UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER

COMPUTER SCIENCE HONOURS
FINAL PAPER
2017

Title: Smart Locking System

Author: Raees Eland

Project Abbreviation: LockIt

Supervisor(s): Gary Stewart, Craig Balfour, Samuel Chetty

Category	**Min**	**Max**	**Chosen**
Requirement Analysis and Design	0	20	20
Theoretical Analysis	0	25	0
Experiment Design and Execution	0	20	0
System Development and Implementation	0	15	15
Results, Findings and Conclusion	10	20	13
Aim Formulation and Background Work	10	15	12
Quality of Paper Writing and Presentation	10		10
Quality of Deliverables	10		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	0	10	0
Total marks		**80**	

LockIt: Smart Locking System

Raees Eland
Department of Computer Science
University Of Cape Town
elnrae001@myuct.ac.za

Abstract

Traditional locking systems make use of a key and a lock, with access granted to those who have a key to that specific lock. With the emergence of embedded devices being used as in an Internet of things environment, smarter and more secure ways of locking can be implemented. This project looks at implementing one such way to control access to the lockers in the honours lab. With students out numbering the amount of lockers available, a smart locking and storage system is needed for all to use it in a fair manner.
The Locking system is built using an Internet of things device and various electronics that minimises power consumption and is easily scalable. Interaction with the system is done using either a mobile application or website. An onsite touch screen application is also made available to the users alongside an RFID reader. A dedicated webserver processes commands and reservations and notifies the locking system on when to open a specific locker.
Testing the performance and reliability of the system, it was found that this first initial prototype of the system was a success and can be used as a stepping stone for a final product that can eventually be implemented in the honours lab.

Keywords

Raspberry Pi, Smart Locking, Internet of Things, Performance, Switching Circuit, AES, CBC

CCS Concepts

•**Hardware → Communication Hardware, interfaces and storage**; Displays and imagers; Wireless devices; **Integrated circuits;** Input/output circuits; Digital switches; Transistors; **Hardware Test;** Hardware reliability. •**Software and its Engineering → Embedded Software**;

1 Introduction

1.1 Lockit

Traditional locking systems comprise of a key and a lock which can be prone to error. Some issues include but not limited to be theft of keys, loss of keys and misplacement. This systems also assumes each user has one or more locks for each key they possess. Meaning that no two uses can use the lock unless both have a key or share the key to a locker. This poses problems related to security and privacy, unless users are fine with that.

With current advances in technology a smart locking system can be designed and implemented to address the issues of traditional locking systems. The goal of this project is to develop such a system that allows the management of a set of lockers using cost effective methods that are user friendly and scalable.

Users will be able to use this system through various means. The user can communicate with our system through their smart phones or computers (Website). An RFID reader and a touch screen app will also be implemented for the user. The system admin can also manage this system through a separate website.

1.1.1 Motivation

The newly renovated Honours lab lockers currently have no locking mechanism, which is not ideal for security and usability purposes. There are currently 34 lockers in the honours lab and roughly 60 Computer Science honour students. Thus a smart locking system is needed to satisfy the student's storage needs.

Since all Computer Science students have access to either a smart phone or computer that is connected to the internet a smart locking system can be created to cater for students storage needs. The idea is to build a system that is usable, convenient, efficient and self-managing. This system uses technologies that currently exists and incorporates it to develop the system.

The aim of this project is to allow the user to communicate with our system through various methods. Thus enabling them to book a locker for a specific time and open their locker. The admin website will also allow the administrators full control over the system.

The focus of this report will be on the creation of the physical locking mechanism using cost effective and scalable methods and how it will interact with the rest of the system.

1.1.2 System overview

The system has three distinct parts:

1. The physical locking mechanism alongside the RFID reader and touch screen application.

2. The Webserver that receives requests and information from the other two sections.

3. A smart phone application and website for users as well as a system admin website.

Each part is needed and there is a logical flow to the system. The user will request a locker or open a locker though the phone application or website, this information is relayed to the webserver and either stores a user's booking or sends a requests the locking mechanism to open a specific lock. Figure 1 shows an overview of the system and its different parts.

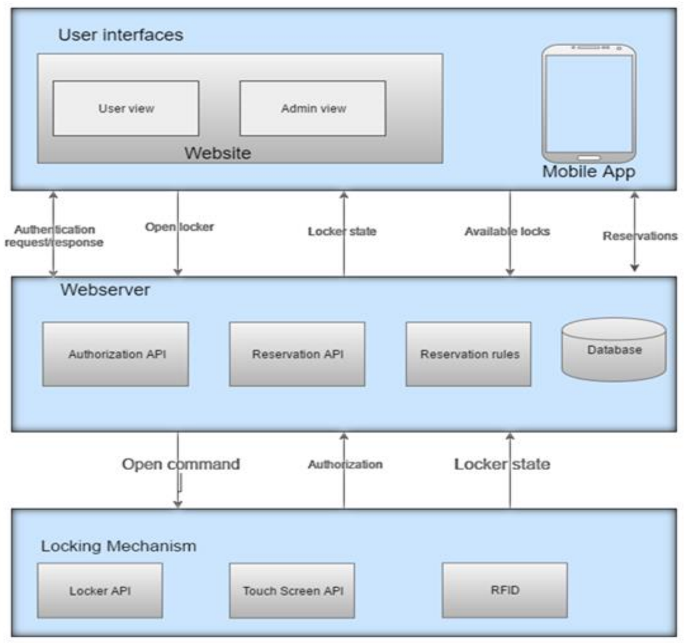

Figure 1: System Overview

1.1.2.1 User Interfaces

This part serves as the primary means of communication between the user and the system. The main purpose of the user application is to allow the user to be able to book a locker or open their locker. The system admin website will be created to allow full control over the system. Some features of the user application includes:

1. Booking a Locker
2. Opening a Locker
3. Viewing available lockers
4. Free their lockers

Features of the system admin website include:

1. Delete a user
2. Open any locker
3. Reserve any locker for an indefinite amount of time
4. Add new lockers

These applications will communicate directly to the webserver through http requests.

1.1.2.2 Webserver

The webserver serves as the brain of the system. This is where users are stored in a MySQL database and various information regarding the user. The webserver serves as an intermediary between the physical locking system and the user applications. The webserver processes requests given by the user through the user applications and decides what to do with the request. Communication to the server is done through https requests. The webserver is hosted on a virtual machine running Linux and uses apache. The APIs of the webserver is created using Flask (Flask is a light weight python web framework). Some features of the webserver include:

1. Self-managing
2. Process requests in real time
3. Versatile

1.1.2.3 Physical Locking Mechanism

This is the topic of this report. The locking system will only communicate with the webserver through http requests. An RFID reader and touch screen is included in this part of the system, since it will communicate directly with the locking system, however the touch screen will also be communicating with the webserver to get information on users who use this as a means of booking a locker or opening their locker using their UCT student card.

1.2 The Smart Locking System

1.2.1 Problem Statement

Students require lockers to provide a personal secure storage for their belongings to reduce their load as they commute between classes on campus. The use of traditional physical keys for locks poses problems in the modern world.

1.2.2 Research Question

The main research question for this project consists of three parts.

1. Is it possible to control an array of locks by developing software for an embedded system to allow users to remotely open their locks from a mobile phone, website or using an RFID.

2. Is it possible to use a webserver to integrate a cross platform applications with an embedded system?

3. Is it possible to integrate a cross platform application with embedded devices to create a secure smart locker system?

Once we investigate these three parts we can finally answer the big question: can we develop a smart locking system for the honours lockers that is self-managing, scalable and cost effective that will be accepted by the users. This report will focus on the design, implementation and testing of a system that will answer the first question posed above.

1.2.3 Design Overview

Figure 2 is the communication diagram that this report will focus on. It includes the communication between the webserver and Raspberry pi, the RFID and touch screen that is attached to the Raspberry Pi. The Raspberry Pi sends signals over its GPIO pins to the switching circuit which is powers by an external 12V power supply which in turns controls the opening of the locks.

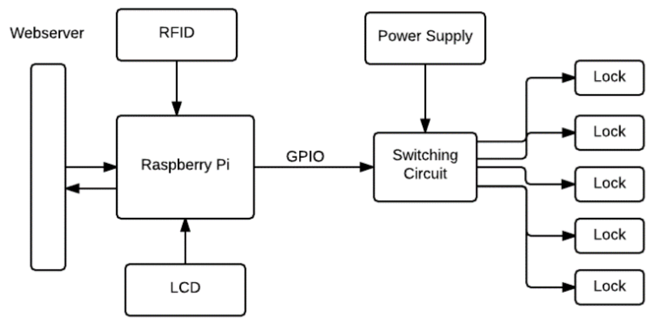

Figure 2: Locking System Overview

Switching Circuit

The switching circuit is made up of transistors (transistors act as switches, hence switching circuit) and resistors. Its main purpose is to direct the control signals coming from the GPIO pins of the Raspberry pi to the locks. The circuit allows the system to have individual control over each lock. It will also control the drive voltage from a 12V power supply, turning it on when a lock needs to be opened for only a few seconds. This limits current being drawn and saves power costs since the lock that will be used is a solenoid lock that requires 12V to open and no voltage or current need be drawn in order to keep the lock closed. This particular circuit will only control 9 lockers, however it can easily be scaled up to control many more locks if need be.

Raspberry Pi

The embedded device that will send control signals to the switching circuit is the Raspberry Pi 3. The Raspberry Pi is a credit sized computer which was inspired by the 1981 BBC Micro. The original intention of the device was to improve programming skills and hardware understanding [1]. Some advantages of using the Raspberry Pi 3 include:

1. It comes with Wi-Fi and Bluetooth enable
2. It is possible to form an expandable system with various electronic components (sensors and electronic circuits) using digital inputs and outputs, I2C or SPI protocols.
3. Expansion and communication with network devices over a LAN adapter is possible.
4. C, Python or object oriented languages such as C++ and Java can be used for programming of Raspberry Pi [2].

The above advantages of the Raspberry Pi 3 make it the perfect device for our system, however it is noted that some delay is expected between a program call that activates a pin and the GPIO pin activation.

Webserver

The Raspberry Pi and the webserver communicates through http requests. The server sends open commands to the Raspberry Pi to open a specific locker. When users use the touch screen to book a locker, the Raspberry Pi sends requests to the webserver.

