Golden Phoenicks

Locker Automation System

Jan Stanley Go, Yohaan Anthraper, Jeremy Rende

1.1 Declaration of Joint Authorship

We, Jan Stanley Go, Yohaan Anthraper, and Jeremy Rende, hereby verify that this document submitted for assessment is a collaborative effort amongst ourselves, and is written in our own wording. Usages of works of other authors, in any way (whether it be core concepts, diagrams and figures, previous technologies, programs and source code, or text from their works) are cited properly at the point in which they were used. Included in this document is a list of references used. Stanley handled the LCD display device and the mobile application's display feature, Yohaan handled the door opener device and its corresponding feature in the mobile application and Jeremy managed the database and the door lock device, along with the corresponding feature in the mobile application.

1.2 Proposal

Proposal for the development of Golden Phoenicks Locker Automation

Prepared by Jan Stanley Go, Yohaan Anthraper, and Jeremy Rende

Computer Engineering Technology Students

https://github.com/stango25/lockerautomationsystem

Executive Summary

As a student in the Computer Engineering Technology program, I will be integrating the knowledge and skills I have learned from our program into this Internet of Things themed capstone project. This proposal requests the approval to build the hardware portion that will connect to a database as well as to a mobile device application. The internet connected hardware will include a custom PCB with the following sensors and actuators 16x2 LCD Display, DC Gear Motor, Lock Solenoid. The database will store User and Product info along with signals for products. The mobile device functionality will include Lock, Unlock, Open, Close, Display statuses. and will be further detailed in the mobile application proposal. I will be collaborating with the following company/department N/A. In the winter semester I plan to form a group with the following students, who are also building similar hardware this term and working on the mobile application with me This is winter semester?. The hardware will be completed in CENG 317 Hardware Production Techniques independently and the application will be completed in CENG 319 Software

Project. These will be integrated together in the subsequent term in CENG 355 Computer Systems Project as a member of a 2 or 3 student group.

Background

The problem solved by this project is During a busy day of study, students are often burdened with handfuls of learning materials that may impede their ability to open and shut their locker. This compounded with potential disabilities makes the manual unlocking and opening of a locker a day-to-day issue. Our product will solve this issue.. A bit of background about this topic is The problem solved by this project is ease of access to a locker. Access to lockers can be difficult for student with many items to carry, or those with disabilities. Currently almost all lockers are manually operated and therefore not accessible by anyone with a severe physical disability.

Existing products on the market include [1]. I have searched for prior art via Humber's IEEE subscription selecting "My Subscribed Content"[2] and have found and read [3] which provides insight into similar efforts.

In the Computer Engineering Technology program we have learned about the following topics from the respective relevant courses:

- Java Docs from CENG 212 Programming Techniques In Java,
- Construction of circuits from CENG 215 Digital And Interfacing Systems,
- Rapid application development and Gantt charts from CENG 216 Intro to Software Engineering,
- Micro computing from CENG 252 Embedded Systems,
- SQL from CENG 254 Database With Java,

- Web access of databases from CENG 256 Internet Scripting; and,
- Wireless protocols such as 802.11 from TECH152 Telecom Networks.

This knowledge and skill set will enable me to build the subsystems and integrate them together as my capstone project.

Methodology

This proposal is assigned in the first week of class and is due at the beginning of class in the second week of the fall semester. My coursework will focus on the first two of the 3 phases of this project:

Phase 1 Hardware build.

Phase 2 System integration.

Phase 3 Demonstration to future employers.

Phase 1 Hardware build

The hardware build will be completed in the fall term. It will fit within the CENG Project maximum dimensions of 12 13/16" x 6" x 2 7/8" (32.5cm x 15.25cm x 7.25cm) which represents the space below the tray in the parts kit. The highest AC voltage that will be used is 16Vrms from a wall adaptor from which +/- 15V or as high as 45 VDC can be obtained. Maximum power consumption will be 20 Watts.

Phase 2 System integration

The system integration will be completed in the fall term.

Phase 3 Demonstration to future employers

This project will showcase the knowledge and skills that we have learned to potential employers.

The brief description below provides rough effort and non-labour estimates respectively for each phase. A Gantt chart will be added by week 3 to provide more project schedule details and a more complete budget will be added by week 4. It is important to start tasks as soon as possible to be able to meet deadlines.

Already purchased for previous course. We might get some supplies under 20 dollars for connecting devices.

Concluding remarks

This proposal presents a plan for providing an IoT solution for Our product is a culmination of our past three products (Lock, Display, and DC Motor). This will be used to create an automated locker system that will solve any underlying issue that prevents a student from controlling the locker manually.. This is an opportunity to integrate the knowledge and skills developed in our program to create a collaborative IoT capstone project demonstrating my ability to learn how to support projects such as the initiative described by [3]. I request approval of this project.

References

[1] Electronic Lockers. (n.d.). Retrieved February 01, 2018, from http://www.tiburonlockers.com/storage-solutions/electronic-lockers.php

[2] Institute of Electrical and Electronics Engineers. (2015, August 28). IEEE Xplore Digital Library [Online]. Available: https://ieeexplore.ieee.org/search/advsearch.jsp

[3] V. Stangaciu, V. Opârlescu, P. Csereoka, R. D. Cioargă and M. V. Micea, "Scalable interconnected home automation system," 2017 21st International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, 2017, pp. 169-174.

1.3 Abstract

In today's day and age, consumer products revolve around the transition from physical/manual system to automated electronic systems. Despite this, every workplace has an installation that has yet to be automated. In this case, that installation is *the Locker*. To assist those who are either physically impaired or occupied with bulky or heavy items, a system should be made to enable locker users to unlock and open their lockers from afar and at a simple tap of a button. The system's main feature would be a door opening mechanism and a smart lock mechanism. If necessary, a display with buttons would be placed to facilitate manual unlocking. The door opening mechanism and the smart lock mechanism would both rely on information from an offsite database to perform tasks, which would be updated using either an Android application or through the display and its buttons. Both the application and the display would have security features implemented through the use of an account with a password and through this account, the user would have their own devices attached to it. The database will contain user account information and user device information, as well as the statuses of the devices. This system has the potential to be used in all sorts of workplaces, including schools, to simplify the user's locker experience and monitor their lockers anywhere and anytime, so long as they are connected to the internet. This report contains the documentation of this idea becoming a reality.

1.4 Table of Contents

1.5 List of Illustrations/Diagrams

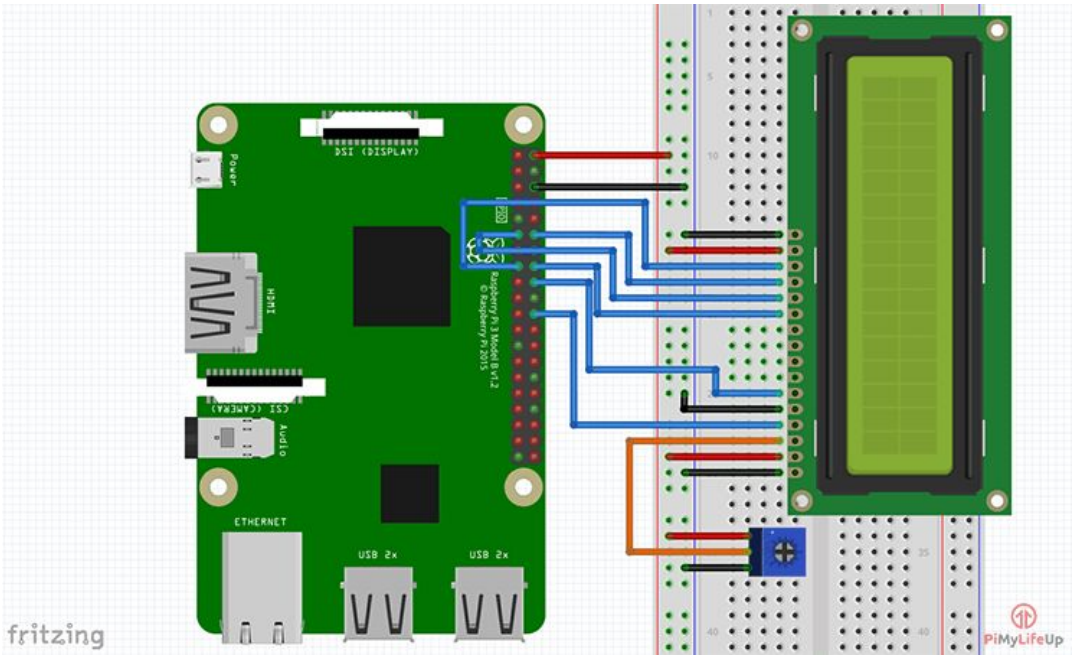


Image 3.2.3.1 Fritzing diagram for the HD44780 pinouts, courtesy of PiMyLifeUp.com

