorri, et al., 2022). IOT is often employed in applications such as home automation or industrial monitoring as a result of this, allowing for fast data collection from disparate locations and sources.

Current trends indicate that IOT device usage is only increasing, see Figure 1, with the current estimate exceeding 75 billion devices by 2025. This represents a significant issue when it is considered that IOT devices often make use of simple internal hardware often reducing the security capabilities of such devices. When combined with the fact that IOT devices are primarily focused on ease of use, convenience and rapid production it's clear to see how these devices can pose a threat to infrastructure if deployed improperly. (Kabir, et al., 2023).

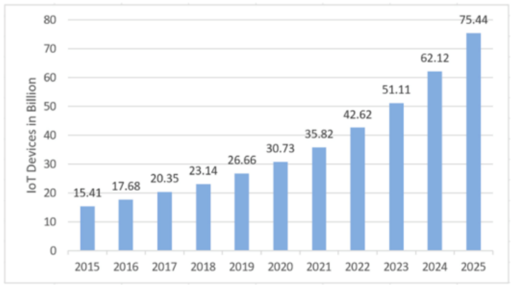


Figure 1 - Internet of Things prediction from 2014 to 2025 (in billions) (Hayajneh, et al., 2025)

IOT allows for the creation of non-static security codes that can be transferred to static security devices (for example, an electronic lock) meaning that access can be controlled in a timed fashion. This is an ideal solution already employed by some systems, such as an Amazon Locker where access codes are uniquely provided to an individual based on an account. Based on this the researcher elected to create an access control system mirroring that of an amazon locker, wherein a keypad prevents access to a box until a provided randomly generated string of characters is entered, allowing access.

1.2 Objectives

The objectives of this project are as follows:

- Integrate hardware such that keypad and servo motor respond to each other preliminarily
- Develop a system for account generation
- Develop a means for number communication from the online system and subsequent verification on the hardware side

- Integrate the systems to create an IOT vault capable of generating unique codes and unlocking it in response

2 PROCEDURE

2.1 CREATING THE INITIAL BOX

The amazon locker is ultimately a recycled cardboard box. This is not realistic material for a secure storage solution but serves as an effective proof of concept, as well as being easy to work with. The locking mechanism is a simple vertical latch employing the servo as a lock, with the keypad on the front.

2.2 SOFTWARE LINKING OF SERVO AND KEYPAD

A series of preliminary programs were developed to interact on a hardware level from the RPI. These were as follows:

- A program that tested positioning the servo from two vertical positions
- A program that took user response from the keypad and verified it against a shown code
- A program that once it received the correct code, moved the servo, unlocking the door.

This experimentation produced a simple offline control of the servo in response to a correct code entered into the keypad. This formed the basis of the locking system. This makes use of a library that accompanies the keypad to read and verify the responses, moving the servo motor right or left as is appropriate. These components can be seen below in Figure 2 and Figure 3, with the basic code found in appendix A.

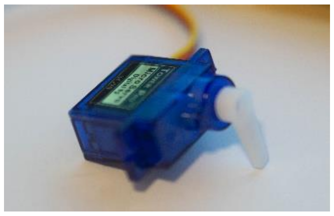


Figure 2 - TowerPro Servo Motor - SG90

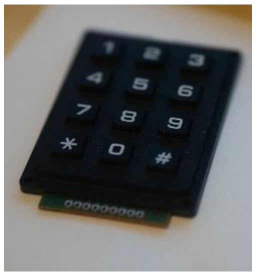


Figure 3 - 3x4 Matrix Keypad

This system fundamentally operates primarily in the userspace owing the fact that the kernel space should be ideally avoided if at all possible due to the immense security risks presented. The primary interaction with the kernel relates to how the GPIO pins are managed through kernel space drivers. The

PigPIO library is employed for hardware PWM for servo operation which makes use of the pigpio daemon to interface with the kernel's GPIO and PWM subsystems.

2.3 Creation of web based GUI

The chosen cloud storage solution was Amazon DynamoDB – this is a NoSQL database with high performance and low latency. This was chosen primarily owing to it's cheap operation price, as despite the fact that the researcher would generally prefer to work with SQL architecture, DynamoDB is an ideal solution for working efficiently on a budget. For instance, the developer used 0 of the available $50 such was the economy of this database.

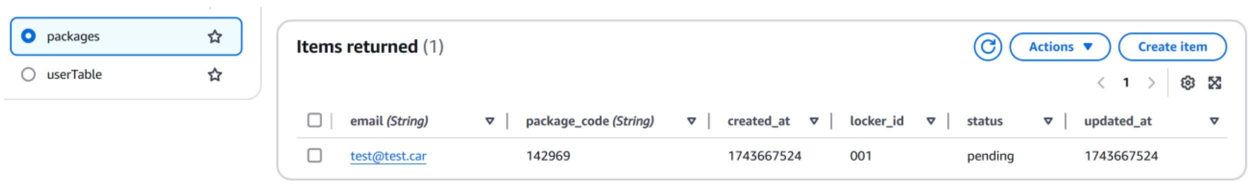
Flask was deemed the best and lowest resource solution for hosting a webserver on the Raspberry Pi, allowing for online interaction with the hardware components, over say Apache where the interfacing proved to be significantly more complicated, with the AWS SDK For PHP being decidedly temperamental and causing significant problems when attempted to be ran through either composer or locally – meaning this was abandoned as an idea. This still posed an unbelievable amount of technical issues owing to the Rasberry Pi being outdated. For example: Bcrypt was initially intended to be employed but this proved to disagree with the Rasberry Pi, meaning hashlib SHA-256 was employed instead.

