**Automated Entry System with Graphical Display Operating as Roomi**

Prepared by

Denald Demirxhiu

Jacob Ladan

Marko Javorac

Humber College Institute of Technology and Advanced Learning

Discipline: Computer

January 24, 2019

# 

# **Declaration of Group Authorship**

Roomi hereby certifies that this technical report we are submitting is entirely our own original work except where otherwise indicated. We are aware of the College’s regulations concerning plagiarism, including those regulations concerning disciplinary actions that may result from plagiarism. A detailed breakdown of the individual work to be completed for this project can be found in *Figure 1 (pg. 6)*, as per the list of illustrations.

Student #1 Name: **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Signature**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Student #2 Name: **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Signature**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Student #3 Name: **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Signature**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Date of Submission: **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# 

# **Abstract**

This project combines the sum of our skills and knowledge gained through the first five semesters of the Computer Engineering Technology program at Humber College. As a result, the final product will encompass a door latching mechanism with NFC/RFID reader, graphical display, cloud storage database, and an accompanying Android application. The goal is to give users a modern approach to security systems for residential and commercial dwellings. This project demonstrates our extensive knowledge of both hardware and software. This technical report will outline the complete process of research, construction, testing, and implementation of each aspect of the project.

# 

# **Introduction**

Today, many rooms being electronically secured lack the flexibility of information display and mobile user management. The problem herein lies with the decrease in usability, and the sole focus of Roomi is to remedy this. Through the tying together of an LCD screen and existing RFID technology, users will be able to view additional important information about secured rooms, as well as allow administrators to update this information easily and without the need for support from IT.

A 2.4" Digole LCD Display will be used to display relevant information unique to every room. A PN532 NFC/RFID reader will allow individuals access to configured rooms, based on a predefined access level setup through the Android application. A mini push-pull solenoid will be used to demonstrate how the locking mechanism will function.

The mobile application will allow the administration to modify user and room access and display settings in a user-friendly environment. The hardware and application will work in unison with a cloud service hosted on Google's Firebase platform to store relevant information for rooms, user authentication, and personnel registered through the application.

*Professor Notes: Consider scope: Finished product preventing physical access to rooms, or a prototype?*

Table of Contents

[Declaration of Group Authorship i](#_Toc4083508)

[Abstract iii](#_Toc4083509)

[Introduction iv](#_Toc4083510)

[List of Illustrations 1](#_Toc4083511)

[Requirements Specification 3](#_Toc4083512)

[Software 3](#_Toc4083513)

[Mobile Application 4](#_Toc4083514)

[Database 5](#_Toc4083515)

[Firmware 8](#_Toc4083516)

[Hardware 9](#_Toc4083517)

[Development Platform 9](#_Toc4083518)

[Interface boards and sensors 10](#_Toc4083519)

[Other Accessories and Enclosure 11](#_Toc4083520)

[Build Instructions 13](#_Toc4083521)

[Budget 13](#_Toc4083522)

[Time Commitment 13](#_Toc4083523)

[Mechanical Assembly and Bread Boarding 14](#_Toc4083524)

[Digole 2.4” LCD 14](#_Toc4083525)

[NFC Reader 16](#_Toc4083526)

[Mini Push-Pull Solenoid 19](#_Toc4083527)

[PCB and Soldering 20](#_Toc4083528)

[Digole 2.4” LCD 20](#_Toc4083529)

[NFC Reader 21](#_Toc4083530)

[Mini Push-Pull Solenoid 21](#_Toc4083531)

[Software 22](#_Toc4083532)

[Digole 2.4” LCD 22](#_Toc4083533)

[NFC Reader 24](#_Toc4083534)

[Mini Push-Pull Solenoid 24](#_Toc4083535)

[Enclosure 24](#_Toc4083536)

[Application 26](#_Toc4083537)

[Database 26](#_Toc4083538)

[Conclusion 27](#_Toc4083539)

[Bibliography 29](#_Toc4083540)

[Appendix 31](#_Toc4083541)

# **List of Illustrations**

[Figure 1 - Work Breakdown Structure 3](#_Toc4085132)

[Figure 2 - Application Flow 4](#_Toc4085133)

[Figure 3 - Database Structure 7](#_Toc4085134)

[Figure 4 - Access Level Representation 8](#_Toc4085135)