1.3 Requirements

In order for this project to be a success the system needs to meet a few requirements. Students require a safe storage space that can be used by all. This part of the project requires multiple locks to be controlled by a single device for cost effectiveness. The system needs to be reliable so students can store their items safely and use the system without complaints. It also needs to be secured to protect user’s items. The students must feel as though the system is safe and convenient to use, therefore the system must be:

1. Cost Effective
2. Scalable, lockers can be added to the system with minimal change to the current system.
3. Secure, hackers should not be able to break the system and user’s information should be secure.
4. Reliable
5. Self-managing
6. The locking system needs to communicate well with the webserver to achieve greater reliability and performance of the entire system.

Therefore the aim of this project is to develop a smart locking system that meets the above requirements.

2 Background

With advancements in technology faster and smaller embedded systems can be built and used for education purposes, security and other useful projects. One such application is smart locks. Smart locks in the context of this project is a lock that can be controlled over the internet. This has some issues and benefits. Issues regarding architecture arise mainly from security, however there are many mitigating factors and tools to reduce the risk arising from security issues. Some benefits include: Users no longer need to carry a key with them, better security than traditional locking systems and can be beneficial to visually or physically impaired people.

2.1 Related Smart Locking Systems

Smart locking systems is not a new thing. There are many projects and papers written about smart locks and smart security systems over an internet of things environment. Looking at three such projects, discussing hardware and software used as well as differences, similarities and limitations of each.

In the project done by Rafid Karim and Haidara Al-Fakhri [3]. Their smart locking system is achieved using NFC (Near Field Communication), Microcontrollers and a smart phone to control the lock. The NFC module makes it impossible to remotely open the lock if the user is not within a certain distance of the lock.

This gives extra security measures. They used a PoE powered circuit board containing a MSP430 microcontroller to work along with a NFC reader, which was connected through the Serial Peripheral Interface. The following steps were followed in order to realize their goal of a smart locking system:

- Connecting the microcontroller to a network
- Make the microcontroller download its application at boot time
- Connect a NFC reader to the microcontroller
- Create a smart phone application
- Create Custom UDP packets to be sent and received by the microcontroller
- Connect sensors to the microcontroller
- Connect and Control a servo motor connected to the microcontroller
- Setup a web server and homepage to control the microcontroller. [3]

It is noted in the paper that some of the above goals were not achieved. They were able to send UDP packets and develop a working NFC reader.

Pandurang [4] et. al. looked at smart locking systems that detects user's motion by a camera and only then will the user be able to open the lock. The camera need only detect the user in front of it for the user to open the lock. They achieved their smart locking system using an ARM7 microcontroller with Bluetooth capabilities to act as a motor to unlock and lock the lock. A camera was attached to the controller to detect user's movement in front of the lock.

Hashim[5]used a PIC microcontroller (PIC16F877A) with an extra WiFi peripheral to control a lock. The WiFi module used is an XBee WiFi module. The PIC microcontroller is connected to a relay which in turn is connected to the lock. An Android smart phone application was also developed which connects to the WiFi module to control the lock.

Many other projects based on smart locking systems were done and tested. The following shows a design diagram of such a project.

The following diagram is used by Kamelia et. al. [6]

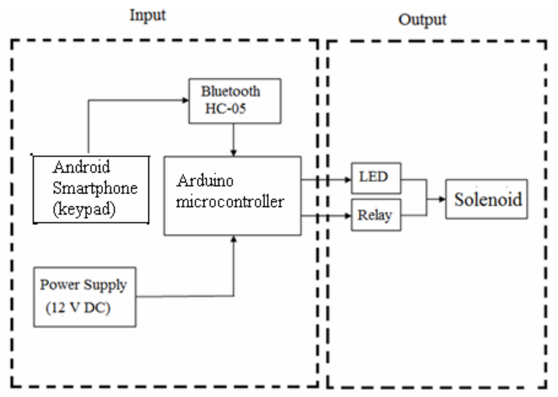

Figure 3: Block diagram of door automation system using android.

2.2 Security and Limitations

A project done at UC Berkeley [7], highlights security threats an automated lock system could potentially face. The four security issues they focus on are physical attacks, Theft of authorizing device, revoked attack and a relay attack, focusing on a security measure for unauthorized access. The paper looks at five different types of smart locks and how they each overcome the security problems. The locks are Kevo, August, Dana, Okidokeys and Lockitron. Four of the five locks tested relied on the user's phone for connectivity to the Internet. The only way the lock can receive state updates from the manufacturer's servers is through the user's phone. State consistency attacks focus on this type of model that allows the attacker to avoid revocation and access logging. All the locks tested allowed the owner to revoke other user's access. This is done when lost or theft of the device occurs. Kevo, August and Dana provide some form of auto-locking system. Whenever a device with the correct access permissions enters the communication range of the smart lock, the door will unlock automatically. While this interaction model greatly improves the usability of lock systems, it was found that all existing locks that provide this functionality can undesirably unlock the door by accident, allowing a physically-present attacker to gain unauthorized access [7].