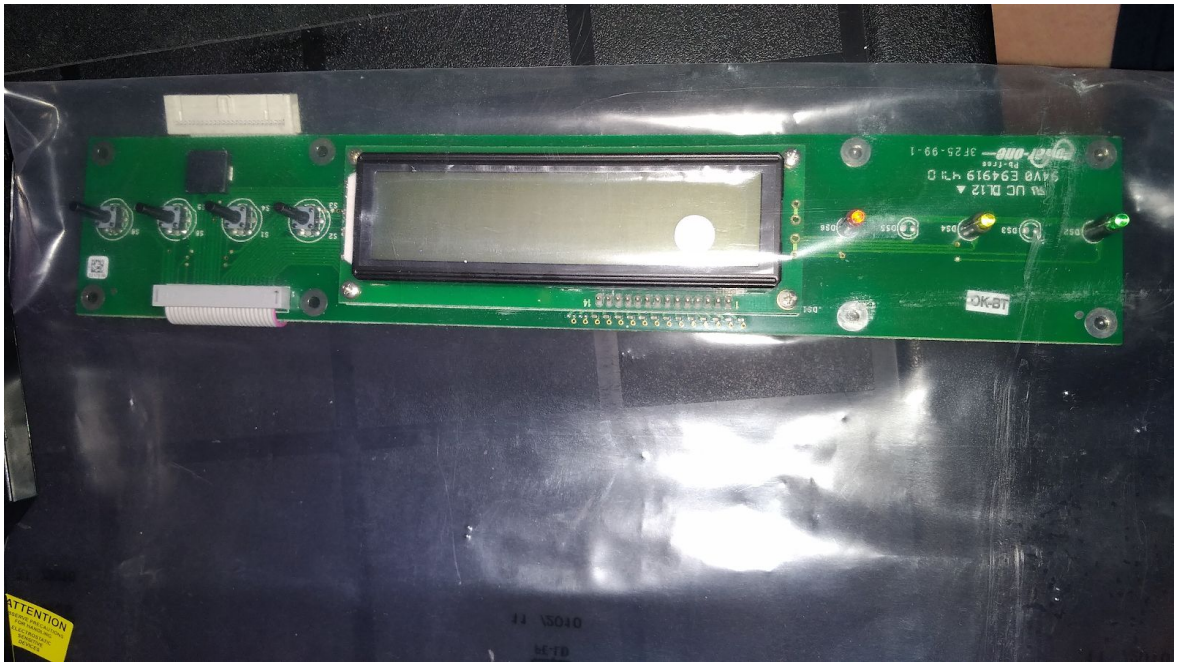


Image 3.2.3.2 LCD acquired from A1 Electronics. Image provided by Stanley

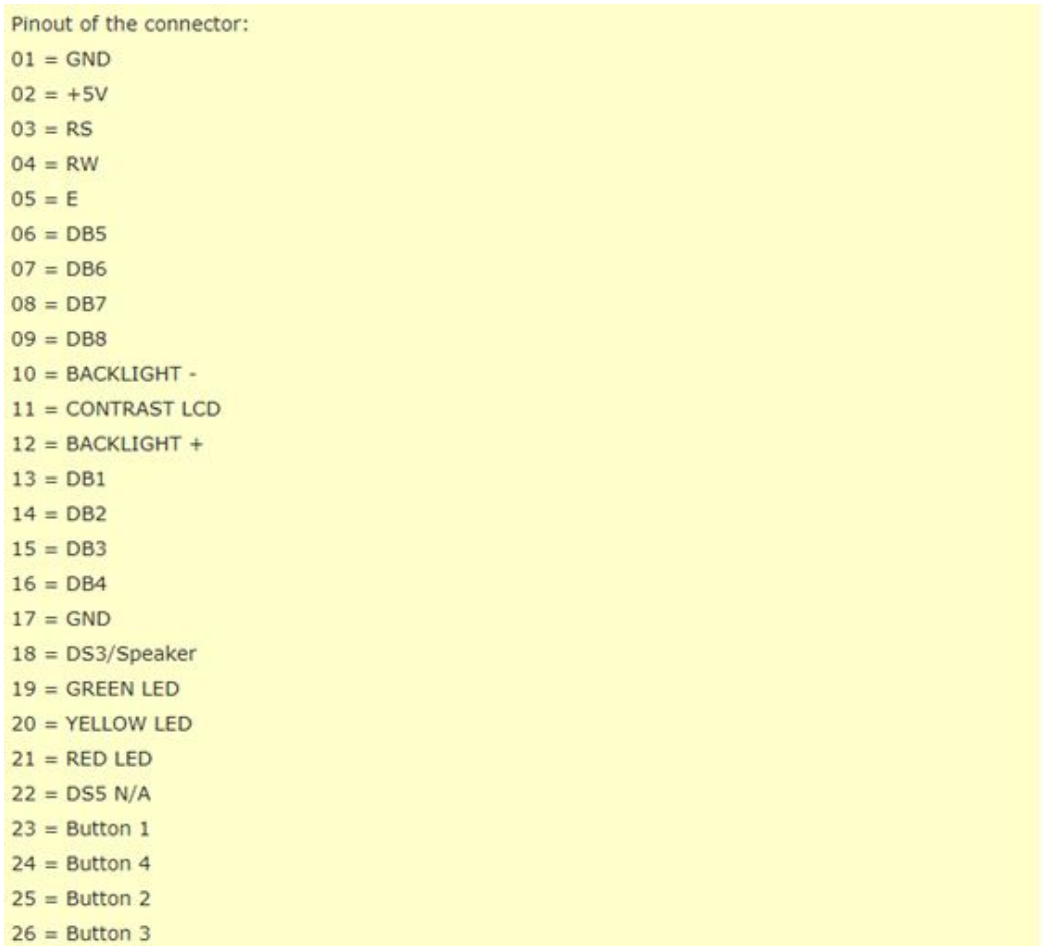


Image 3.2.3.3 Pinout for the ribbon cable soldered on to the LCD above. Note that for the purpose of powering up the LCD only pins 1-12 out of the 26 pins will be used. Image courtesy of u/devicemodder on Reddit.com.

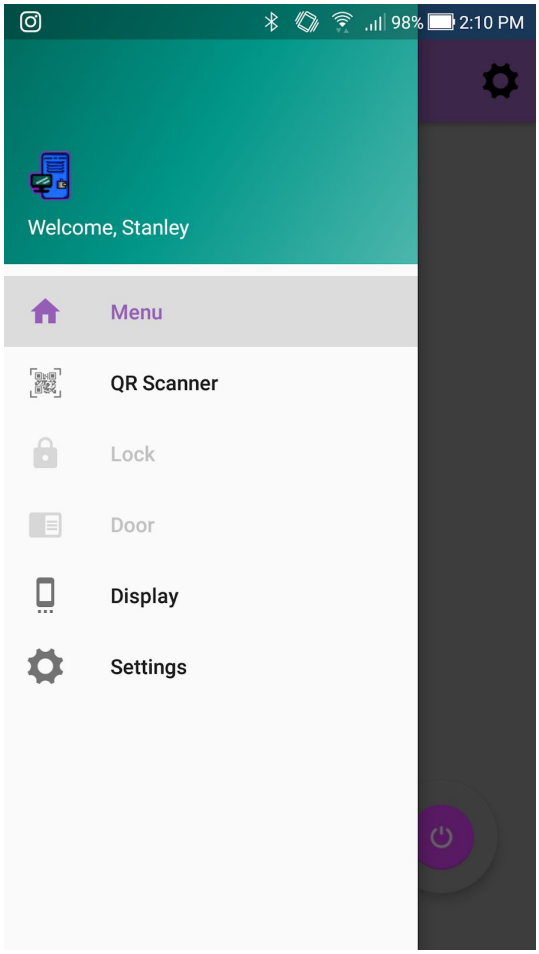


Image 3.2.4.3.1 Example of the application of the GetProducts AsyncTask. The user "Stanley" does not have any devices registered to his account and as such, because the GetProducts AsyncTask will not retrieve any values back and as such the application will disable the activities. As such the only action the user "Stanley" can do is scan new devices using the QR Scanner.

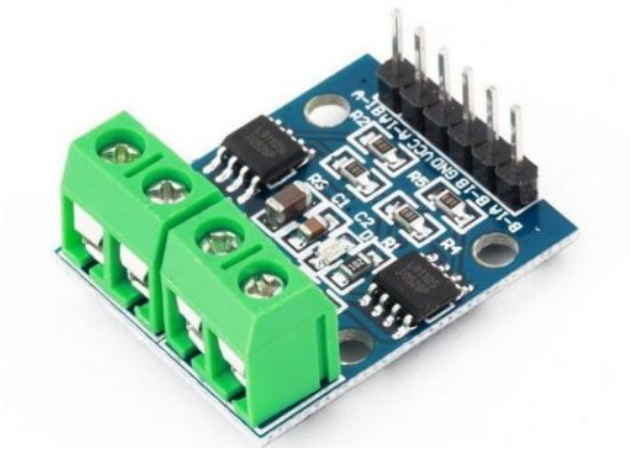


Image 4.0.1 DC Motor Transistor Driver board courtesy of robokits.net

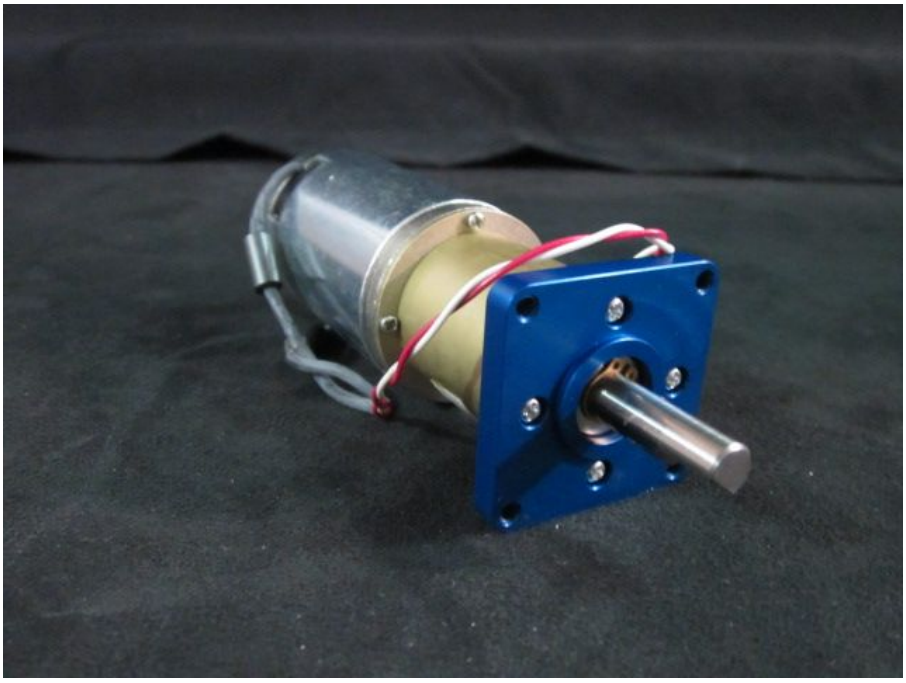


Image 4.0.2 DC Motor courtesy of capitolareatechnology

1.6 Introduction

Currently, hundreds of secondary and post-secondary institutions make use of conventional manual lockers for student and faculty use. Over the years as institutions have made strides to become more accessible and secure, the locker has not evolved. The Locker Automation System outlined in this report is a proposed solution to improve on: accessibility, usability, and security of existing locker installations.

Using a modular, Internet of Things (IoT) connected system the Locker Automation System will allow a user (student, faculty) to unlock, lock, open, close and check the status of their locker. These functions will be accessible via a mobile app and by a physical input and LCD display affixed to the front of the locker. Data will be centralized on an IBM Bluemix Cloud mySQL server. At launch, the mobile app will be available to Android users who possess a device running on API level 21 or higher, with an active Google Play Store account.

Throughout the following report different hardware and software interfaces, as well as product dimensions and physical requirements will be discussed. Also included will be data on the proposed cost of a unit conversion and the learning curve that will be faced by an average user following the transition. Additionally, and particular localization adaptation issues as well as accommodations that may be possible for those who are differently-able will be discussed, as well as security improvements and possible concerns.

2 Project Description

2.1 Technical Problem

The technical problem addressed in this report is the conversion of a manual locker, into an automated “smart” locker system. The manual locker design poses a problem for any person with an upper body physical disability. It also makes it difficult for any individuals carrying multiple objects to retrieve items from inside storage.

2.2 Reason for Project

The reason for the undertaking of this project is to facilitate increased accessibility and security over a conventional manual locker system. In other words, students from around the nation will be able to

2.3 Scope

The project will be completed over two standard college semesters (eight calendar months). The first semester will consist of creating both the separate devices and the Locker Automation application and the following semester will be reserved for the integration of of the software and hardware.

2.4 Objective

The objective of this project is to convert an existing locker installation into an accessible, automated, “smart” locker, providing ease of access to consumers and users. This system would also allow security services to monitor locker usage if necessary.

2.5 Unique Problems

Different styles, shapes and materials used in existing locker installations provide unique challenges as the goal is to make a one-style-fits-all solution. During our surveyance of different locker installations around the Humber College Institute of Technology and Advanced Learning, North Campus in Etobicoke, Ontario, Canada, we found that we would need to be flexible with how close the smart controller (Raspberry Pi) would have to be located to each of the component modules. Another unique challenge that we determined would be a factor is construction material of the locker installation and hinge design (piano hinge vs. individual hinges), these factors make mounting the DC motor a challenge.