Do this stuff for build instructions

# 

# 

# **Requirements Specification**

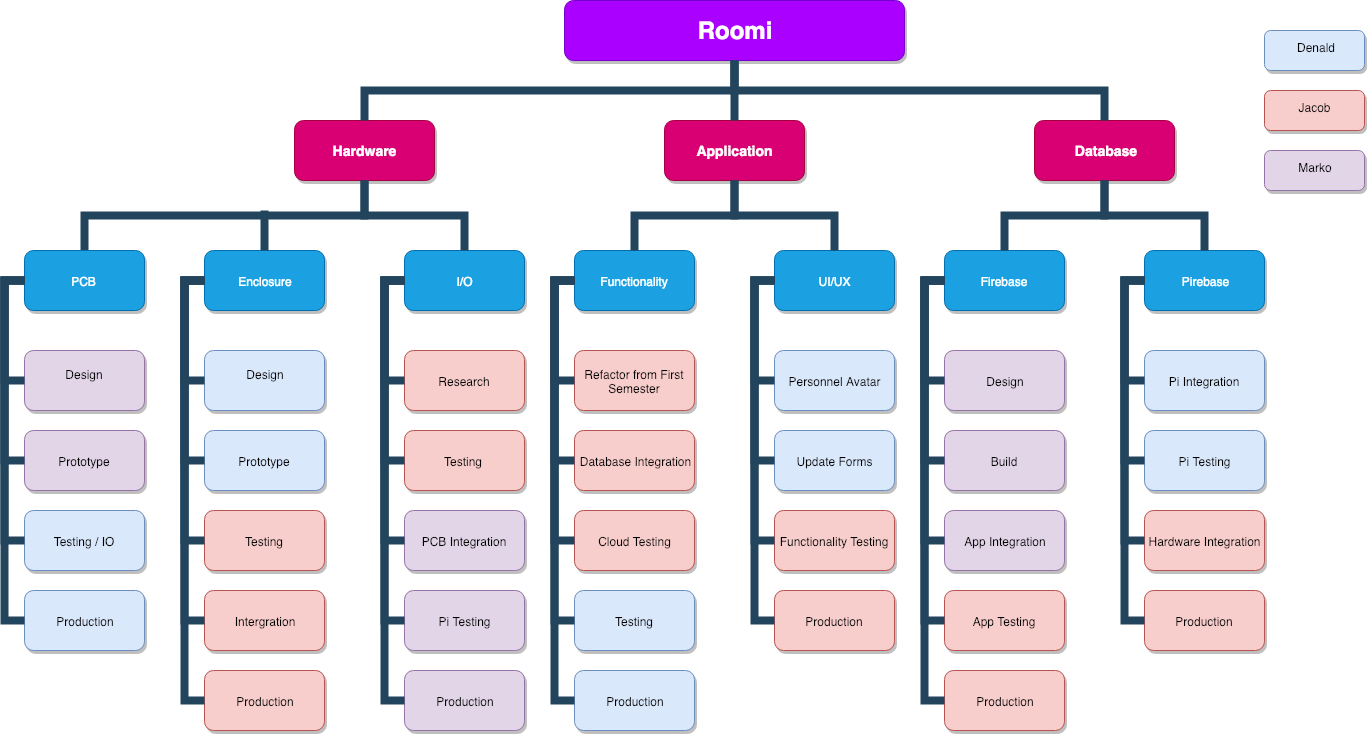


Figure 1 - Work Breakdown Structure

## Software

In this section, we will describe the requirements for the application and the database that will work in parallel with the hardware for the capstone project of the Computer Engineering Technology program at Humber College. These requirements will outline the specifications of the Broadcom development platform as well as the three sensors/effectors that will be interfaced.

### Mobile Application

#### Purpose

This application will give administrators the utilities needed to manage room and lab access as well as display relevant information regarding each individual room and lab. A visual representation of the application’s flow can be found in Figure 1.

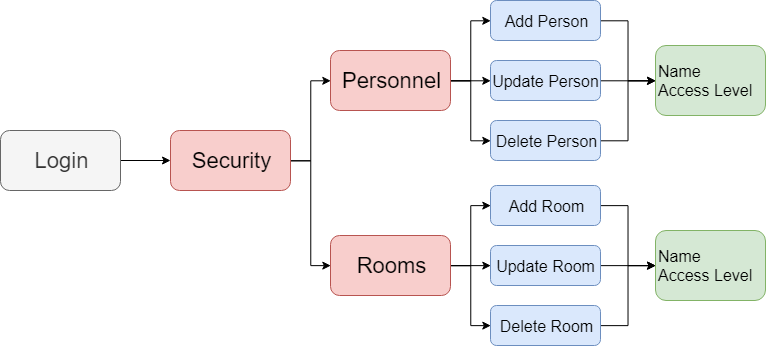


Figure 2 - Application Flow

#### Operating Environment

This application will operate on a mobile device running Android OS updated to at least version 5.0 (Lollipop). User settings will be delivered to a cloud service database (Firebase) through which a Raspberry Pi 3B+ will be also communicating. The Raspberry Pi will perform the necessary actions to control room, user, and personnel access/traits.

#### Assumptions and Dependencies

The most significant dependencies of this mobile application include the server-side database (where the data will be stored), the actual hardware it is going to communicate with, and time as the time constraint we have is restricted to 10 weeks. These constraints do affect the development time and the list of features that are going to be implemented, therefore in the final version some of the features might not be fully integrated and compatible, but the functionality will be implemented. There might be feature drop as a result of the time constraint, which can be implemented at a later version of the application.

### Database

#### Purpose

The database will be used to store relevant room information for rooms that have been registered with the application, as well as information about employees/students who have been registered with RFID cards. Additionally, administrators using the application to configure rooms will be required to log in, as a result, account information will be stored.

#### Operating Environment

The database will be hosted on Google’s Firebase cloud service. The information will be stored in JSON format, utilizing RESTful architecture as a NoSQL structure. User authentication information will be stored using Firebase’s authentication database, which allows us to ensure obfuscation of sensitive information such as email and password.

#### Assumptions and Dependencies

Due to the nature of our application and its hardware, the most significant dependency going forward is the requirement of having 100% connectivity to the internet. In order for the hardware, and the application to function accordingly, they will both need to be connected to our cloud service at all times. Additionally, we are utilizing Firebase’s free plan to store data. As such, in order to not incur additional charges or reduce functionality for the user, we will need to limit data rates to the specified plan.

#### Database Structure

At the top level, only one branch is presented, each child holding a unique string value for registered users. Continuing, each user node holds the pertinent information used for authentication as well as two branched nodes; rooms and personnel. The rooms branch holds all the information associated with the user’s rooms that have been set up through the application (access level and name). The personnel branch, similar to the rooms branch, holds all the information associated with the personnel set up by the user through our application (name, access level, and avatar colour). A visual representation of the database’s structure and the functionality of the access level can be found in Figure 2 and 3 respectively.

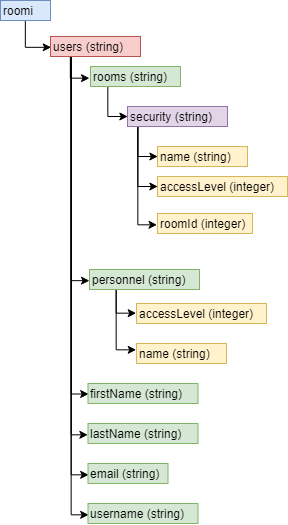


Figure 3 - Database Structure

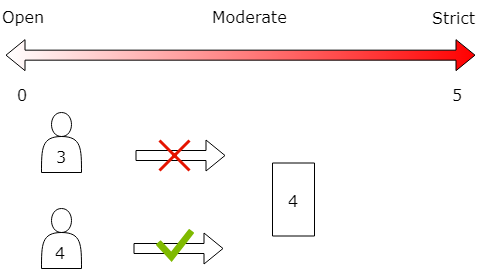


Figure 4 - Access Level Representation

### Firmware

### Purpose

The purpose of the firmware will be to provide the software platform of communication between the hardware, database, and mobile application. The firmware will also provide the user interface at the point of contact.