The Raspberry Pi 3 model B has a 1.2GHz quad core ARM CPU with 1GB of RAM. The processing power of the Pi is sufficient for the purposes of this project. The only limitation of this Raspberry Pi is the amount of GPIO pins it has, however it will be shown in this paper that, one only needs a few GPIO pins to control multiple pins with the use of some circuitry and design. It is noted that the Raspberry Pi does not have a real time clock, thus when a call is made to open a lock some delay is expected. Since users will be opening their lockers over a network some extra delay can be expected depending on signal strength of the network and traffic. Therefore the complexity of this design is based on how many lockers there are. The prototype only consists of 9 lockers, therefore circuit complexity is minimal and can be built by one person, however when the prototype is scaled up it is recommended to use PCBs (Printed circuit boards).

2.3 Discussion

All three projects reviewed make use of a smart phone either using WiFi or Bluetooth to control the lock. Each uses a different microcontroller and have their own extra features. They all control a single lock making it ideal for private use, however some security issues are noted. For the lock controlled with Bluetooth, any person with a Bluetooth capable phone can open the lock. In Hashim's [5] project they use a self-made application, therefore anyone with that application can open the lock when connected to the same WiFi as the WiFi module used. All these project assume personal use which causes less concern over security and more over functionality.

This project differs to the rest in that we now want to be able to control an array of locks using multiple devices. The Raspberry Pi will be connected to the eduroam network, thus users must be connected to the eduroam network as well to open their lockers. However we need to control access to the lockers since only

Computer Science Honour students can use them. This calls for extra security measures to be put in place.

3 Design

The purpose of the Lockit project is to design and implement a working prototype of a smart locking system for the honours lockers situated in the honours lab. The system should be reliable, safe, cost effective and self-managing. This moves away from the traditional methods of locking and provides a more modern and secure approach to storing ones belongings.

The Raspberry Pi is used as the primary device to control an array of lockers. The Raspberry Pi communicates with the middleware to determine which lock should be opened. Accompanying the Raspberry Pi is a LCD screen which contains the touch screen app with very basic functionality. Functionality of the touch screen app is reduced to encourage users to use the smart phone application or the website. The touch screen app is only for fast booking of a locker. An RFID reader will also be situated on the Raspberry Pi to allow the user to open their locker with their UCT card.

3.1 Technology and Tools

Tools used are the Raspberry Pi, RFID reader, LCD screen, Solenoid locks and a switching circuit. On the software side Raspbian Jessie was used as the operating system, flask, and python to develop the touch screen application and to read information coming from the RFID reader. AES (Advanced Encryption Standard) is used to send messages between the flask server on the Raspberry Pi and the webserver. AES is a symmetric encryption algorithm, meaning one key in used for both the decryption and encryption process. It was designed to be efficient in both hardware and software.

3.1.1 Hardware

3.1.1.1 Raspberry Pi

As described before: the Raspberry Pi is a credit sized SoC capable of performing a variety of tasks. The Raspberry Pi Model 3 B will be used because of its Wi-Fi capabilities and faster processor to run all the applications while receiving information from a webserver. The Raspberry Pi is an easy to use device with plentiful documentation online and a Linux operating system. With all these features the Raspberry Pi is basically a mini-computer. This Raspberry Pi also has 34 GPIO pins which can be used to control a lock, making this device perfect for this project. The GPIO pins send signals to the locks to open a lock.

3.1.1.2 Solenoid Locks and Push Latches

The solenoid locks require 12V to turn on. The reason for this type of lock is because its default state is closed and a 12V voltage is needed to open it. Since the locks will be closed for majority of the time, no current will be drawn. When a lock needs to be opened, 12V will be sent to the lock for a few seconds. This is done to reduce heating of the locks and minimize electricity costs. A push latch will then open the locker for the user.

3.1.1.3 Touch Screen LCD

A 5’’ touch screen will be used. This will allow the user to book a locker. A PyQt GUI application will run. A Raspberry Pi 5’’ LCD display is used.

3.1.1.4 RFID Reader

The RFID (Radio-frequency identification) reader will scan a user’s UCT card can transfer the information to the Raspberry Pi via a USB connection. The RFID uses electromagnetic fields to identify the cards and obtain information from it.

3.1.1.5 Switching Circuit

The switching circuit consists of multiple TIP122 and TIP127 transistors as well as 8.2K ohm resistors. The reason for using the TIP122 and TIP127 include a high amplification factor of 1000 and a low turn-on voltage of 0.2V. This is ideal since the locks require a 12V drive to unlock it and the GPIO pins of the Raspberry Pi can only produce 3.3V.The resistor is used to control the amount of current flowing to each row of locks. This is done to limit the amount of current drawn from the Raspberry Pi, since drawing too much current can lead to the Raspberry Pi breaking. The TIP127 (only one is used) is an npn transistor that controls the 12Vs coming from the power supply. The rest of the TIP122 (pnp transistors) is used to control each lock.

3.2 Software

3.2.1 Operating System and Python

The operating system used is the latest for the Raspberry Pi: Raspbian Jessie. This operating systems comes pre-installed with development tools such as Python, java as well as some IDEs. This makes set-up easier and more efficient. Python is used to develop all software. A Flask server was needed and Rapbian Jessie contains Python packages that control the GPIO pins of the Pi, hence Python is used. Python 3 is used.