2.6 Unique Approaches

Unlike most projects that integrate hardware devices on the Raspberry Pi with a web interface or an Android application, we decided to use PHP scripts that the Android application and the hardware devices call as opposed to the common solution in which the database querying and insertion or updating is done inside the code of the application itself. On the hardware side of things, a modular approach was taken using barrel connectors and off-the-shelf components to ensure that if a certain component did not work, due to space or design limitations, another similar, but more suitable component could be swapped in with minor modifications to the rest of the system. To mitigate potential interruptions due to power failures, an Uninterrupted Power Supply (UPS) was incorporated into the design of the system. This UPS came in the form of a dual 5V/12V

3000mAh lithium-ion battery pack. The pack is supplied mains power at 12V and should that power fail, can provide enough power to run the system for a tested 3 hours (potentially more depending on the amount of usage during the outage period).

3.1 Project Description

3.1.1 Software Requirements Specifications

3.1.1.1 Introduction

3.1.1.1.1 Purpose

The Golden Phoenicks Locker Automation App is intended to be used simultaneously with all three devices. Ideally, the students/customers should be able to control all aspects of the Locker Automation Device from anywhere using their mobile devices. This will be used in order to combat limited mobility for disabled students and students that have their hands full.

3.1.1.1.2 Intended Audience and Reading Suggestions

This document is intended to be read by the Computer Systems Project teachers, any students who would like to duplicate the project, and any users who would like to learn more information on the utilities. It is recommended that future audiences will first read through and understand the report thoroughly before moving forward with the creation of anything tangible.

3.1.1.1.3 Product Scope

The software includes database and Android integration that are to work hand in hand in order to enhance the user's experience as efficiently and comfortably as possible. The user will be able to control each device in the hardware at will without lag and with utmost security in mind.

3.1.1.2 Overall Description

3.1.1.2.1 Product Perspective

This software product came to be as a result of a need to integrate our hardware devices to modern standards and to allow users to access their devices easily.

3.1.1.2.2 Product Functions

With this software, the user is intended to be able to perform the following:

- Lock or unlock their locker doors. Jeremy will be responsible for this feature.
- Open or close their locker doors. Yohaan will be responsible for this feature.
- Display the status of their devices. Stanley will be responsible for this feature.
- Create an account to register these devices to. The handling of the database will be Jeremy's task as well.

3.1.1.2.3 User Classes and Characteristics

Users include:

- Students
- Teachers
- Company professionals
- And the general public

The three user classes can use the product so long as their workplaces or schools have lockers beforehand. The users need to be capable of operating Android devices to download the software from the App Store and must have a constant internet connection to stay connected to their devices.

3.1.1.2.4 Operating Environment

The software is to be run on any Android devices with an Operating System of 5.0 and above. The Locker Automation System hardware devices will also be needed to use the system in its full potential.

3.1.1.2.5 Design and Implementation Constraints

The developers will be limited in that they will only be producing the software on the Android platform, with a specific Operating System, and MySQL will be used for database. PHP scripts are used to communicate with the MySQL.

3.1.1.2.6 User Documentation

The software will be distributed with an online video tutorial.

3.1.1.2.7 Assumptions and Dependencies

It is assumed that the MySQL database, to be hosted on IBM Cloud in Texas, is still active. Presumably the user either has a mobile data connection or a WiFi connection as well. The Google Play Store also needs to be up and running just fine. Any updates to the software will need to be done in Android Studio.

3.1.1.3 External Interface Requirements

3.1.1.3.1 User Interfaces

The Android application must follow the Google Android specifications for proper UI design.

3.1.1.3.2 Hardware Interfaces

The Raspberry Pi will be used to establish a connection to the database which the Android application will then connect to. The Android phone needs to run with an Operating System version of 5.0.

3.1.1.3.3 Software Interfaces

The product will be using the following software libraries and components:

- Python
    - Adafruit Python LCD Libraries
    - WiringPi Python Library
- Raspbian
- Android
    - Java
    - XML
- PHP
- MySQL
- HTML
- GitHub

3.1.1.3.4 Communications Interfaces

The software design dictates that the device the application will be installed to will need a working touch interface for navigating the menus, a functioning camera for the QR Code Scanner and a constant connection to the Internet either through Mobile Data or WiFi.

3.1.1.4 System Features

3.1.1.4.1 Lock Control System

To be used in conjunction with the Door Control System. The Lock Control System would allow the user to disarm or arm the lock on the door through the solenoid lock device registered to their account through the application.

3.1.1.4.1.1 Description and Priority

This feature is a high priority item, and it's intended to be used to control the lock hardware system. Ideally it needs to be ready as it is one of two main features of the software.

3.1.1.4.1.2 Stimulus/Response Sequences

When the user chooses to navigate to the lock screen of the application, the user will either have the option to navigate back to the main menu or push a button that would affect the status of the lock registered to their account. If the user taps on the button that says "Lock", the Lock System should update the database accordingly and arm the lock so that it cannot be opened by the hardware device even if the Door Status is modified. If the user taps on the button that says "Unlock", the Lock System would set its status to be disarmed and update the database accordingly.

3.1.1.4.2 Door Control System

To be used in conjunction with the Lock Control System, the Door Control System allows the user to open or close the door through the mechanical door device registered to their account via the application

3.1.1.4.2.1 Description and Priority

The door control system is a high priority item, and it's intended to be used to control the door control hardware system. Ideally it needs to be ready as it is one of two main features of the software.

3.1.1.4.2.2 Stimuli and Response Sequences

When the user chooses to navigate to the lock screen of the application, the user will either have the option to navigate back to the main menu or push a button that would affect the status of the lock registered to their account. If the user taps on the button that says "Open", the Door System updates the status of the database, informing the user that the door should be opening if the door device status is successfully updated in the database and if the lock status is set to disarmed. If the user taps on the button that says "Close", the Door System sets the door device status accordingly on the database and informs the user that on the software side the door is closing.

3.1.1.4.3 Status Display System

Provides user feedback by allowing the user to view the status of the devices they own and ensure that the door and the lock devices are in the user's desired status.

3.1.1.4.3.1 Description and Priority

This feature is a medium priority item, and it's intended to be used to be used to allow the user to interact with the rest of the system if they do not have the software needed. Ideally it needs to be ready as it a backup interface for the user to interact with the device.

3.1.1.5 Stimuli and Response Sequences

The application consists of several pages that the user can navigate through the use of buttons and prompts. The first page that the user sees upon starting the application is the login page, where the user can then either input their username and PIN in the text fields if the user has an existing account, then confirm the input of their credentials by tapping on the login button, or tap on the button to register a new account, which will prompt the user to input their desired Username and PIN for the next time they log in, and a cancel and register button. Cancel closes the pop up window and Register sends the information to the database, where a unique user ID will be generated and their information will be registered as a new account. From either option, the user can then either register physical hardware via the QR Code Scanner, which will require Camera Access, or access their existing devices to use either the Lock Control System, the Door Control System, or the System

Display System. The QR Scanner opens the phone's camera after permissions are granted and checks for QR Codes which will be attached to the device, with a unique code for each device. If it finds an unregistered QR Code, the application updates the database and registers this device to the user's account.

3.1.1.6 Other Nonfunctional Requirements

3.1.1.6.1 Security Requirements

To maintain information security, the following measures have been decided upon and implemented:

- Secure login screen with authentication in the form of a user PIN
- Database created under a reputable company's server (IBM BlueMix)
- Database querying and updating performed in PHP scripts called by the application itself as opposed to storing connection strings inside the source code itself, which can be reverse engineered

It is due to these security implementations that the application might experience delays from time to time. To ensure a smooth operation, please allot room for a delay of up to 4 seconds between the door being opened on the application and the door actually opening.

3.2 Build Instructions

3.2.1 Lock Control System

MATERIALS

- Adafruit 12V Lock Solenoid
- 9V power supply
- ON Semiconductor 1N4004
- ON Semiconductor MJE3055T NPN BiPolar Junction Transistor
- 1KΩ 1W resistor
- Printed Solenoid board
- Fritzing PCB Software

PROCEDURE

Step 1: Order all required parts/components. Download Fritzing.

Step 2: Exercising caution, solder together PCB. (Review PCB diagram in Fritzing for component locations). To ensure that the board is soldered together correctly, connect the 9V adaptor (+) to the driver board 9V post and the (-) to the driver board GND post. Plug the power supply into the wall. Then, using a Digital Multimeter, measure the voltage between the "Pi In" and GND pins on the driver board, there should be no voltage passing.

Step 3: Connect your Raspberry Pi to a monitor, keyboard and mouse. If this is your first time booting up the Raspberry Pi, select Raspbian as the OS from the NOOBS screen. Once the OS has booted, connect the Raspberry Pi to a WiFi network then open the Terminal and run the following command "$sudo apt-get install wiringPi"

Step 4: Download the lockcontrol.py script from the repository.

Step 5: Read the code to determine which GPIO pins are being used.

Step 6: Connect the GPIO pin to the "Pi In" terminal on your driver board.

Step 7: Connect a ground pin on the Raspberry Pi to the ground pin on the driver board.

Step 8: Connect the 9V Adaptor Negative to the ground pin on the driver board.

Step 9: Connect the 9V Adaptor Positive to the 9V post on the driver board.

Step 10: Connect the Lock solenoid positive and negative to the solenoid (+) and (-) on the driver board.

Step 11: Plug the 9V Adaptor into the wall.

Step 12: Run the Python script on the Raspberry Pi. Once the spacebar is pressed, the lock should cycle (1s unlocked, then re-locked).

Step 13: If the lock does not cycle, please troubleshoot the PCB, but most likely, you overheated the BJT while soldering it into place.

Step 14: If the lock cycles, congratulations!

Prototyping notice: Do not leave any prototype device plugged in and unattended, this is how you burn down your house, or worse! I do not take any responsibility for any unintended outcomes, injuries, or damages incurred by undertaking this project.

3.2.2 Door Control System

MATERIALS

-Heatshrink

-For the purposes of this project, it was decided to use a method of connection between the rod and motor that would solidify firmly. It also makes sure that the motor tip won't bulge when opening the door.

-Metal Rod

-This can be purchased/retrieved from any scrap yard or mechanical junction. The gerth does not have to be exact as long as the diameter and radius matches that of the motor tip. The length is also up to choice as long as there is enough space for a ar to attach to the locker door.

-Transformer/Power Supply

-purchasable from amazon.ca. Voltage depends on dependencies of the driver board and motor.

-Wires/Cables

-Can be purchased in bulk from electronics stores and/or amazon

-Wire Cutters

-Electrical Tape

-Gear Motor

-Found in any old electronics parts stores (A1- Electronics/ Sayal Electronics).

-Transistor Driver Board

-The most important portion of the driver board is the L9110 S chip that can be found on creatron or sparkfun websites.