#### Operating Environment

The operating environment is called Raspbian. Raspbian is a free operating system based on Debian GNU/Linux and optimized for the Raspberry Pi hardware.

#### Assumptions and Dependencies

It is critical for Raspbian to receive continual support throughout its planned lifetime. This is important for security and I2C driver compatibility.

## Hardware

In this section, we will describe the hardware specifications for the development of the capstone project. These requirements will outline the specifications of the 3 unique sensors, the Broadcom Development Platform (Raspberry PI 3B+), and system enclosure. The functionalities of all these components will be tightly coupled together to provide seamless communication between the hardware operations and the software aspect of the project.

### Development Platform

#### Purpose

The development platform that is going to be the main focus of the entire project is the Raspberry PI 3B+. It offers a wide variety of features including the ability to communicate with sensors and actuators via I2C, a fast CPU for handling various tasks, 40 general input/output pins that can be used as the main way of communication with external components, and more. The purpose of this device is to process the information retrieved from the RFID sensor mounted on to it and based on the data stored on the database make decisions by triggering the push-pull solenoid to act as a door lock. Additionally, it will display relevant information to the Digole display as a way to communicate with users of the device.

#### Operating Environment

As it was mentioned above, the Raspberry Pi will be used as the central board to connect to and communicate with the respective sensors/actuators. The board will be mounted on the wall where it will provide a way to interact with RFID cards. The cards will be used to grant access to specific rooms whose access level information has been set by the appropriate administrators. With the exception of the mini push-pull solenoid, the RFID sensor and the Digole display will be mounted on the Raspberry PI, which will allow a level of interactivity with the whole system.

#### Assumptions and Dependencies

The major dependency of the hardware will be power, which will be assumed to be supplied at all times. The power to each of the components will be provided by the Raspberry PI itself, however, the power for the Raspberry PI will be supplied by the micro USB charger. The operation of the entire system also depends on a reliable network that ensures communication with the database to make the appropriate decisions at all times.

### Interface boards and sensors

The main sensors/actuators that are going to be used in the making of the entire project include the Digole LCD Display, the PN532 NFC reader/writer, and the mini push-pull solenoid. The purpose of each sensor is described briefly below.

#### Digole LCD touch screen

The purpose of the LCD screen will be to display pertinent information related to the room it is associated with. The information will include the room’s name and access level. Additionally, the LCD screen will display general information such as the date. The LCD will be implemented using a Raspberry Pi 3B+. The Pi will connect to the internet using wifi and communicate directly with the database retrieving and updating any information necessary for display.

#### PN532 NFC reader/writer

The purpose of the NFC reader/writer will be to control access to the room it is associated with. Along with the LCD screen, the NFC will be implemented using a Raspberry Pi 3B+. The Pi will connect to the database using wifi to retrieve the access level associated with the room. Knowing this information the NFC will be able to grant access if the personnel’s access level is equal to or greater than the room’s access level or deny them if it is lower.

#### Mini Push-Pull Solenoid

The purpose of the mini push-pull solenoid will be to demonstrate the mechanism that will deal with locking and unlocking doors.

### Other Accessories and Enclosure

The development platform will require a power source, display, and a network connection. The enclosure should be designed in CorelDraw and laser cut from acrylic. It may be further improved by 3D printing (responsibility: Mechanical Engineering Technology collaborators).

## 

# 

# **Build Instructions**

Repository: https://github.com/roomi-develop/roomi

## Budget

A full list of materials along with a detailed view of costs can be downloaded from within the project’s repository. The total cost of producing this prototype is heavily inflated due to the cost of the soldering kit that was supplied in the lab during development. Any generic soldering iron can be used for this project.

The total cost after removing the soldering kit is: $474.66 after taxes. This includes all the tools used in completing the prototype (e.g. wire cutters, needle nose pliers, breadboard, etc.).

Notable purchases include: Raspberry Pi 3B+ Kit ($94.95 CAD), Digole 2.4” LCD Display with Touch Capabilities ($17.49 USD), RFID/NFC Reader ($39.95), and the Mini Push-Pull Solenoid ($21.23).

## Time Commitment

A detailed view of the schedule followed for this project can be downloaded from within the project’s repository. The schedule uses a weekly breakdown that follows the CENG 317 and CENG 355 class schedules for fall 2018 and winter 2019. This prototype could be completed in a more compact time frame if the builder so chooses. The schedule is helpful in outlining the overall flow and the order in which each milestone for the project is completed. Originally, the project was completed over the 30 weeks needed for the school year. However, it could more reasonably completed from within 2-4 weeks dependent on how many of the materials the builder already possesses. Access to the facilities necessary for producing the PCG/case, and shipping times.

## Mechanical Assembly and Bread Boarding

### Digole 2.4” LCD

Firstly, in order for the Digole LCD to use I2C communication with the Pi, one modification must be made to the screen's logic board. The builder must solder a short between the middle and the I2C adapters as outlined in the image below.



Figure 5: Digole LCD I2C Jumper Configuration

Next, the following connection scheme will be used in order to connect the Digole LCD to the Raspberry Pi's GPIO pins.

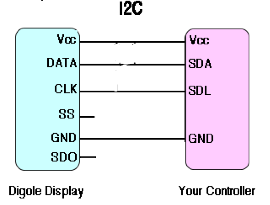


Figure 6: Digole LCD and Raspberry Pi Connections for I2c

The LCD will be connected to a breadboard, and the following connections will be made from the board to the GPIO pins on the Pi. With GPIO on the left and the corresponding LCD connections on the right. The SS and SDO pins on the LCD will be left open:

* Pin01(3.3V) --> VCC
* Pin03(BCM 2 Data) --> DATA
* Pin05(BCM 3 Clock) --> CLK
* Pin06(Ground) --> GND

As a result the builder will now have produced something similar to the construction depicted below.