3.2.2 Flask Server

Flask is a lightweight web framework for python. It is easy to use and well documented. A simple server was needed to communicate with the middleware, thus Flask was chosen since it is easy to implement, understand and install. The main purpose of the Fask server is to receive and process an http open request from the webserver. It then turns on the GPIO pin associated with the lock that needs to be opened on, which opens the lock.

3.2.3 Touch Screen Application

The touch screen application was developed using PyQt. PyQt is a Python wrapper around the QT framework for creating graphical user interfaces. PyQt is one of the most popular GUI frameworks used in Python. It has a wide variety of functions and methods which are all well documented and easy to use. PyQt is a free software developed by Riverbank Computing. The touch screen application will allow users an extra way of booking a locker if they have no access to a phone with the mobile application on it or a computer.

3.2.4 Security

The Raspberry will only need to communicate with the webserver. Since the Raspberry Pi will communicate with the webserver via http requests and device on the same network can also access the Pi once they know the url to open a locker. Thus a way is needed to ensure requests are coming from the webserver and not some other device. One way to achieve this is so send encrypted messages between the Raspberry Pi and webserver. This is where AES is used. It will send encrypted messages between the Flask server situated on the Raspberry Pi and the webserver.

3.3 Functionality and Features

3.3.1 Unlocking and Locking

Users can unlock their lockers in one of three ways. The first two ways is done via the user interfaces, namely the smartphone application and website. The third is using the RFID reader situated next to the lockers. Users would first need to book a locker in order to open one. Users cannot book more than one locker at a time. Once a lock is open the push latch will automatically push open the door. To lock their lockers, users would just need to push the door close.

3.3.2 Booking a Locker

Booking a locker can also be done in one of three ways. Either using the smartphone app, user website or using the touch screen situated next to the lockers. It must be noted that the touch screen application can only be used for booking a locker starting at the current time of login. The touch screen app is made for users who want quick and simple booking.

3.3.3 Communicating with the Webserver

There is a two way communication with the webserver. The first way is from the webserver to the Raspberry Pi flask server. The only request that the flask server listens for is an open request from the webserver using http. This increases security and performance. The Raspberry Pi sends get and post requests when a user wants to open a lock or book a locker. This is done through https requests. A token is needed when communicating with the webserver to increase security of our system. A token can be obtained when a valid UCT ID logins into the touch screen application or when a valid UCT card is read by the RFID reader. A valid UCT ID is that of a UCT Computer Science Honour student and the System Admins.

3.3.4 Security

Physical security will be about where the Raspberry Pi will be placed. A secure place and controlled area is needed to keep the Pi physically secured. Since the Raspberry Pi communicates with the webserver over http requests it is vulnerable to cyber-attacks. However the Pi only listens for one specific request. The request also comes in the form of an encrypted post message via the webserver using a key only know by the server and the Raspberry Pi flask server. Attackers would not only need to know the key used but also the encryption algorithm used. Therefore the main security issue is keeping the shared key safe and a secret, however if the system administrators feel that the key has been compromised they can easily change it.

4 Implementation

4.1 Hardware

The component that allows for an array of locks to be controlled with a limited amount of pins is the switching circuit. Figure 4 shows the wiring diagram of the system

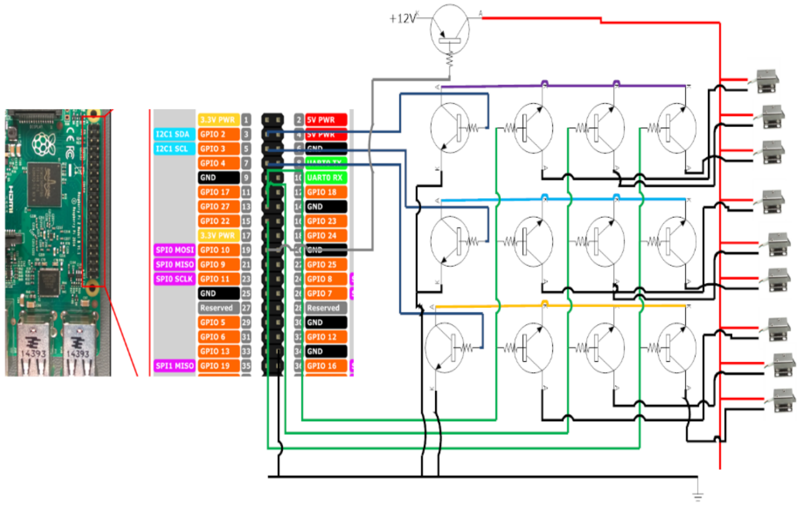

Figure 4: Wiring Diagram

9 TIP122 transistors are arranged in a 3x3 grid. Each of these transistors is connected to one solenoid lock. The emitters of each transistor in each row are connected, which in turn is connected to the collector of another TIP122 transistor. That transistor is then connected to a GPIO pin of the Raspberry Pi. Each column of transistors is connected by their base which in turn is connected to a GPIO pin of the Raspberry Pi. One TIP127 transistor controls the 12V that goes directly to the positive terminals of the solenoid locks. This allows for 9 solenoid locks to be controlled by 6 GPIO pins. Each lock can be turned on by pulling the row GPIO pin and column GPIO pin high. In Figure 4, in order to open the very top lock pins 2 and 17 would need to go high (i.e. supply 3.3V). The resistors are placed on the base of each transistor to protect it and limit current as discussed before. The circuit and Raspberry Pi is also

grounded as needed for current flow to be completed (see Figure 4). This system can be greatly scaled up by using a multiplexer (MUX) between the connection of the GPIO pins and transistors. A multiplexer is a device that selects one of several inputs and outputs it to a single line. If a 3-8 MUX is used one could control locks in an 8x8 grid, therefore 64 locks can be addressed using 6 GPIO pins (excluding pins needed for drive voltages and addressing, since a mux needs an extra pin for addressing), but this leads to extra circuit complexity which it is then advised to use PCBs and this is left for future work.