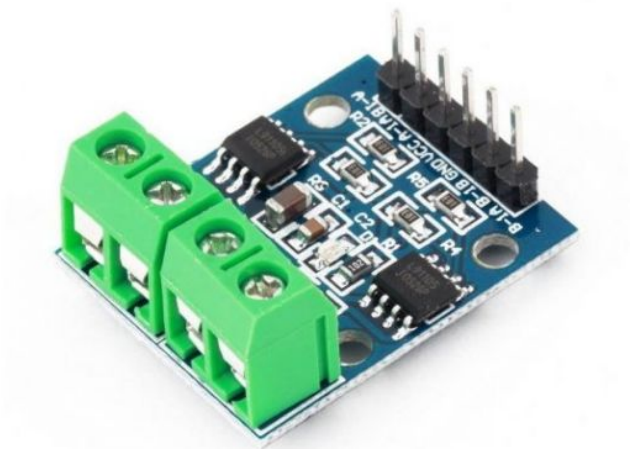


Image 4.0.1 DC Motor Transistor Driver board courtesy of robokits.net

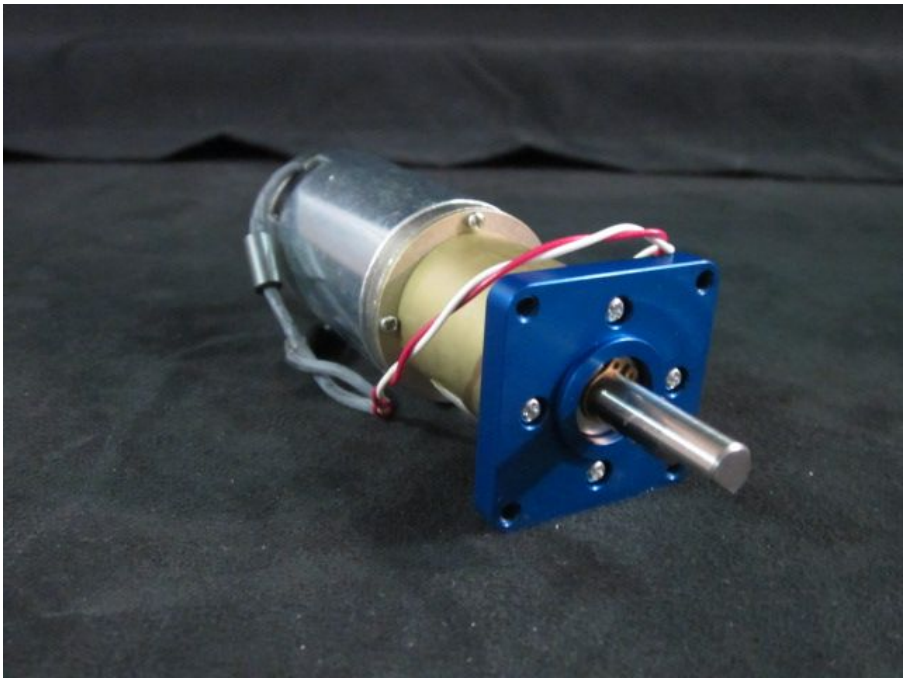


Image 4.0.2 DC Motor courtesy of capitolareatechnology

Time: Approximately 1-2 hours if all preparations are made

PROCEDURE

(10 mins) Step 1: Type in the following Python program (install python libraries if necessary)

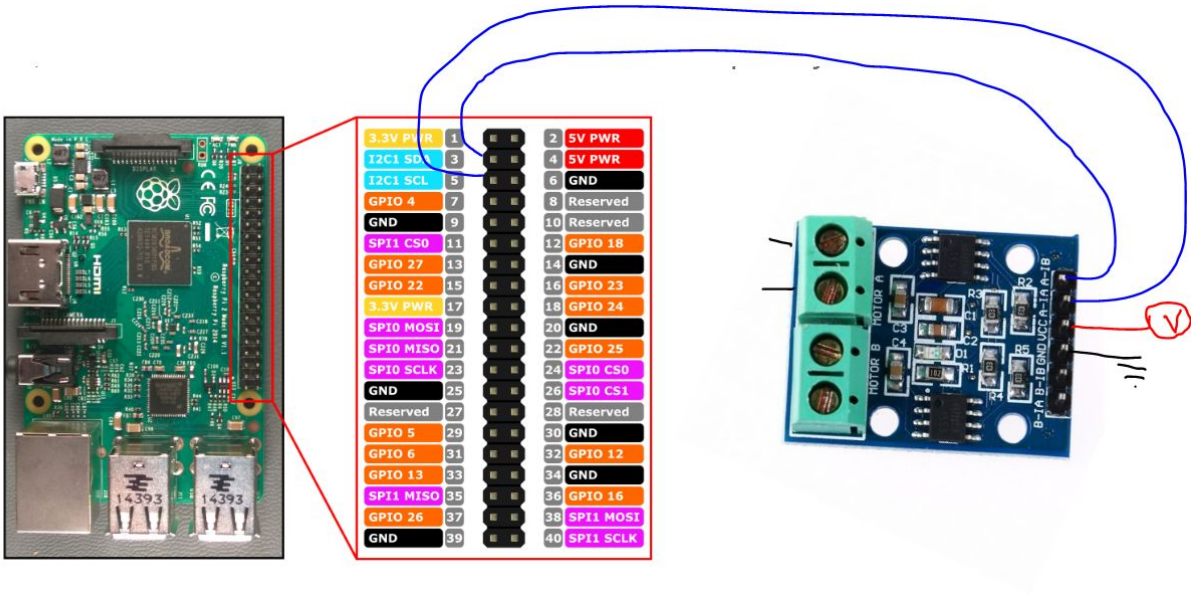
```
from Tkinter import *
```

```
import RPi.GPIO as GPIO

from time import sleep

import sys

GPIO.setmode(GPIO.BOARD)



Motor1A = 3

Motor1B = 5



GPIO.setup(Motor1A,GPIO.OUT)

GPIO.setup(Motor1B,GPIO.OUT)







main =Tk()



def d(event):

     GPIO.output(Motor1A,GPIO.HIGH)
```

```
     GPIO.output(Motor1B,GPIO.LOW)
     print "Going forwards\n"


def a(event):
     GPIO.output(Motor1B,GPIO.HIGH)
     GPIO.output(Motor1A,GPIO.LOW)
     print "Going Backwards\n"


def s(event):
     GPIO,cleanup()
     sys.exit(0)
     pass


main.bind('<Left>',a)
main.bind('<Right>',d)
main.bind('<Down>',s)
```

```
main.mainloop()
```

(20 mins)Step 2: Use jumper cables and wires to connect the Raspberry Pi to the transistor driver board and then the motor



(10 mins)Step 3: Bend Metal Rod to L - shape and add electric tape (for safety)

(15 mins)Step 4: Wrap Plumbing Heat shrink around rod and motor tip and heat with Heat gun

(20 mins)Step 5: Re-organize wires to make product portable

(05 mins)Step 6: Run the Previously Mentioned Code (python program.py)

3.2.3 LCD Display System

MATERIALS

- HD44780 LCD Display*
    - This LCD can either be acquired online through Amazon, or through spare parts shops like A1 Electronics. The ones in A1 electronics will be following a different pinout. More information in Step 2B.
- Jumper wires
- One of the following
    - PCB with traces from the GPIO pins of the Raspberry Pi to the LCD display or;
    - Breadboard

*Note: If you are to be using a Raspberry Pi it is ideal to use the HD44780 as other LCDs are designed to receive IO signals that the Pi cannot provide.

Time needed: 30 minutes or less

PROCEDURE

Step 1: If you use a brand new LCD instead of one from an old device (which is most likely soldered on to another board), it should take no less than 15 minutes to figure out the wiring. Otherwise, you will need to test which pin is which as the boards may possibly have different wiring. Researching your specific LCD might take longer than anticipated

Optional: Designing a PCB will eliminate the need for the breadboard and the jumper wires.

Step 2A: In assembling your LCD and numeric keypad, PiMyLifeUp.com has a fritzing diagram you can refer to. This is their recommended pinouts. Additionally if you do not have a 10k potentiometer or choose not to buy them you may opt to wire pin 3 (V0) directly to ground for maximum contrast.

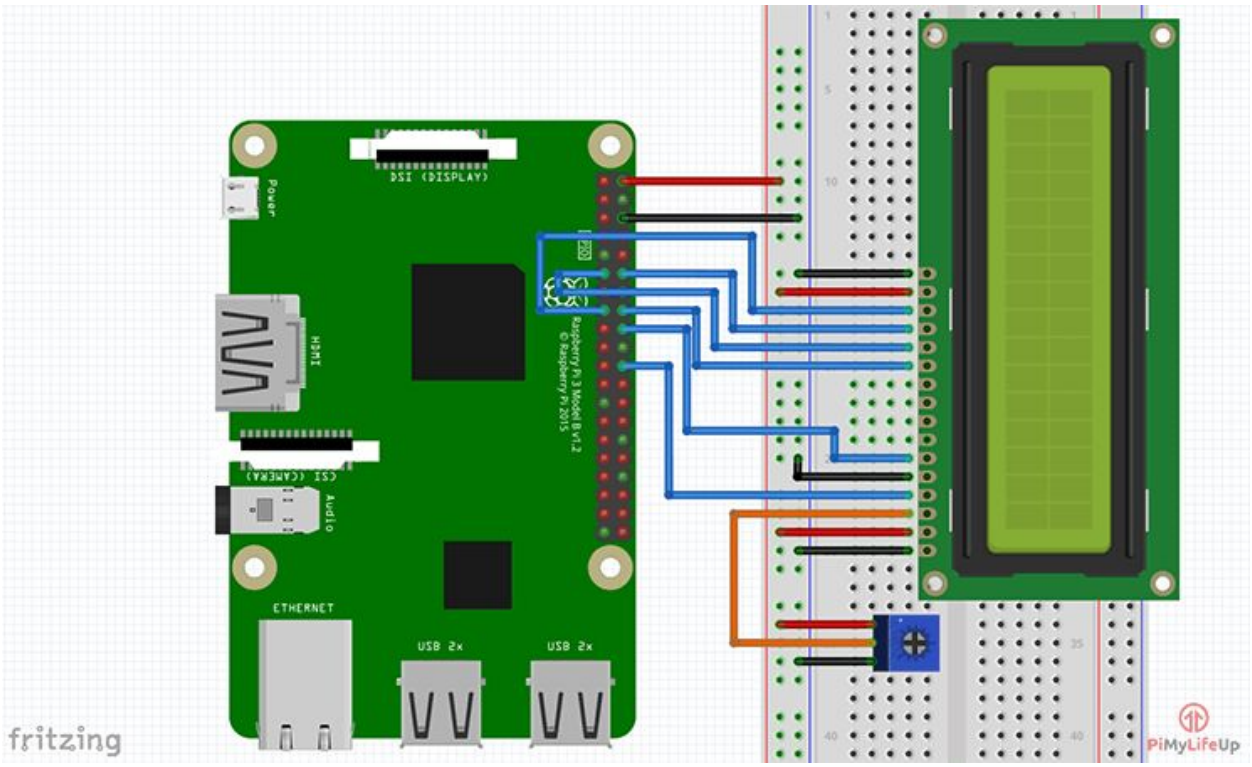


Image 3.2.3.1 Fritzing diagram for the HD44780 pinouts, courtesy of PiMyLifeUp.com

Step 2B: Alternatively, if you acquired the LCD below specifically from A1 Electronics, it is recommended you print the PCB instead as it follows a different pinout.