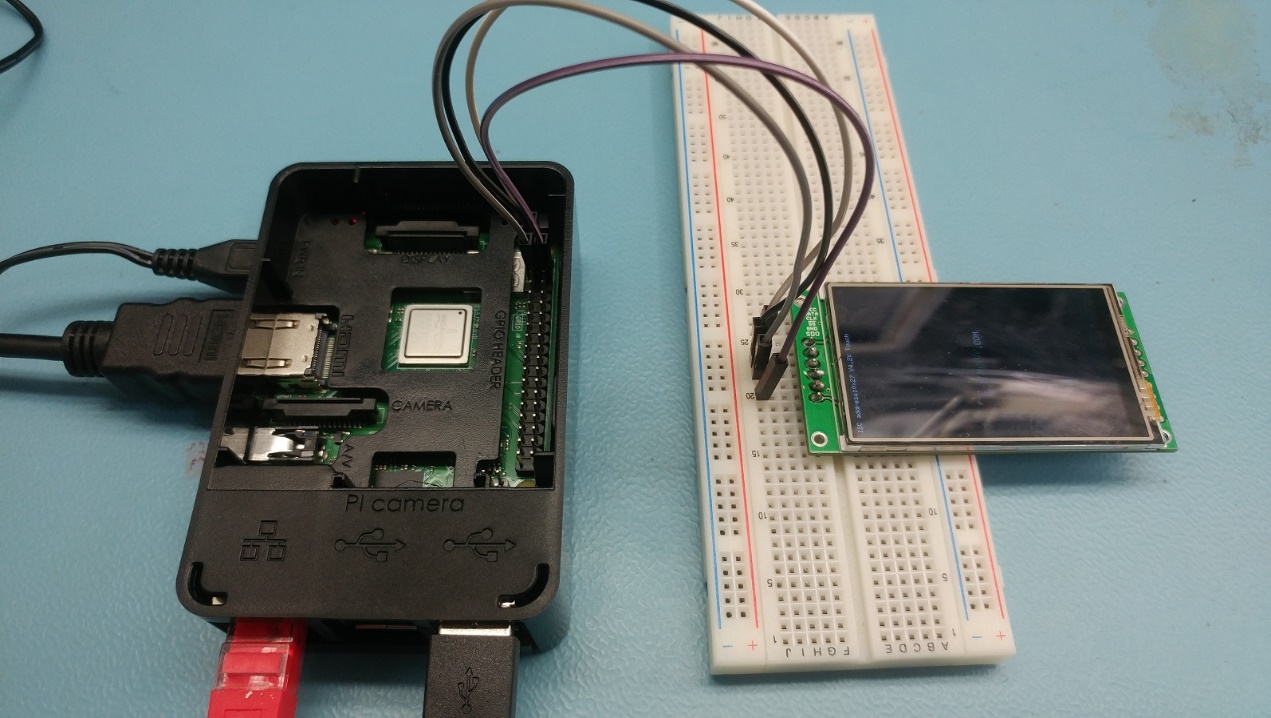


Figure 7: Digole LCD to Raspberry Pi Breadboard Connection

Note: A 6 pin header has been soldered to the LCD connectors in order to allow it to attach properly to the breadboard and later the PCB.

With the Pi powered on, and I2C communications enabled, the LCD will power on and display its factory default message indicating that proper power has been supplied, ground has been connected, and the LCD is in working order.

Next, to ensure the LCD has been connected properly for I2C communications, the following command should be entered in to the Pi's terminal: sudo i2cdetect -y 1. This will display a simple graphic listing each device connected to the I2C bus and its corresponding address. The address the LCD uses is 0x27.