The circuit was built using components supplied from the engineering department and was built on veroboard. The power supply used to power the circuit and locks comes from an old computer. The Raspberry Pi draws its power from a power outlet.

4.1.1 Raspberry Pi

In order for the system to function as it's supposed to, three pieces of software needs to run on the Pi:

- The touch screen application, since the LCD will be connected directly to the Raspberry Pi
- The flask server
- RFID card reader software.

All software was written in python. The Raspberry Pi needed to connect to UCTs network via Ethernet to allow the flask server to listen for connections. This was done by registering the MAC address of the Raspberry Pi on the network and assigning it a static IP address (DHCP does this for us). Once this was done http requests to the flask server could be requested by anyone on UCTs network (eduroam).

4.2 Software

4.2.1 TouchScreen Application and Flask Server

The Desktop application was developed using PyQt4. It allowes for quick booking of a locker. Once a user logs into the application and is authenticated (authentication is done by the webserver) they then need to select a locker. The application first checks with the webserver if a locker is available. Once a user selects their locker and specifies a time greater than 1 hour but less than a maximum time specified by the server the locker is then booked for that specific user. Users can only specify how long they want the locker for in hours starting from the current time of login. If a user already has a locker, the application will notify the user that they have a locker. They then can use the RFID reader to open their locker

The flask server has two route methods. The first is when the RFID reader is used to open a lock and the second when an open request is made by the webserver. A thread class was needed to run both the flask server and touch screen application simultaneously. GUI applications in python execute an 'infinite loop' that always listens for user input and flask servers follow the same approach, but instead listens for http requests. Thus a thread class was needed to run both. Python GUIs must always be run from the main thread or else the application will not run. Using this structure one could update GUI elements using the emit() function from the flask application.

4.2.2 Starting App at boot time

Since the OS that runs on the Raspberry Pi boots up a desktop GUI, some configuration files and extra files needed to be changed and created so only the touch screen application would run upon boot. First a package called nodm had to be installed on the Raspberry Pi using apt-get. Nodm is an automatic display manager which automatically starts an xsession at system boot. The nodm file located at /etc/defaults had to be modified to the following:

```
NODM_ENABLED = true
NODM_USER = pi
```

Once those changes were made an .xsession file had to be created in the directory /home/pi which contained the following:

```
exec openbox-session &
while true; do
  python3 /home/pi/Desktop/Locker.py &
  python3 /home/pi/Desktop/card_reader.py
done
```

Here the python files that needed to be run at boot time were located in the Desktop folder. The while loop ensured that if either of the flask server, touch screen application or card_reader application crashed it would restart, thus recovering from a failure. Locker.py runs both the Flask server and touch screen GUI and card_reader.py contains the code needed to scan and process a UCT card.

4.2.3 Security and Communication with Webserver

For a lock to be open the webserver would need to send an http post request to the flask server, posting which lock it would like to open using the static IP address of the Raspberry Pi as the url. The url used to open a lock is: http://IP_addr_Pi/open/<message>. Where IP_addr_Pi is the static IP address of the Raspberry Pi and message contains the locker the Raspberry Pi must open. With this current system in place anyone with knowledge to the Raspberry Pi's IP address could open a locker and it is quite easy to obtain the IP address using sniffing tools. Thus a message encryption system is used. The encryption algorithm used is AES (Advanced Encryption Standard) in CBC (Cipher Block Chain) mode, using a 256bit key that is hashed before it is used to encrypt the message. The encryption algorithm sits on the webserver and the decryption algorithm on the Raspberry Pi flask server. Both the Raspberry Pi and webserver share a common key for encryption and decryption. It is impossible to try and decrypt the message sent without knowing the key. The Raspberry Pi will always decrypt messages it has been sent and if it fits the specific format the flask server is looking for, only then will it process that request. This makes it almost impossible for a hacker to send the Raspberry Pi a message to open a lock. However there are

weaknesses. If the webserver is hacked and the API is obtained to open a locker then the locks will open since the hacker will then know the format of the message and key. This can be averted by putting up firewalls for the webserver and changing the format of the message and key. Another problem that could arise is that someone gains physical access to the Raspberry Pi that controls the locks.
It is safe to say that the system is secure and will take considerable effort and skill to hack it.

4.3 Method

The following method was implemented to realize the project goals, once all the components were present and decided upon.

- Building and testing of the switching circuit
- Development of the flask server to unlock lockers
- Installation of the locks and circuitry
- Connecting the Raspberry Pi to the switching circuit to test unlocking capabilities over a local network
- Obtaining a static IP for the Raspberry Pi
- Development of the touch screen application
- The integration of the RFID reader and software to system, then testing it.
- Integrating touch screen application with the system
- Implementing threading to run both the GUI and flask server
- Implementing message encryption
- Integration with the webserver
- Only booting touch screen GUI
- Constant testing and debugging of the system.