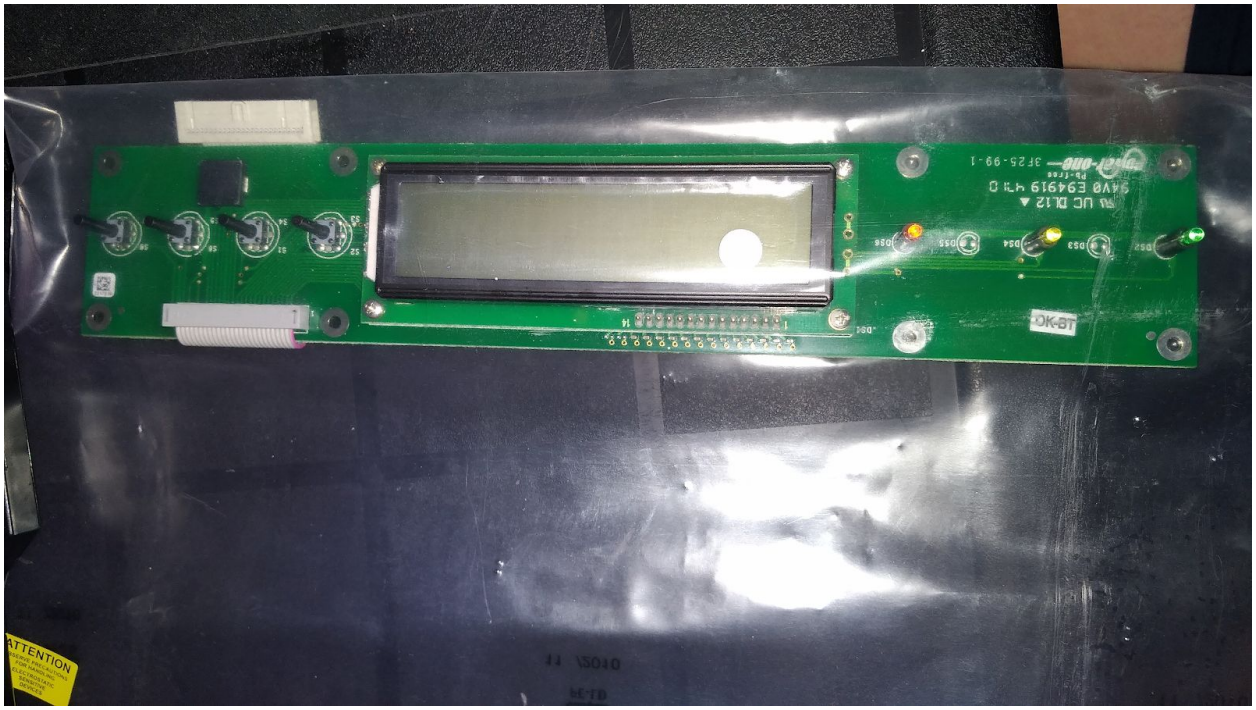


Image 3.2.3.2 LCD acquired from A1 Electronics.

You will need to solder on header pins and barrel pin connectors, but afterwards the LCD should work out of the box. Specifically you need the 40 pin male to female header to connect to the Raspberry Pi, and a 20 pin female to female header pin for connecting the LCD to the board.

The Fritzing Files for the PCB can be found here:

https://github.com/StanGo25/LockerAutomationSystem/tree/master/PCB%20Fritzing%20Files

If you choose not to print the PCB, you may follow along the pinouts below, courtesy of /u/devicemodder on Reddit

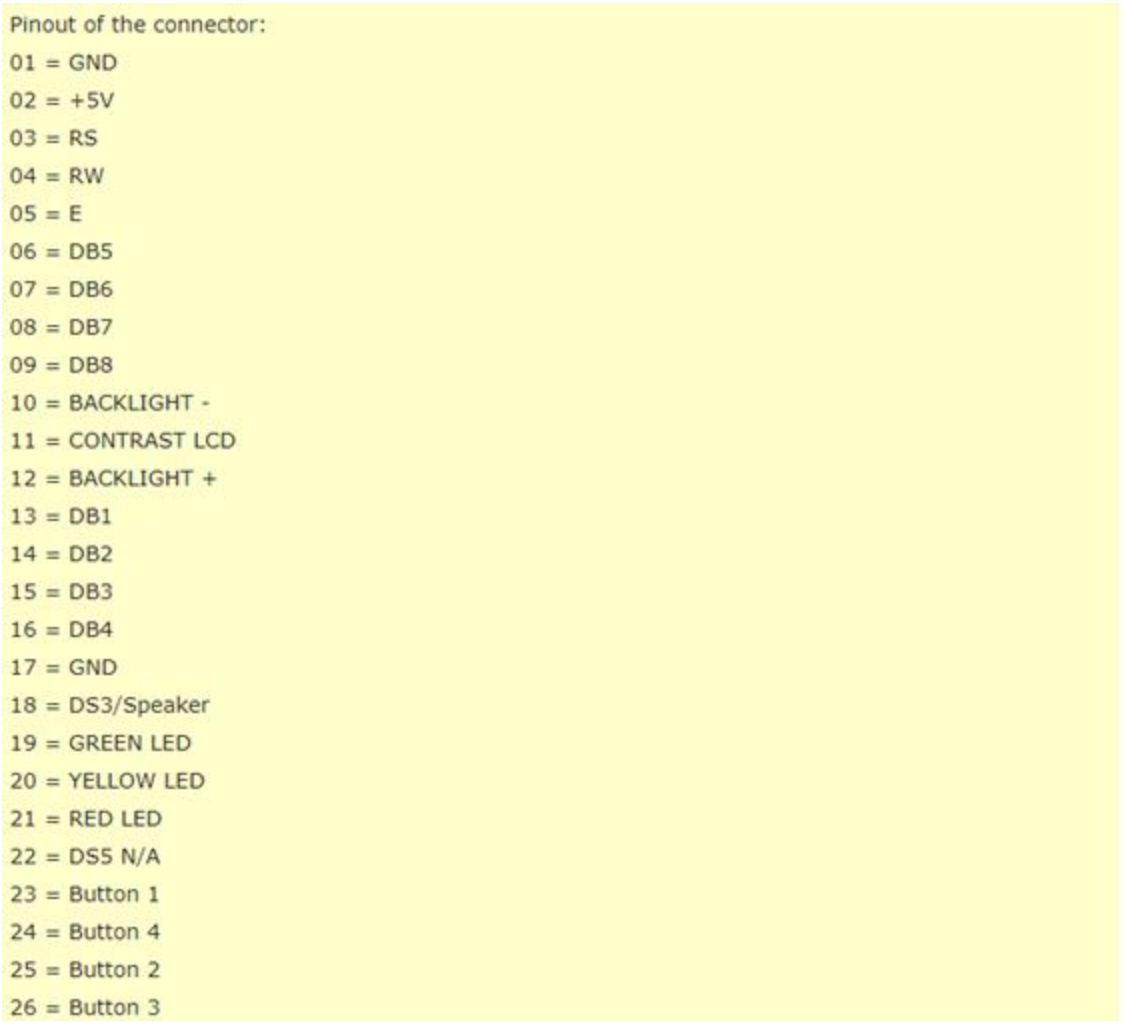


Image 3.2.3.3 Pinout for the ribbon cable soldered on to the LCD above. Note that for the purpose of powering up the LCD only pins 1-12 out of the 26 pins will be used.

Step 3: Assuming you have already installed the Raspbian OS, download **python** and **Adafruit's LCD library** and install them using the following commands:

sudo apt-get update

sudo apt-get install python

git clone https://github.com/adafruit/Adafruit_Python_CharLCD.git

Afterwards, navigate to where the library was downloaded to and then run

sudo python setup.py

From there, you can then use the following Python script as a template and an example to follow. If you used different GPIO pins change the value in the pin setup to the corresponding pin header number. Note that the Adafruit libraries have only been tested on the HD44780, and we cannot guarantee that the code would work with other models.

```
#!/usr/bin/python
# Example using a character LCD connected to a Raspberry Pi
import time
import sys
import datetime
import Adafruit_CharLCD as LCD

# Raspberry Pi pin setup
lcd_rs = 25
```

```
lcd_en = 24

lcd_d4 = 23

lcd_d5 = 17

lcd_d6 = 18

lcd_d7 = 22

lcd_backlight = 4


# Define LCD column and row size for 16x2 LCD.

lcd_columns = 16

lcd_rows = 2


lcd = LCD.Adafruit_CharLCD(lcd_rs, lcd_en, lcd_d4, lcd_d5,

lcd_d6, lcd_d7, lcd_columns, lcd_rows, lcd_backlight)


LockStat = 1


DoorStat = 2


time.sleep(0.005)


while True:

  lcd.message("Time\n" +
```

```
datetime.datetime.now().strftime("%I:%M%p"))

  time.sleep(5.0)

  lcd.clear()

  lcd.message("Date\n" + datetime.date.today().strftime("%b %d,
%Y"))

  time.sleep(5.0)

  while True:
    lcd.clear()
    lcd.message("Input your PIN\n")
    lcd.blink(True)

    PIN = raw_input()

    try:
      int(PIN)
    except ValueError:
      lcd.message("Not a number!")
```

```
    time.sleep(2)
    continue

  else:
    if(str(PIN) == "12345"):
      lcd.clear()
      lcd.blink(False)
      lcd.message("Access Granted\n*****")
      break
    if(str(PIN)=="0"):
   lcd.clear()
   lcd.blink(False)
   lcd.message("Goodbye")
   time.sleep(2)
   lcd.clear()
   sys.exit(0)
    else:
      lcd.message("Incorrect PIN")
      time.sleep(2)
   continue

time.sleep(3.0)
```

```
lcd.clear()
lcd.message("Welcome:\nStanley")

time.sleep(2.0)

lcd.clear()

if(LockStat == 1):
  lcd.message("Lock: Unlocked")
elif(LockStat == 0 ):
  lcd.message("Lock: Locked")

time.sleep(2.0)
lcd.clear()

if(DoorStat == 0):
  lcd.message("Door: Closed")

elif(DoorStat == 1):
  lcd.message("Door: Opening")
```

```
elif(DoorStat == 2):
  lcd.message("Door: Open")
time.sleep(2.0)

lcd.clear()
```

3.2.3 MySQL Database Instructions

The project uses a MySQL Database to store valuable data such as user information and product information. The user information table includes data on the user's name, a primary key ID, and a user's PIN. The product information includes the product's ID, product's name, the product's status and the user ID associated with that product. For this project, we used IBM Cloud's compose MySQL Database, which configured the database autonomously. After it has been set up, the following PHP scripts can be used with the server to communicate with the Android application.