The account system operated on a simple html frontend with a Python backend that interfaced with the user table in the NoSQL database to allow for registration and login with an Email and password. All requests were sanitized as POST requests due to GET requests being inherently vulnerable. This would present even less of a security risk were a certificate to be obtained allowing for HTTPS traffic, further obfuscating the data and increasing security. This was not done due to the expense of what serves as mostly a proof of concept system.

All data taken as input was sanitized including the login/registration systems to ensure a reduction in the risk of NoSQL injection.

2.4 Package/Number verification system

A second table was employed to store the package code and the status of the package (See Figure 4). This could then be interacted with when a user entered the correct corresponding code for a package, setting the package to "Completed" rather than pending. The Subprocess library was used to retrieve the response of the user app from the keypad due to the fact that real-time response from the user app proved complex and a lack of constant hardware polling was desirable for performance reasons.



email (String)	package_code (String)	created_at	locker_id	status	updated_at
test@test.car	142969	1743667524	001	pending	1743667524

Figure 4 - Amazon DynamoDB Packages Table Example

2.5 Combination

When the previously created individual elements were combined this formed a system wherein a user would login to a package frontend displaying data of their packages, the user could then elect to “complete” one of these packages, wherein they were provided with the ability to enter the package code on the physical keypad, which in turn set the package status to “completed” and moved the servo motor, opening the door. The servo motor can be seen below in Figure 5 & Figure 6 “unlatching” the door after a successful code entry.

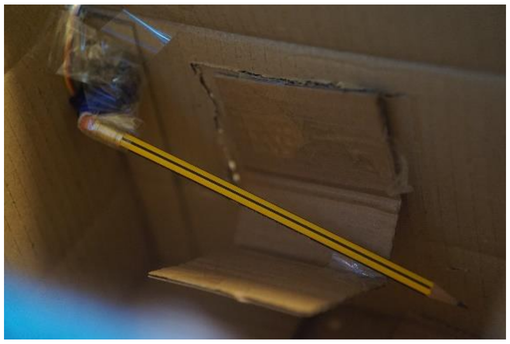


Figure 5 - Door "latched" using servo and a pencil

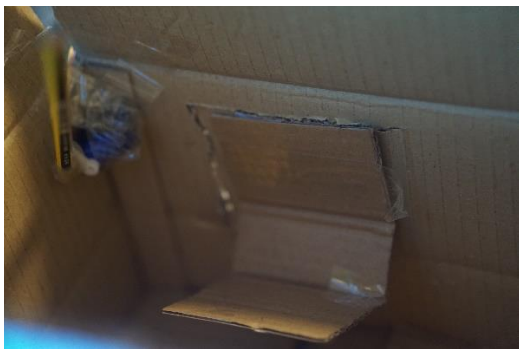


Figure 6 - Door "unlatched" after successful code entry (Position of servo changed)

3 Conclusion

3.1 Summary

In conclusion, the project managed to hit all of the intended objectives. The project makes appropriate use of an IOT device to interact physically with the environment, there is a software component integrated with amazon cloud services to form a cohesive system. The software component employs a reasonable degree of software design making use of best practice when it comes to interaction with hardware components and generally employs a high degree of security.

3.2 Future Work

In the future greater abstraction into the software architecture would be ideal, for instance programming a framework for the keypad making use of a custom-built device driver. This would likely be time consuming and was deemed unnecessary to facilitate the goal of creating the system at present.

An IR beam cutter was obtained and was intended to be integrated such that the system could automatically detect when a package was removed and mark it as completed, but time constraints prevented this from being properly implemented (see Figure 7).

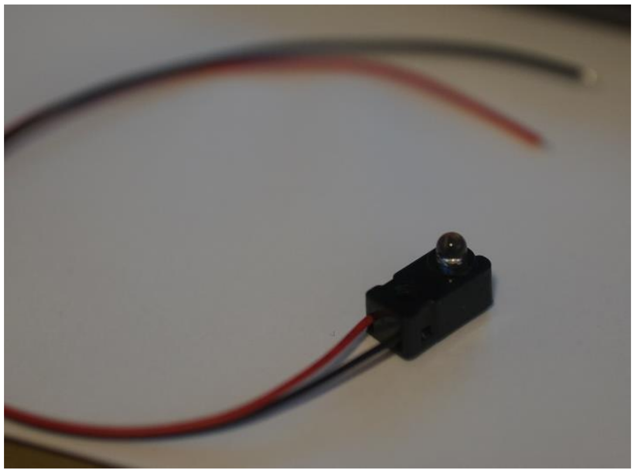


Figure 7 - IR beam cutter

4 REFERENCES

Hayajneh, A. A., Bhuiyan, M. Z. A. & McAndrew, I., 2025. Improving Internet of Things (IoT) Security with Software-Defined Networking (SDN). *Computers*, 2 April, p. 9.

Kabir, M. A. A., Elmedany, W. & Sharif, M. S., 2023. Securing IoT Devices Against Emerging Security Threats: Challenges and Mitigation Techniques. *Journal of Cyber Security Technology ,* 7(4), pp. 199-223.

Sorri, K., Mustafee, N. & Seppänen, M., 2022. Revisiting IoT definitions: A framework towards comprehensive use. *Technological Forecasting and Social Change,* Volume 179.