5 Evaluation

In order to determine is the physical locking systems works as it's supposed to, two types of tests are needed. The test is to rest the reliability of the physical locking system (When an open command is sent to it will it open the correct locker always). And the second test is a performance test (How long does it take to process the open command). The touch screen application and RFID reader will also be tested. The UI design of the touch screen will be evaluated and the performance of the RFID reader will be recorded.

5.1 Touch Screen Application

The touch screen application is meant to have a simple looking UI with limited functionality. It was explained to users that the mobile application and website should be the primary means of communicating with the system and that this application was not intended to have all the features and functionality the mobile application has. The application is meant for quick booking of a locker only as described before. Therefore users felt the application had the features it needed. Feedback from the users after the first iteration of testing included:

- Making the application blue, same as the UCT colours
- Create a confirmation page after a user has booked a locker
- Centring Text on the Screen
- Greying out the Back option since users should use it less.

A second round of tests was conducted after UI changes was made and feedback was recoded, users seemed satisfied with the functionality of the application since it was explained that the touch screen application is only meant for quick booking of a locker and was only intended to offer the basic features. Changing of the font size was also suggested. This will be done for the final product.

5.2 Performance Testing

The testing set-up includes the Raspberry Pi connected to a laptop (i7, 8GB RAM) over VNC with a 8Mb/s internet connection. The Flask server on the Raspberry Pi is run using localhost. Using Google Chrome a query is sent to open a locker: ‘http://localhost:5000/open/<locker number>/’. The time taken to open that specific lock is then recorded. The RFID is then also used to open a lock and time taken to open the lock using the RFID is also recorded.

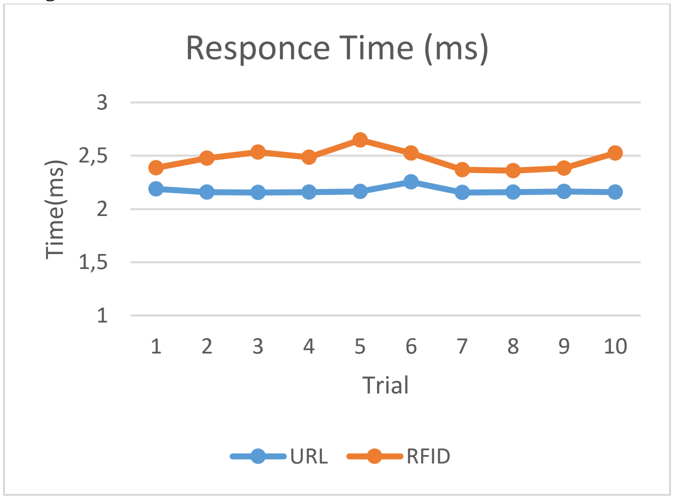

Figure 5: Line graph the time it takes to open a locker

The time it takes to open a locker using the URL or RFID is in milliseconds. This can be attributed to the fast processing power of the Raspberry Pi 3 and 100Mb/s connection between the laptop and Raspberry Pi. The RFID is expected to take longer since it first needs to process the UCT card before opening a locker.

The next step was to test how long it would take the RFID reader to process a UCT card when first using the server to authenticate a user before login the user in or opening their locker. The test above did not authenticate users.

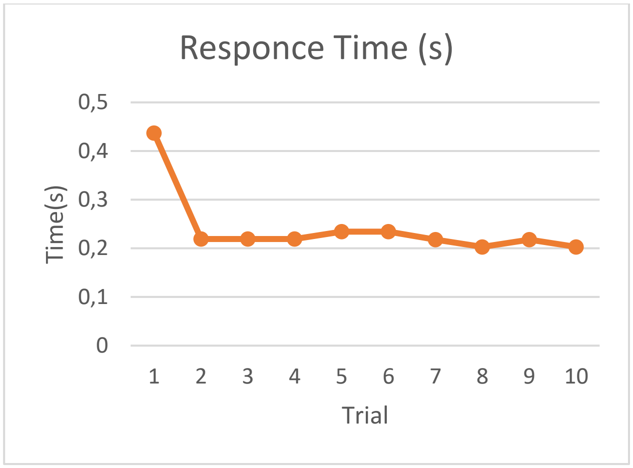

Figure 6: Time taken to process an RFID card

As expected, authenticating a user requires more time, however the time it takes to process the card is quick an users will not notice the difference since it takes less that a second to process a card.

In order to test the true performance of the system we would need to record the time it takes to open a locker using the mobile application or web app, however some difficulties were present when trying to do just that (see section 5.5).

5.3 Reliability Testing

The reliability of the system is depends whether or not will stand the test of time and not fail under extended use. As it stands all locks can lock and the system is functioning as it should. The physical locking system was installed one month ago at the date of writing this paper, and multiple tests have been conducted to test if the locking system still works. It passed the test every time. The locks was installed in the makers lab which is not used by many students, therefore the lockers were never disturbed.