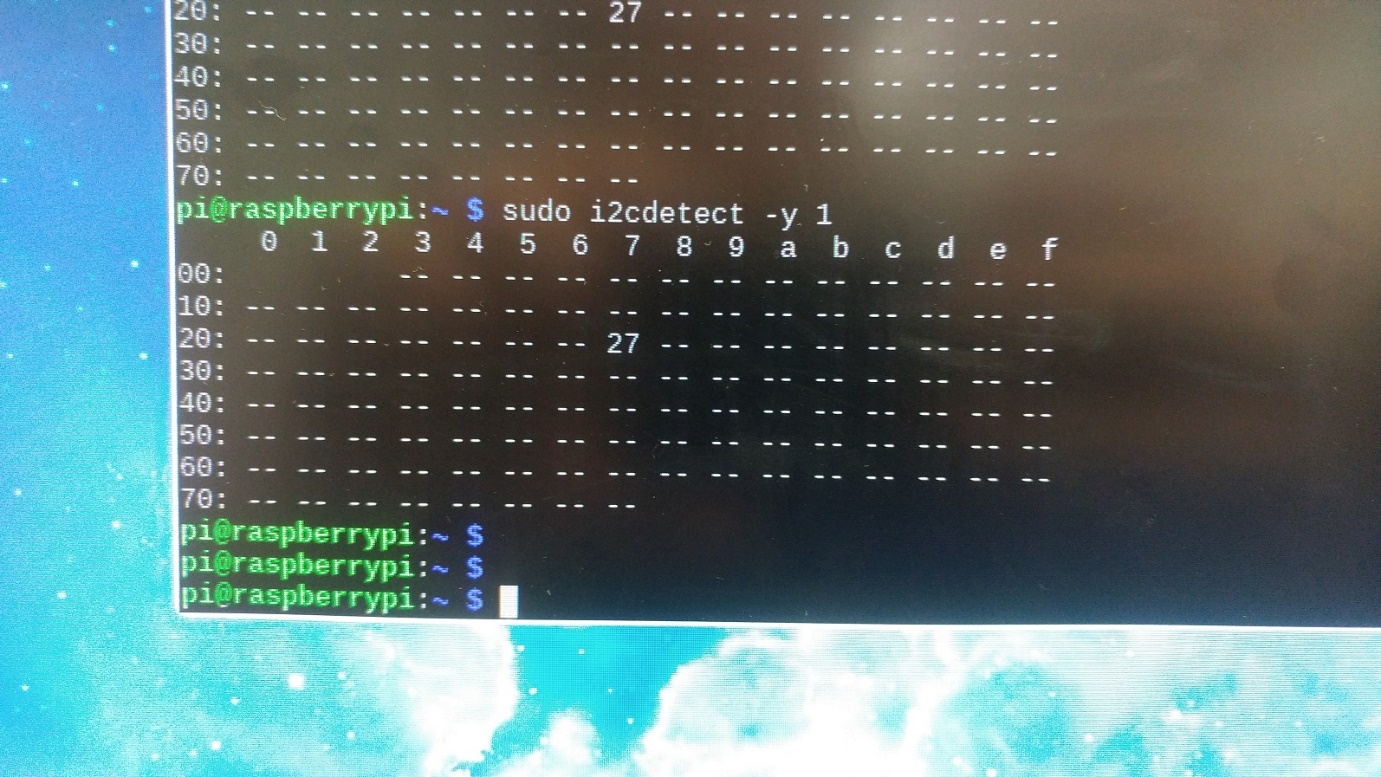


Figure 4: Digole LCD I2C Connectivity Verification

### NFC Reader

To enable communication between the Pi and the PN532 Board, you will have to choose a communications protocol and solder jumpers on the designated protocol. For this tutorial, we will be using I2C. The jumpers are included in the PN532 kit.

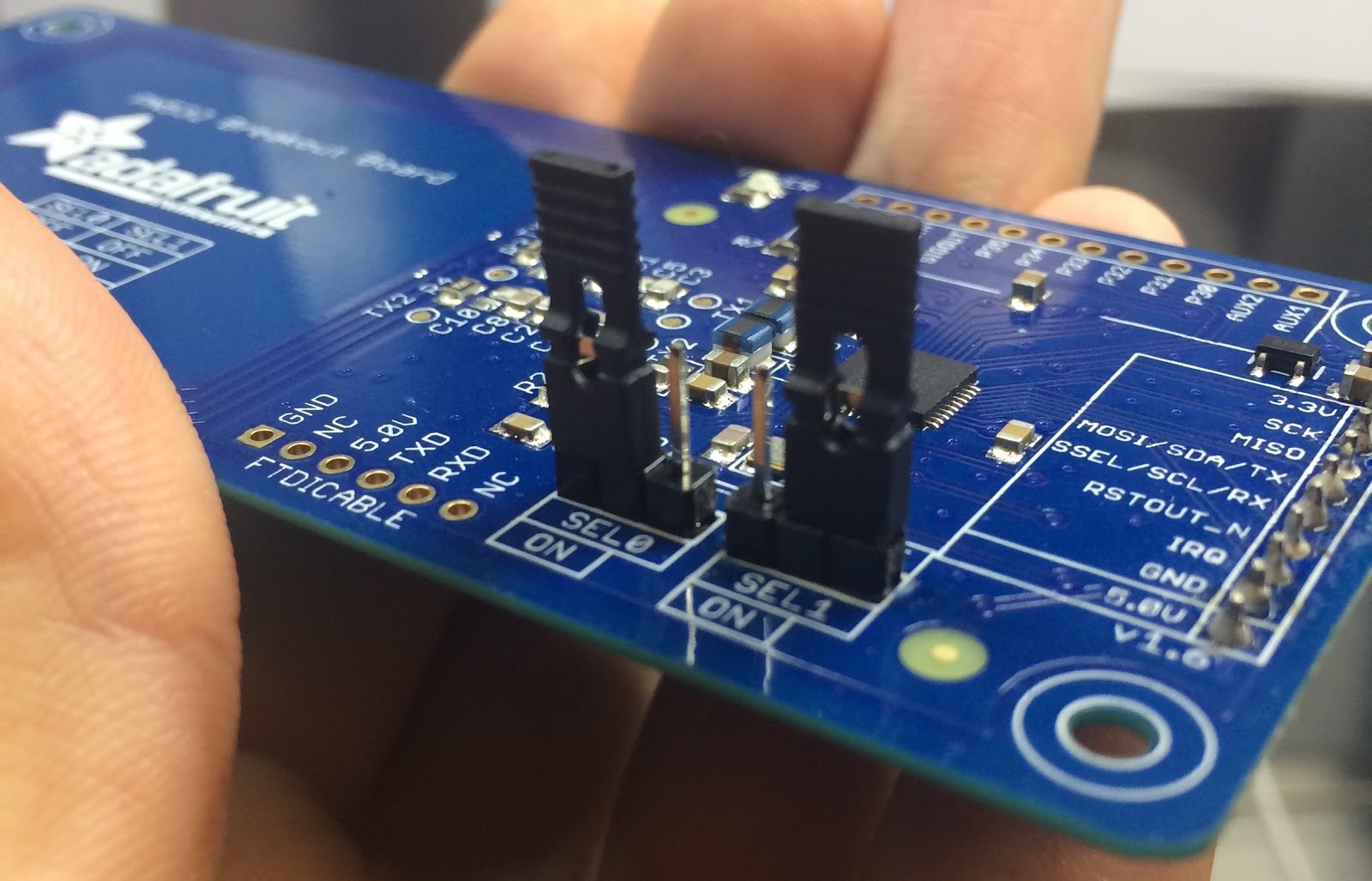


Figure 9: PN532 I2C Jumper Configuration

To enable I2C or any other communication protocol, the jumpers must be configured correctly. Setup table is below.

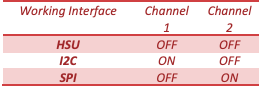


Figure 10: PN532 Jumper Connection Configurations

We will give it a quick test to make sure all our components are working. A prototyping breadboard is a great way to do this. The pins must be configured the right way for our sensor. Be extra careful as you do this to not damage you board.