The table fields should follow this naming scheme to ensure that the PHP scripts work with the database:

Table Name: users

- user_id
- user_name
- user_pin

Table Name: products

- product_id
- user_id
- product_name
- product_status

Table Name: actions

- product_id
- user_id
- action_id
- action_type

The source code for the scripts used to query the database can be found in this GitHub page (https://github.com/StanGo25/LockerAutomationSystem/blob/master/PHP%20Scripts/) :

Note that the code in the GitHub link has been modified and the connection settings have been removed to protect our database security and to prevent unauthorized access to the database.

3.2.4 Android application Build Instructions

The Android application is designed in a way that there is no database querying anywhere in the source code. Instead it runs PHP scripts hosted on an offsite server which the application then queries.

The source code can be found here: https://github.com/stango25/LockerAutomation

The application currently has English and French support, which is determined by the System Language of the device.

To access the source code:

Step 1: Download the whole repository, and place it in a folder.

Step 2: Open Android Studio as the file is an Android Studio module file

Step 3: Upon opening Android Studio, there will be an option to either create a new project or open a new project, unless you already have an existing project it is recommended that a new project be created specifically for this module to be imported into.

Step 4: Following the default settings, upon finishing set up of the project, in the menu bar, under File and under New, click on Import Module, and select LockerAutomation.iml.

Step 5: The module will then be imported into Android Studio. To modify the code for personal use so that it uses a different PHP script hosted on another server, change the URLs in the .java files.

3.2.4.1 Android Activity Breakdown

The Android application consists of 18 Java classes and 12 XML classes. Of the 18, DatabaseHandler.java, Actions.java, AzureServiceAdapter.java, Products.java and Users.java are unused in the present build and will be deprecated in future updates.

3.2.4.1.1 Login Activity

The LoginActivity.java class, alongside the activity_login.xml class, generates the layout for the main Login page. It consists of 2 buttons and 2 text fields, the 2 fields are for inputting the username and PIN if the user already has an account. After they input their login credentials, they can tap on the Login button to proceed to the Menu Activity, otherwise new users will need to tap on the Register button, which opens a pop up window with prompts for inputting the desired username and password. When the user taps on cancel they are brought back to the main activity, otherwise tapping register allows them to send this information to the database and to proceed to the Menu Activity. In navigating to the Menu Activity, the application first runs a PHP script that checks the user's login credentials and if the user is found to be within the database, the user is allowed to proceed, otherwise the user will need to input their PIN and Username once more.

3.2.4.1.2 Menu Activity

The MenuActivity.java class and activity_menu.xml class work hand in hand to generate the layout of the application's main menu. content_menu.xml, nav_header_menu.xml and app_bar_menu.xml are overlain on top of activity_menu.xml. The nav_header_menu defines the Navigation Drawer and its contents, with nav_header_menu being added on to app_bar_menu.xml, which defines the application bar of the activity. The content_menu.xml class defines the contents of the main page prior to the navigation drawer being opened. This class consists of a Floating Action Button which, when tapped, allows the user to log out. The devices' status is displayed via a TextView and the Door Control System button is placed in the main menu. In an earlier build the Door Control System and the Lock Control System were in individual Activities and had to be navigated to manually. As of now these Activities are still accessible if the user has already registered the devices, but to allow the user to have a more streamlined experience, the Door Control button in the main menu automatically disarms the lock and opens the door for the user. In the application bar, a quick access button for navigating to the Application's settings has also been placed.

3.2.4.1.3 QR Activity

The activity_qr.xml class provides the layout for that activity, or page and the Java class QRActivity.java generates the parameters defined by activity_qr.xml. BarcodeCaptureActivity, BarcodeTracker.java and BarcodeTrackerFactory.java are all used hand in hand to request for permissions to use the phone's camera and to scan

barcodes or QR Codes. The value corresponding to the QR code or barcode is then returned to BarcodeCaptureActivity.java and is overlain on a TextView field in QRActivity.java. If the code contained in the TextView matches the defined parameters for what constitutes a Locker Automation System product code, the device this code is tied to is registered to the user's account via a PHP script call and the information is sent to the MySQL database. From there the user will either have the option to open the Barcode Scanner Activity again to register another product or to tap on the back button to return to the main menu. Note that the Camera and Barcode .java classes were taken from Daniell Algar.

3.2.4.1.4 Door Activity

The XML class activity_door.xml provides the layout for the Door Activity. The DoorActivity.java class then uses the XML class' layout and generates the activity for the application. The primary feature in this activity is the Door Open / Close button. When tapped the button runs a PHP script in an offsite server and this script sends updates to the MySQL Database, dictating changes to be made to status of the Door Control Device registered to the user. This is done by defining an onClick listener method which is then attached to the button upon creating the Activity. This activity also contains a slider as the original intention was to allow the user to manually take control of the speed and direction that the door swings open or close. This feature has not been implemented and the slider will be removed in future builds of this application. Also note that in the current build that when the user returns to the Menu Activity and goes back to the Door Activity

the Door Control System's Button resets back to prompting the user to "Tap to Open" regardless if the physical Door Control Device is already set to open. As such it is recommended that users do not exit out of the activity until they have finished closing their Door using the Door Control System. In the future this activity is to be deprecated in favor of moving all Device Control to the Menu Activity.

3.2.4.1.5 Lock Activity

The XML class activity_lock.xml provides the layout for the Lock Activity. The LockActivity.java class then uses the XML class' layout and generates the activity for the application. The primary feature in this activity is the Lock Arm / Disarm image button. When tapped the button runs a PHP script in an offsite server and this script sends updates to the MySQL Database, dictating changes to be made to status of the Lock Control Device registered to the user. This is done by defining an onClick listener method which is then attached to the image button upon creating the Activity. Note that in the current build that when the user returns to the Menu Activity and goes back to the Lock Activity the Lock Control System's Button resets back to prompting the user to Disarm their lock regardless if the physical Lock Control Device is already set to disarmed or armed. As such it is recommended that users do not exit out of the activity until they have finished closing their Lock using the Lock Control System. In the future this activity is to be deprecated in favor of moving all Device Control to the Menu Activity.

3.2.4.1.6 Display Activity

The XML class activity_display.xml provides the layout for the Display Activity. The DisplayActivity.java class then uses the XML class' layout and generates the activity for the application. The primary feature in this activity is allowing the user to see the product codes for the devices registered to their account and to view the status of these products registered to their account. This is done by querying the database using a PHP script which takes the user's ID and returns the devices owned by the user whose ID matches the product entries. In future builds, we intend to move this to the Menu Activity to streamline the user experience as opposed to having the users navigate through multiple Activities, but the general implementation is going to be the same.

3.2.4.1.7 Settings Activity

The XML class activity_settings.xml, alongside the SettingsFragment.java class which uses the preferences.xml class, provide the layout for the Settings Activity. This is done to implicitly create selectable settings options for the Settings Activity. The SettingsActivity.java class then uses all these classes together and generates the activity for the application. The Settings Activity allows users to customize the theme of their application through the use of predefined themes. There is also an option to allow the user to modify their account, either to change their PIN or delete their account, but as of writing it has not been implemented yet. The user can tap on the "About Us" portion of the settings page, which brings them to our application's GitHub page.

3.2.4.2 Known Issues

In the current build at the time of writing, there are several bugs and issues in the application that, though do not require immediate addressing, may still make the user experience a poor and frustrating experience.

Examples include:

- In certain phone models (such as the LG G4), the application occasionally crashes when the Lock Button is tapped. We suspect that it might be an issue in memory but further testing will need to be done.
- Performing any actions involving the calling of the PHP scripts that communicate with the database lead to the application freezing for seconds at a time. We suspect that it might be an issue with the application waiting for the script to finish running and for the values to be returned before it performs the next action.
- Occasionally, the Display text in the Menu Activity does not update at all, possibly due to the fact that the issue above causes the text to not reflect on the changes that occured when the script is run.

3.2.4.3 Database Application Integration

In connecting the Android application to the Database, we call PHP scripts depending on the situation and the way this is implemented, we used AsyncTasks which run in the background and have them run either at the start of the activity or when the user taps on a button.

It is important to remember that all AsyncTasks involving database querying will fail and can potentially crash the app if the device is not connected to the Internet as not all AsyncTask implementations have error handling.

Examples of the AsyncTasks used can be found below:

```
   class GetProducts extends AsyncTask<String,Void,String> {
        @Override
        protected String doInBackground(String... strings) {
            String result = "";
            HttpURLConnection urlConnection = null;
            String validateUrl =
"http://munro.humber.ca/~n01116269/returnproducts.php?u_name=" +
pref.getString("u_name", "");
            try {
                URL url = new URL(validateUrl);
                urlConnection = (HttpURLConnection)
url.openConnection();
                InputStream is = new
```

```
BufferedInputStream(urlConnection.getInputStream());
                BufferedReader br = new BufferedReader(new
InputStreamReader(is));
                String line = "";


                while((line = br.readLine())!=null)
                {
                    result+=line;
                }
                is.close();
                Log.i("Result",result);
                return result;
            } catch (Exception e) {
                e.printStackTrace();
            }
            return null;
        }
    }
```

In the code snippet above, it queries the database and requests for the devices registered to the user by passing the username. The PHP script that is called then queries the database to check if there is a user with the same username. The ID is then used to query the products table for any product entries with that user ID. Afterwards the PHP script

returns a list of product types associated with the user. This is done so the application can then determine which Activities are to be made available to the user, the rest will be set so that they cannot be accessed.

It would look like this:

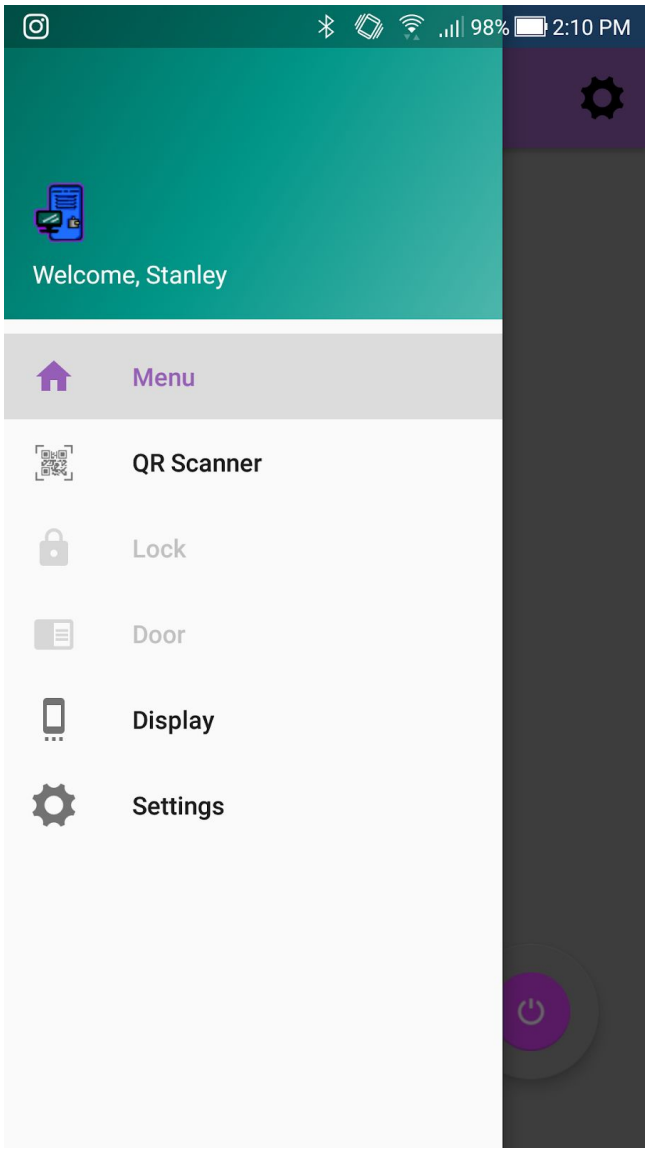


Image 3.2.4.3.1 Example of the application of the GetProducts AsyncTask. The user "Stanley" does not have any devices registered to his account and as such, because the GetProducts AsyncTask will not retrieve any values back and as such the application will disable the

activities. As such the only action the user "Stanley" can do is scan new devices using the QR Scanner.