5.4 Other Tests Completed

Other tests done includes:

- Testing of the switching circuit before it has installed. Testing was conducted on bread board before it was ported to veroboard.
- Testing each lock to determine if it locks and unlocks.
- Testing the RFID reader (scanning a UCT card to determine if a lock will open)

Results of the above tests were not recorded, instead the tests done above was to determine if the system was working at the current time the tests were conducted. If not the system was debugged using components such as ammeters, voltmeters, and oscilloscopes, and once an issue was found it was fixed immediately. Constant testing of each component was needed in order to ensure optimal performance and reliability. Threshold voltages cannot be tested since the locks will only open and operate at 12V with 3.3V coming from the Raspberry Pi. Any less the locks will not open and voltages exceeding 12V can possibly fry the circuitry and locks.

5.5 Discussion

Emphasize on the design and implementation was placed on this part of the project due to it being a development project. A good design and implementation will lead to a working product. All tests that needed to be conducted in order for the first prototype to work was conducted and passed. The locking system performs as it should and can process multiple requests, since http is used. Open and booking requests can comfortably be handled by this system and it does so efficiently.

Since the lockers are installed in another area where the Raspberry Pi does not have access to internet, testing the performance of the system as a whole was limited.

6 Future Work

This is a first working prototype of a smart locking system for the honours lab. Taking the design of this project a final working implementation of a smart locking system can be achieved and installed in the honours lab, however some improvements can be made to the prototype.

Better and bigger locks need to be used. One design flaw of this prototype is that the locks used were too weak to be used with the push latches. The locks were unable to open due to the strong force of the latch pushing against the door, therefore a bigger and stronger lock is needed for a final implementation of this system. A back up battery can be installed to power the system in case the power goes out, however users will not be able to open their lockers if the internet (Ethernet) goes down, therefore system admins need a back door to access the Raspberry Pi without relying on the internet. This can be done using ssh. The touch screen application can also be turned into another user interface that has the same features as the mobile application if need be.

Additional testing of this system would also be favourable. Due to the time constraints not all tests could be conducted thoroughly and only the performance of unlocking a lock was tested once the prototype was complete. Reliability testing was done while the system was being installed. However those two tests were sufficient to produce a working prototype. Other tests that can be conducted in the future include:

- Testing the strength of the lock.
- The unlocking procedure in different temperature settings
- Overloading the flask server with opening requests

7 Conclusion

This project set out to investigate whether it was possible to design and implement a smart locking system for the honours lockers that is cost effective and scalable. Research shows that there are many projects and reports on smart locks that can be controlled over Bluetooth or WiFi, however none designed and implemented a system that controls an array of locks to be used by multiple users.

The design outlined features and functionalities that a smart locking system should typically consist of as well as what components would be necessary to achieve the features and functionality stated.

The implementation of the system answered the research question: can we develop a smart locking system for the honours lockers that is self-managing, scalable and cost effective that will be accepted by the users? It showed that such a smart locking system can be built and was built as a first prototype.

The tests conducted proved that this prototype was a success and can be used as a design to build a final product. The one design flaw is that the locks and push latches were not compatible. The force of the push latch was too strong for the lock to unlock. This can be overcome by using a stronger solenoid lock or weaker push latch.

With enough time and resources this prototype can be turned into a fully functional final product with extra features that users can enjoy and use to keep their personal belongings safe. The use of such a system can extend to other services and industries such as post offices and courier services.

8 Acknowledgements

I would like to thank my group members, Marion and Norman for making this project work. Our supervisors, Gary, Craig and Sam for guiding the way and Justin Pead from UCT for his help.

9 References

[1] What is a Raspberry Pi? *Opensource.com*. Available at: https://opensource.com/resources/what-raspberry-pi [Accessed May 4, 2017].

[2] Mirjana Maksimović, Vladimir Vujović, Nikola Davidović, Vladimir Milošević and Branko Perišić, Raspberry Pi as Internet of Things hardware: Performances and Constraints, Proceedings of 1st International Conference on Electrical, Electronic and Computing Engineering IcETRAN 2014, Vrnjačka Banja, Serbia, June 2 – 5, 2014, ISBN 978-86-80509-70-9.

[3] Rafid Karim, Haidara Al-Fakhri, “Smart Door Locks”, December 2013

[4] Bhalekar Pandurang, Jamgaonkar Dhanesh, Prof. Mrs. Shailaja Pede, Ghangale Akshay, Garge Rahul, 2016 ,Smart Lock: A Locking System Using Bluetooth Technology & Camera Verification, *International Journal of Computer Applications*, 4(1), pp.136–139.

[5] N. Hashim, N. F. A. M. Azmi, F. Idris , N. Rahim, 2016, Smartphone Activated Door Lock Using Wi-Fi, *ARPN Journal of Engineering and Applied Sciences,* 11(5), pp.3309-3312

[6] Lia Kamelia, Alfin Noorhassan S.R, Mada Sanjaya and W.S., Edi Mulyana, 2014, Door-Automation system using Bluetooth-Based Android for Mobile Phone, *ARPN Journal of Engineering and Applied Sciences*,9(10),pp.1759 – 1762

[7] EECS at UC Berkeley**.** Grant Ho, Derek Leung, Pratyush Mishra ,Ashkan Hosseini, Dawn Song, David Wagner**,**Smart Locks: Lessons for Securing Commodity Internet of Things Devices | EECS at UC Berkeley. Available at: http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-11.html [Accessed May 5, 2017].