* Pin01 --> 3.3V
* Pin03 --> SDA
* Pin05 --> SLC
* Pin06 --> GND

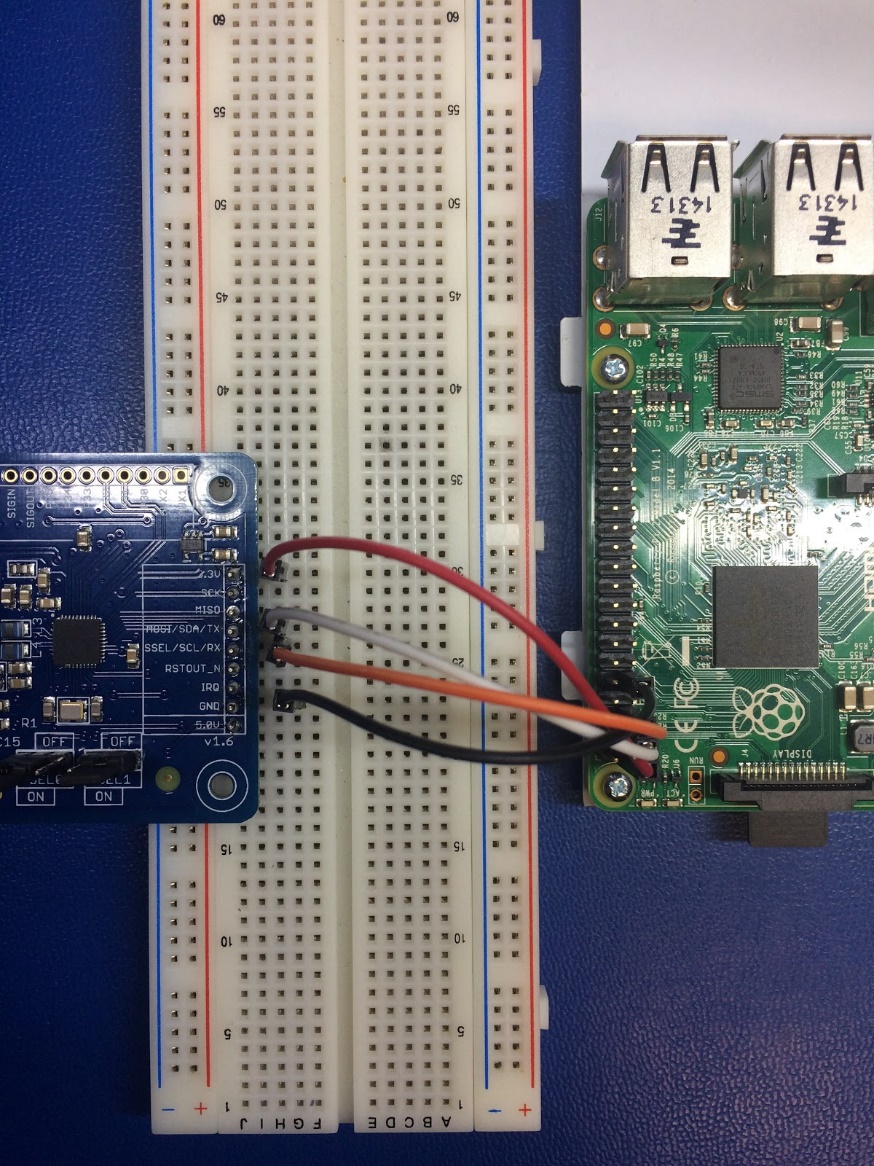
****

Figure 11: PN532 to Raspberry Pi Breadboard Connection

Running the i2cdetect program will let you know if your PI can actually see the i2c device. Your address might vary from mine and that’s fine. The goal is just get an address.

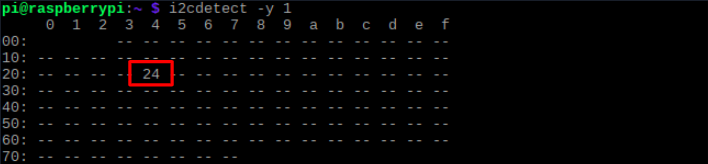
****

Figure 12: PN532 I2C Connection Verification

Using libnfc, we can run a simple poll program that when a card is detected, it will state simple information and confirm we can read NFC cards.

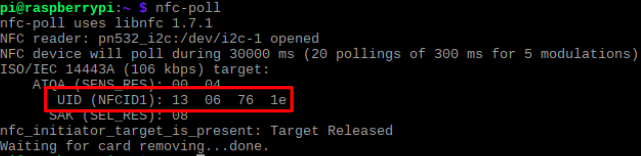
****

Figure 13: PN532 Hardware Test with nfc-poll

### Mini Push-Pull Solenoid

{To be added at a later date. Denald was absent.}

## PCB and Soldering

### Digole 2.4” LCD

The following PCB was designed and used for this project. The Gerber files for the PCB can be found from within this the project’s repository. Included in these files is a save for the application Fritzing which was used to produce this design. This file can be opened and used by the same program to add any modifications to the board as the builder should see fit. Such as, their own name and a description.