To ensure user security, the application does not contain or obtain the user's ID in any form, the user ID stays within the PHP script.

Another example of an AsyncTask used by the application can be found below. This AsyncTask runs a PHP script which takes the user's name, queries the database to retrieve the user ID associated with the user's name. From there the PHP script queries the products table of the database to retrieve the status and product name of the products associated with the user's ID. The result of the PHP script is a preformatted String that follows the syntax "deviceName: (product status)". The tags are then stripped of the String and displayed on the TextView.

```
    class GetStat extends AsyncTask<String,Void,String> {
        @Override
        protected String doInBackground(String... strings) {
            String result = "";
            HttpURLConnection urlConnection = null;
            String validateUrl =
"http://munro.humber.ca/~n01116269/display.php?u_name=" +
pref.getString("u_name", "");
            try {
                URL url = new URL(validateUrl);
                urlConnection = (HttpURLConnection)
```

```
url.openConnection();
                InputStream is = new
BufferedInputStream(urlConnection.getInputStream());
                BufferedReader br = new BufferedReader(new
InputStreamReader(is));
                String line = "";
                while((line = br.readLine())!=null)
                {
                    result+=line;
                }
                is.close();
                Log.i("Result",result);
            } catch (Exception e) {
                e.printStackTrace();
            }
            StringBuilder sb = new StringBuilder();
            sb.append(result);
            String realRes =
sb.toString().replaceAll("\\<.*?>","");

            return realRes;
```

```
        }
    }
```

Some more examples of AsyncTasks can be found below, these are used to update the Door Control System's status and the Lock Control System's status.

```
   class setLock extends AsyncTask<String,Void,String> {
        @Override
        protected String doInBackground(String... strings) {

            String result = "";
            HttpURLConnection urlConnection = null;
            String validateUrl =
"http://munro.humber.ca/~n01116269/lock.php?p_stat=" +
strings[0]+ "&u_name=" + strings[1];

            try {
                URL url = new URL(validateUrl);
                urlConnection = (HttpURLConnection)
url.openConnection();
                InputStream is = new
BufferedInputStream(urlConnection.getInputStream());
                BufferedReader br = new BufferedReader(new
InputStreamReader(is));
```

```
                String line = "";
                while ((line = br.readLine()) != null) {
                    result += line;
                }
                is.close();
                Log.i("Result", result);
            } catch (MalformedURLException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
            return result;
        }
    }
```

In the AsyncTask above, the desired value is sent to the AsyncTask when its created, and it takes the first argument as the desired status. As this is for Lock it's either a value of 3 is set for armed or a value of 4 for disarmed. This, alongside the user's name, is sent to a PHP script which updates the Products table by taking the user's name, retrieving the user ID associated with it from the User table, and using the user ID to update all Lock Control Device type products associated with the user ID.

Meanwhile in the AsyncTask below, the desired value is passed to the AsyncTask which runs a different PHP script that updates the Door Control Device's status. It passes a

range of values from 0 - 2 depending on the desired status. Generally, the way the database and the scripts are designed, 1 stands for stationary, 0 for closing and 2 for opening.

```
    class setDoor extends AsyncTask<String,Void,String> {
        @Override
        protected String doInBackground(String... strings) {
            String result = "";
            HttpURLConnection urlConnection=null;
            String validateUrl =
"http://munro.humber.ca/~n01116269/door.php?p_stat=" +
Integer.parseInt(strings[0])+"&u_name="+
pref.getString("u_name", "");

            try {
                URL url = new URL(validateUrl);
                urlConnection
=(HttpURLConnection)url.openConnection();
                InputStream is = new
BufferedInputStream(urlConnection.getInputStream());
                BufferedReader br = new BufferedReader(new
InputStreamReader(is));
                String line ="";
```

```
            while((line=br.readLine())!=null)
            {
                result+=line;
            }
            is.close();
            Log.i("Result",result);
            return result;
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        return null;
    }
}
```

3.2.4.3.1 Source Code Breakdown db234.php

Below is the source code for db234.php. This is the script that is called by the Login Button. Looking at the MySQL statement in the code, one can observe that it queries the database for users with the username and PIN specified. If the PHP script successfully queries the database and finds a user matching the information provided, said user's name is returned to the application, from there the application compares the inputted text and the returned username.

```
<html>
  <head><title>Insert Title Here</title></head>
  <body>
    <?php
     // Open Connection
$con = @mysqli_connect('<host name>', '<username>',
'<password>', '<database name>', <port number>);

if (!$con) {
    echo "Error: " . mysqli_connect_error();
     exit();
}
$uname = $_GET['user_name'];
$upin = $_GET['user_pin'];
```

```
// Some Query
$sql      = "SELECT user_name FROM users where
user_name='$uname' and user_pin='$upin'";
$query    = mysqli_query($con, $sql);

while ($row = mysqli_fetch_array($query))
{
     $go = $row['user_name'];
     echo $go;

}

// Close connection
mysqli_close ($con);
?>
  </body>
</html>
```

3.2.4.3.2 Source Code Breakdown dbproducts.php

Below is the source code for dbproducts.php. This is the PHP script called by the QR Activity when it attempts to register device codes to the database. In the application side,

the code scanned is checked to see if it matches the coding scheme dictated, but on the PHP side the script accepts whatever the application lets through to it and inserts it to the Products table of the database.

```
<html>
  <head><title>Insert Title Here</title></head>
  <body>
    <?php
     // Open Connection
$con = @mysqli_connect('<hostname>', '<user name>',
'<password>', '<database name>', <port number>);

if (!$con) {
    echo "Error: " . mysqli_connect_error();
     exit();
}
$ptype = $_GET['p_type'];
$uname = $_GET['u_name'];
$pname = $_GET['p_name'];
//var_dump($pname);

// Some Query
$sql      = "SELECT user_id FROM users WHERE
```

```
user_name='$uname'";
//var_dump($sql);
$query    = mysqli_query($con, $sql);

while ($row = mysqli_fetch_array($query))
{
     $go = $row['user_id'];
     //echo $go;
}

$sql2 = "INSERT INTO products
(product_id,product_name,product_status,product_type,user_id)
VALUES (0,'$pname',0,'$ptype','$go')";
$query2 = mysqli_query($con, $sql2);
if ($query2)
{
     echo "Success";
}
// Close connection
mysqli_close ($con);
?>
```

```
  </body>
</html>
```

3.2.4.3.3 Source Code Breakdown display.php

Below is the code for display.php. This script is called by the Display Activity to obtain the status of the devices the user owns. This is done by querying the database for all products owned by the user after the user's user ID has been retrieved from the user's table. As can be seen below the text has already been formatted so that all the application needs to do when the results are returned is to strip away all the tags and set the usually empty TextView with the text from this script.

```
<html>
  <head><title>Owned Devices Status:<br></title></head>
  <body>
    <?php
     // Open Connection
$con = @mysqli_connect('<hostname>', '<user name>',
'<password>', '<database name>', <port number>);

if (!$con) {
    echo "Error: " . mysqli_connect_error();
     exit();
}
```

```
$uname = $_GET['u_name'];


//var_dump($pname);
$sql = "SELECT user_id FROM users WHERE user_name='$uname'";
$query = mysqli_query($con, $sql);
while ($row = mysqli_fetch_array($query))
{
     $uid = $row['user_id'];
}
$sql2 = "SELECT product_name,product_status FROM products where
user_id = '$uid' AND product_type IN ('1','2')";
$query2= mysqli_query($con, $sql2);
while ($row = mysqli_fetch_array($query2))
{
     $go = $row['product_name'] . ': ' . $row ['product_status']
. ' <br>';
     print ($go);
     //var_dump($go);
}
// Close connection
mysqli_close ($con);
?>
```

```
  </body>
</html>
```

3.2.4.3.4 Source Code Breakdown door.php

Below is the source code for door.php. This script takes the user's name and first retrieves the user ID from the Users table. From there, the script sends a query to the products table of the database, only this time updating all products of the Door Control System type that the user owns (ownership is determined through their user ID). Their status is changed according to the value that the application dictates, whether it be 1 for stationary, 0 for closing and 2 for opening.

```
<html>
  <head><title>Insert Title Here</title></head>
  <body>
    <?php
     // Open Connection
$con = @mysqli_connect('<hostname>', '<user name>',
'<password>', '<database name>', <port number>);

if (!$con) {
    echo "Error: " . mysqli_connect_error();
     exit();
}
$pstat = $_GET['p_stat'];
```

```
$uname = $_GET['u_name'];



//var_dump($pname);
$sql = "SELECT user_id FROM users WHERE user_name='$uname'";
$query = mysqli_query($con, $sql);
while ($row = mysqli_fetch_array($query))
{
     $uid = $row['user_id'];
}

$sql2 = "UPDATE products SET product_status = $pstat WHERE
user_id = $uid AND product_type = '2'";

var_dump($sql2);

$query2= mysqli_query($con, $sql2);

// Close connection
mysqli_close ($con);
?>
```

```
  </body>
</html>
```

3.2.4.3.5 Source Code Breakdown lock.php

Below is the source code for lock.php. This script takes the user's name and first retrieves the user ID from the Users table. From there, the script sends a query to the products table of the database, only this time updating all products of the Lock Control System type that the user owns (ownership is determined through their user ID). Their status is changed according to the value that the application dictates, 3 for disarm or 4 for armed.

```
<html>
  <head><title>Insert Title Here</title></head>
  <body>
    <?php
     // Open Connection
$con = @mysqli_connect('<hostname>', '<user name>',
'<password>', '<database name>', <port number>);

if (!$con) {
    echo "Error: " . mysqli_connect_error();
     exit();
}
```

```
$pstat = $_GET['p_stat'];
$uname = $_GET['u_name'];



//var_dump($pname);
$sql = "SELECT user_id FROM users WHERE user_name='$uname'";
$query = mysqli_query($con, $sql);
while ($row = mysqli_fetch_array($query))
{
     $uid = $row['user_id'];
}

$sql2 = "UPDATE products SET product_status=$pstat WHERE user_id
= $uid AND product_type ='1'";

//var_dump($sql2);

$query2= mysqli_query($con, $sql2);

// Close connection
mysqli_close ($con);
?>
```

```
  </body>
</html>
```