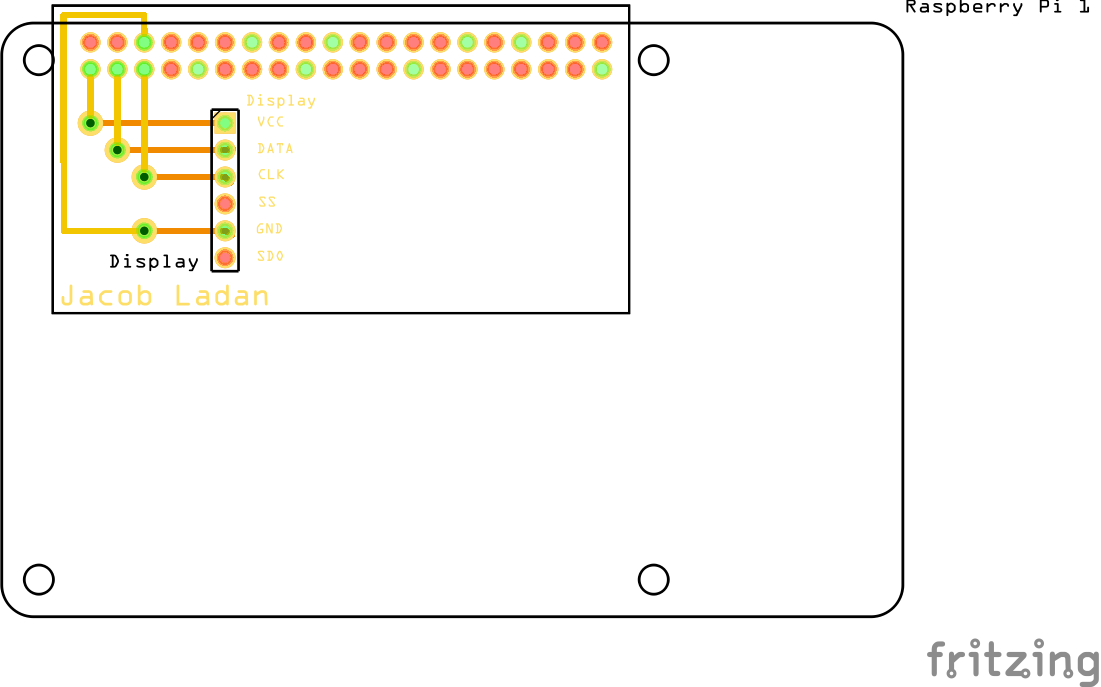
[](https://raw.githubusercontent.com/jacobladan/Digole-LCD-Display/master/documentation/Fritzing%20PCB%20V2.png)

Figure 14: Digole PCB Fritzing Schematic

To construct the PCB, the prototype lab located at Humber College was used. However, any third party production facility may be used, as the files are universally accepted as an industry standard. The two images below show the PCB constructed from both the top and bottom. A 6 pin header has been soldered to the top of the PCB to connect to the LCD and a 40 pin header has been soldered to the bottom to connect to the GPIO pins on the Pi. Additionally, the vias (holes connecting top and bottom) have been connected by soldering a small piece of conductor through them.

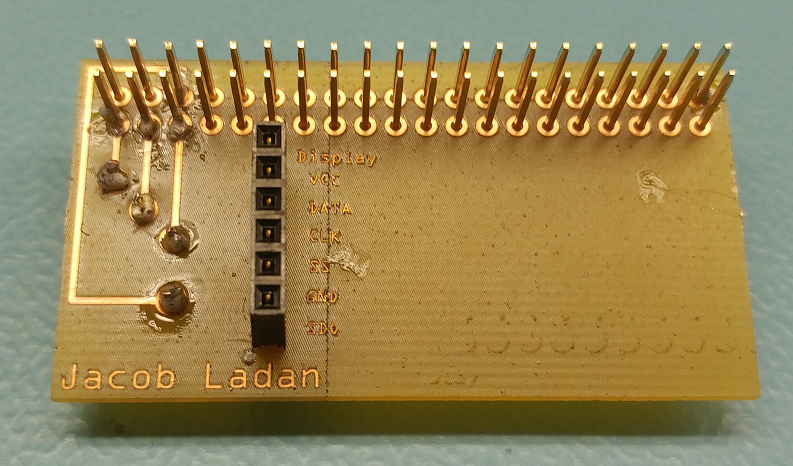
[](https://raw.githubusercontent.com/jacobladan/Digole-LCD-Display/master/documentation/PCB%20Front%20-%20Finished.png)

Figure 15: Digole PCB Soldered

### NFC Reader

{To be added at a later date.}

### Mini Push-Pull Solenoid

{To be added at a later date. Denald was absent.}

*A compiled version of the three hardware components and their respective PCB and soldering will be added at a later date when they have been constructed.*

## Software

### Digole 2.4” LCD

The C program in its entirety can be downloaded from within the project’s repository. Below is the source code for the program:

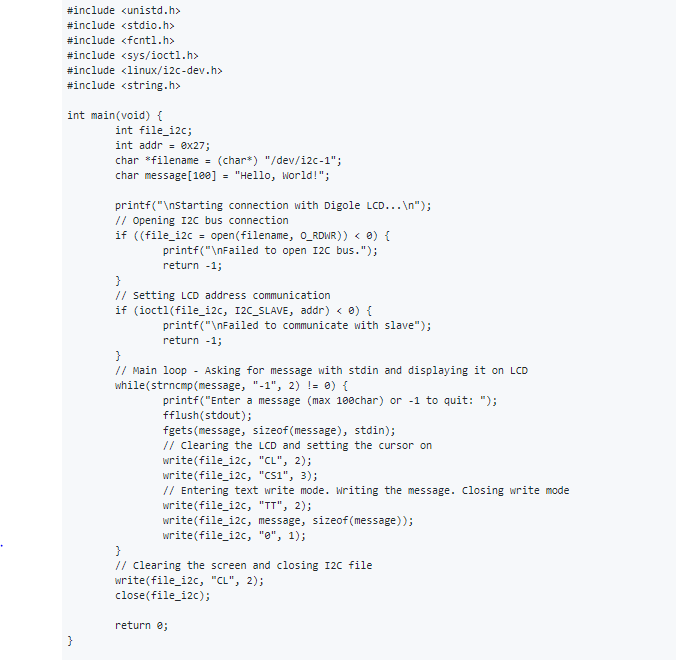


Figure 16: Digole Source Code

#### Program Explaination

1. The program connects to the I2C interface by opeining a file descriptor with read and write permissions that defines the driver for I2C communications on the Pi
2. The address for communicating with the LCD is defined by using ioctl() passing the file descriptor for the driver, the definition stating that the device is an I2C slave, and the address for the device being 0x27 as perameters
3. The program is entered in to a loop prompting the user for a message or if they would like to exit by entering -1
4. message is assigned the value of the string the user has input to the terminal, and will be used when writing to the LCD
5. write() is used to issue commands to the LCD by writing text to the I2C driver that was opened previously. "CL" is the command used to clear all the pixels and "CS1" is used to set the cursor position to the top left corner of the LCD
6. Text write mode must be entered for the LCD so that it knows to interpret the next command as to be what will be written to the screen. "TT" is used to do so
7. The message is then written to the LCD
8. Text write mode is left by issuing "0"
9. Finally, when the user is finished with the program and enteres -1, the program clears the screen with "CL" and then closes the I2C driver file
10. Program exits

#### Compiling

To compile the program using the gcc C compiler issue gcc digoleWrite.c -o digoleWrite in the terminal, making sure to be operating in the same directory as digoleWrite.c. This command compiles the program and creates an executable to be used to run the program.

### NFC Reader

Adafruit provides a guide to enabling communications through the PN532 board. The link below will be your starting point for development. The crucial section to pay close attention to is all the libraries and Pi configuration necessary to get up and running.

<https://github.com/adafruit/Adafruit_CircuitPython_PN532>

### Mini Push-Pull Solenoid

{To be added at a later date. Denald was absent.}

*As with the PCB and Soldering, a compiled version of all three parts of the program will be implemented at a later date when they have been completed.*

## Enclosure

We are currently in the process of building the enclosure in SolidWorks. We were not able to finish the enclosure for the week 12 milestone, since the lab computers do not come with SolidWorks installed. However, the enclosure is partially done and we are currently in the process of finishing the design. We will be ready to demonstrate the fully working project with the actual enclosure in the week 13 milestone. Below are the screenshots of our current progress with our enclosure:

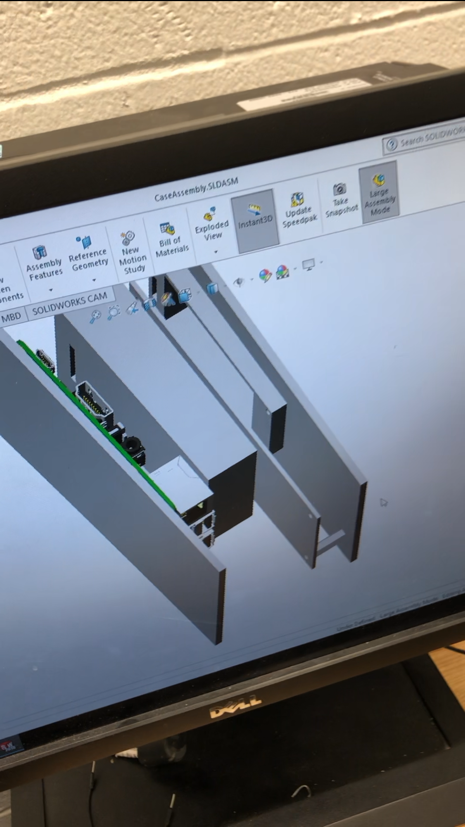
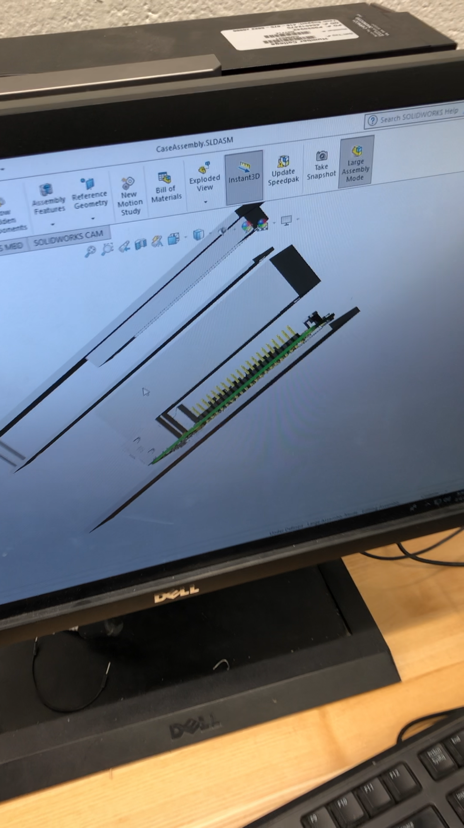


Figure 17: Roomi Enclosure

## Application

The application can be found from within the project’s repository. It is an Android mobile application built with the Android Software Development Kit with Java and XML. It requires android API level 28 or later. Additionally, a database must be configured in order to communicate with the Raspberry Pi.

## Database

The database is hosted on Google’s Firebase. Refer to the software requirement specifications of this technical report for a detailed outline of the structure and implementation of the database.

The construction of both the application and database is beyond the scope of this report.

# 

# **Conclusion**

This project combined the sum of our skills and knowledge gained through the first five semesters of the Computer Engineering Technology program at Humber College. As a result, the final product encompasses a door latching mechanism with NFC/RFID reader, graphical display, cloud storage database, and an accompanying Android application. The goal was to give users a modern approach to security systems for residential and commercial dwellings. This project demonstrated our extensive knowledge of both hardware and software. This technical report outlined the complete process of research, construction, testing, and implementation of each aspect of the project.

# 

# **Bibliography**

# 

# **Appendix**

* NFC - Near Field Communications
* RFID - Radio Frequency Identification
* LCD - Liquid Crystal Display
* PCB - Printed Circuit Board
* I2C - I squared C (serial 2 wire communication protocol)
* UART - Universal Asynchronous Receiver/Transmitter
* SPI - Serial Peripheral Interface
* JSON - JavaScript Object Notation
* SQL - Structured Query Language (database language)
* NoSQL - Not Only SQL
* REST - Representational State Transfer
* OS - Operating System
* GNU - GNU not Unix (open source UNIX based OS)
* CPU - Central Processing Unit