3.2.4.3.6 Source Code Breakdown returnproducts.php

Below is the source code for returnproducts.php. This script queries the database to check for all device types that the user owns. This is done by taking the user's name, retrieving the user ID from the user's database for the user with a name that matches that ID. From there the product types are sent back to the application. Using the product type codes, the application then determines which Activities are to be made available to the user.

```
<html>
  <head><title>Insert Title Here</title></head>
  <body>
    <?php
     // Open Connection
$con = @mysqli_connect('<hostname>', '<user name>',
'<password>', '<database name>', <port number>);

if (!$con) {
    echo "Error: " . mysqli_connect_error();
     exit();
}
```

```
$ptype = $_GET['p_type'];
$uname = $_GET['u_name'];
$pname = $_GET['p_name'];
//var_dump($pname);
$sql = "SELECT user_id FROM users WHERE user_name='$uname'";
$query = mysqli_query($con, $sql);
while ($row = mysqli_fetch_array($query))
{
     $uid = $row['user_id'];
}
$sql2 = "SELECT product_type FROM products where user_id =
'$uid'";
$query2= mysqli_query($con, $sql2);
while ($row = mysqli_fetch_array($query2))
{
     $go = $row['product_type'] . ' ';
     echo $go;
     //var_dump($go);
}
// Close connection
mysqli_close ($con);
?>
```

```
  </body>
</html>
```

3.2.4.3.7 Source Code Breakdown userinsert.php

Arguably one of the most important of PHP scripts that the Android application uses, userinsert.php, as its filename suggests, allows users to create new accounts which are saved to the database. This is done by taking the user name and user PIN from the Registration fields and inserting a new User in the Users table in the database. The user ID is automatically generated by the database as predefined by the database administrator.

```
<html>
  <head><title>Insert Title Here</title></head>
  <body>
    <?php
     // Open Connection
$con = @mysqli_connect('<host name>', '<username>',
'<password>', '<database name>', <port number>);

if (!$con) {
    echo "Error: " . mysqli_connect_error();
     exit();
}

$uname = $_GET['user_name'];
```

```
$upin = $_GET['user_pin'];
//var_dump($pname);

// Some Query
$sql      = "INSERT INTO users (user_id, user_name, user_pin)
values (0, '$uname',$upin)";
//var_dump($sql);
$query    = mysqli_query($con, $sql);

if ($query){

     echo "Success";
}
// Close connection
mysqli_close ($con);
?>
  </body>
</html>
```

3.2.5 System Control Python Script

The main software running on the Raspberry Pi microcontroller was developed in the form of a specific Python script designed to listen for changes on the database through a PHP script which would then activate the GPIO pins to activate the corresponding devices. The script also contains parameters to time the opening and closing of the door which will have to be configured depending on the physical installation that is present on-site.

3.2.5.1 System Control Python Script source code (lockerautoscript.py)

```
import
urllib2,
urllib
            import time
            import sys
            import datetime
            import re
            import wiringpi
            import Adafruit_CharLCD as LCD
```

```
lock = 'TestLock00'
door = 'JI13444983'


currentlock = '3'
currentdoor = '2'


wiringpi.wiringPiSetup()


lcd_columns = 16
lcd_rows = 2


lcd_rs = 10
lcd_en = 05
lcd_d4 = 27
lcd_d5 = 11
lcd_d6 = 22
lcd_d7 = 9
lcd_backlight = 4
```

```
lcd = LCD.Adafruit_CharLCD(lcd_rs, lcd_en, lcd_d4,
lcd_d5, lcd_d6, lcd_d7, lcd_columns, lcd_rows,
lcd_backlight)



#Pin Definitions


LOCKPIN1 = 24
LOCKPIN2 = 25
MOTORF = 22
MOTORB = 23


#Function Definitions


def doorOpen():
```

```
    wiringpi.pinMode(LOCKPIN1,1)
    wiringpi.pinMode(LOCKPIN2,1)
    wiringpi.pinMode(MOTORF,1)
    wiringpi.pinMode(MOTORB,1)
    print "Door Unlocked"
    wiringpi.digitalWrite(LOCKPIN1,1)
    wiringpi.digitalWrite(LOCKPIN2,0)
    print "ALERT! DOOR MOVING!"
    wiringpi.digitalWrite(MOTORF,1)
    wiringpi.digitalWrite(MOTORB,0)
    time.sleep(3)
    wiringpi.digitalWrite(LOCKPIN1,0)
    wiringpi.digitalWrite(LOCKPIN2,0)
    wiringpi.digitalWrite(MOTORF,0)
    wiringpi.digitalWrite(MOTORB,0)
    print "DOOR STOPPED!"


def doorClose():
    wiringpi.pinMode(LOCKPIN1,1)
    wiringpi.pinMode(LOCKPIN2,1)
    wiringpi.pinMode(MOTORF,1)
```

```
    wiringpi.pinMode(MOTORB,1)
    print "Door closing"
    wiringpi.digitalWrite(MOTORF,0)
    wiringpi.digitalWrite(MOTORB,1)
    print "ALERT! DOOR MOVING!"
    time.sleep(3)
    wiringpi.digitalWrite(MOTORF,0)
    wiringpi.digitalWrite(MOTORB,0)
    print "DOOR STOPPED!"


def lcdDisplay(arg1, arg2):
    lcd.clear()
    lcd.message(arg1+'\n'+arg2)
    time.sleep(2)


#main
while True:
    # 'Checking lock status'
```

```
path='http://munro.humber.ca/~n01137746/display.php
?pname='+lock

   req=urllib2.Request(path)
   req.add_header("Content-type",
"application/x-www-form-urlencoded")


   page=urllib2.urlopen(req).read()


   databaseval = re.sub("<.*?>", "", page)
   databaseval = databaseval.replace('\n','')
   databaseval = databaseval.replace(":
","":\n")
   databaseval = databaseval.strip(' ')


   lockval = ''
```

```
   if databaseval == '3' and databaseval ==
currentlock:
       lockval = 'Lock is armed'
   elif databaseval=='3' and databaseval !=
currentlock:
       lockval = 'Lock has been armed by the
application'
       currentlock = databaseval
   elif databaseval=='4' and
databaseval==currentlock:
       lockval = 'Lock is disarmed'
   elif databaseval=='4' and databaseval !=
currentlock:
       lockval = 'Lock has been disarmed by the
application'
       currentlock = databaseval



   # Checking door status

path='http://munro.humber.ca/~n01137746/display.php
?pname='+door
```

```
    req=urllib2.Request(path)
    req.add_header("Content-type",
"application/x-www-form-urlencoded")


    page=urllib2.urlopen(req).read()


    databaseval = re.sub("<.*?>", "", page)
    databaseval = databaseval.replace('\n','')
    databaseval = databaseval.replace(":
",":\n")
    databaseval = databaseval.strip(' ')


    doorval = ''

    if databaseval == '1' and currentdoor == '2':
        doorval = 'Door is closed'
        lcdDisplay(doorval,lockval)
    elif databaseval == '1' and currentdoor == '0':
```

```
        doorval = 'Door is open'
        lcdDisplay(doorval,lockval)
    elif databaseval == '2':
        doorval = 'Door is opening'
        currentdoor = '0'
        lcdDisplay(doorval,lockval)
        doorOpen()
    elif databaseval == '0':
        doorval = 'Door is closing'
        currentdoor = '2'
        lcdDisplay(doorval,lockval)
        doorClose()

    time.sleep(0.05)
```

4.1 Conclusions

To assist those who are either physically impaired or occupied with bulky or heavy items, a system was made to enable locker users to unlock and open their lockers from afar and at a simple tap of a button. The system's main feature was a door opening mechanism and a smart lock mechanism. A display with buttons was placed to facilitate manual unlocking. The door opening mechanism and the smart lock mechanism relied on information from an offsite database to perform tasks, which was updated using either an Android application or through the display and its buttons. Both the application and the display had security features implemented through the use of an account with a password and through this account, the user had their own devices attached to it. The database contained user account information and user device information, as well as the statuses of the devices. This system has potential to be used in all sorts of workplaces, including schools, to simplify the user's locker experience and monitor their lockers anywhere and anytime, so long as they are connected to the internet.

4.2 Recommendations:

In bringing the Locker Automation System to full scale production, changes have to be made. For one, the Android application will need to be reworked to provide a better user experience, the known issues listed above will need to be addressed.

Parts will need to be sourced elsewhere, preferably directly from manufacturers instead of through third party retailers. The devices would also benefit from an enclosure that would protect the devices from the elements.

Ultimately, the database and the PHP scripts will need to be hosted in the same server, one that we would have constant access to. As of writing, the scripts are hosted on Humber College Institute of Technology and Learning's servers and upon completion of the program we will lose all access to the files. And the database is hosted on IBM BlueMix, which requires a fee to maintain the servers. As such to pursue full scale production, retail price adjustments will need to be made to accommodate the additional fees the software side of the System will incur.

5 Bibliography

Adafruit Industries. (2018). Lock-style Solenoid - 12VDC. Adafruit.com. Retrieved 22 November 2017, from https://www.adafruit.com/product/1512

Arduino and Power-One 3F25-99-1 LCD module • r/electronics. (n.d.). Retrieved from https://www.reddit.com/r/electronics/comments/5uc0tr/arduino_and_powerone_3f25991_lcd_module/ddswkdm

Capitol Area Technology. 409A751. Retrieved from http://pics.capitolareatechnology.com/409A751.JPG

Medri, K. (2018). Engineering Technology. Six0four.github.io. Retrieved 29 January 2018, from https://six0four.github.io/ceng355/

Rada, Z., & 라다자카리아. (2016). GPIO on Raspberry Pi 3 B+ model. Research Gate. Retrieved 9 February 2018, from https://www.researchgate.net/figure/GPIO-on-Raspberry-Pi-3-B-model_fig2_312218161

Raspberry Pi LCD Tutorial. (2017, November 06). Retrieved from https://pimylifeup.com/raspberry-pi-lcd-16x2/

Robokits. h_bridge_1. Retrieved from

http://www.robokits.net/pub/media/catalog/product/cache/1/image/700x560/e9c3970ab036de70892d86c6d221abfe/h/-/h-bridge_1.jpg

6 Appendices

6.1 Definition of Terms

Android – refers to the operating system developed by Google, based on Linux Architecture.

Locker - a small lockable closet or compartment.

Automation – act or process of automation

Lock - a contrivance for fastening or securing something.

Door - a movable, usually solid, barrier for opening and closing an entranceway.

Display - to show or exhibit; make visible.

Database - a comprehensive collection of related data organized for convenient access, generally in a computer.

Dallas – a city located in Texas, United States of America, North America, Planet Earth.

IBM – Our database and cloud services provider.

Phone – a portable electronic telephone device.

Application – software used by Android applications.