NFC Inventory Management System

Tuesday, April 14, 2020

Robert Dinh, Jonathan Luong, Colin LeDonne.

Declaration of Joint Authorship

We, Jonathan Luong, Robert Dinh, and Collin LeDonne, confirm that this work submitted is the joint work of our group and is expressed our own words. Any uses made within it of the works of any other author, in any form (ideas, equations, figures, texts, tables, programs), are properly acknowledged at the point of use. A list of the references used is included. The work breakdown is as follows: Each of us provided functioning, documented hardware for a sensor or effector. Student A provided the Adafruit HD44780 LCD. Student B provided the Adafruit PN532 NFC/RFID Controller. Student C provided the Adafruit VCNL 4010 Proximity Sensor. In the integration effort Collin LeDonne is the lead for further development of our mobile application, Robert Dinh is the lead for the Hardware, and Jonathan Luong is the lead for connecting the two via the Database.

Proposal

We have created a mobile application, worked with databases, completed a software engineering course, and prototyped a small embedded system with a custom PCB as well as an enclosure (3D printed/laser cut). Our Internet of Things (IoT) capstone project uses a distributed computing model of a smartphone application, a database accessible via the internet, an enterprise wireless (capable of storing certificates) connected embedded system prototype with a custom PCB as well as an enclosure (3D printed/laser cut), and are documented via this technical report targeting OACETT certification guidelines.

Intended project key component descriptions and part numbers:

Development platform: Raspberry Pi 3 B+

Sensor/Effector 1: Adafruit HD44780 LCD

Sensor/Effector 2: Adafruit PN532 NFC/RFID Controller

Sensor/Effector 3: Adafruit VCNL 4010 Proximity Sensor

We will continue to develop skills to configure operating systems, networks, and embedded systems using these key components to create a functional device that will trigger a proximity sensor to enable a RFID controller in read mode and turn on an LCD display. This device will be used to read an NFC NDEF message from a phone application to identify the student and their requested parts. The Android application that we will develop will allow students to select the parts they need and the application will also generate the NDEF message to be scanned over NFC using dedicated protocols. Also, a web browser application will also be developed mainly to monitor requests and

inventory. It will also act as an alternative method for requesting parts. These three components will create an inventory management system that will benefit both the students and the staff. It will seamlessly allow students to request items and pick their parts up without the hassle of purchasing bags and RFID tags for individual students.

Our project description/specifications will be reviewed by, Professor Kristian Medri and Vlad Porcila (Both within the Faculty of Applied Sciences and Technology at Humber College), ideally an employer in a position to potentially hire once we graduate. They will also ideally attend the ICT Capstone Expo to see the outcome and be eligible to apply for NSERC funded extension projects. This typically means that they are from a Canadian company that has been revenue generating for a minimum of two years and have a minimum of two full time employees.

The small physical prototypes that we build are to be small and safe enough to be brought to class every week as well as be worked on at home. In alignment with the space below the tray in the Humber North Campus Electronics Parts kit the overall project maximum dimensions are 12 13/16" x 6" x 2 7/8" = 32.5cm x 15.25cm x 7.25cm.

Keeping safety and Z462 in mind, the highest AC voltage that will be used is 16Vrms from a wall adapter from which +/- 15V or as high as 45 VDC can be obtained. Maximum power consumption will not exceed 20 Watts. We are working with prototypes and that prototypes are not to be left powered unattended despite the connectivity that we develop.

Executive Summary

The purpose of this document is to lay out the requirements and specifications for the hardware and the client-side application for the NFC Inventory Management System. This system will work in conjunction with an online management interface to maintain inventory in an efficient and organized manner. This product will mainly be developed as a Parts Crib solution at Humber College North Campus but it can also be utilized by other companies for their own purposes. This product will allow the client's customers(students) to request items through a mobile or online interface and sign the items out with their phones as a form of identification through NFC. This removes the hassle of providing a card as a form of ID and eliminates the use of RFID tags that have to be individually programmed for each person. All the data that needs to be organized and managed to make the system work flawlessly, will be handled by the database with an online interface, which allow clients to audit their inventory and manage their customer's information.

Contents

List of Figures

1.0 Introduction

Today, modern-day memberships require plastic cards as a form of identification. Whether it be a bank card, gym membership, library card or even a points card, they often require people to tap or swipe their cards. It can be viewed as troublesome for some people as they often keep the cards in their wallets which can build up over time.

A solution would be to develop an android application that can be read over an NFC interface to a physical reader and be managed by a web interface with a database. Even though this type of solution is already being utilized today for transit and payments in proprietary formats, it has not yet been popularized in the areas mentioned above. To demonstrate the concept, the solution was based around the Humber Parts Crib, a campus facility that lends tools and equipment for students to use during their labs and projects. The current system has the user download buggy software that is not very intuitive and rather difficult to use without instruction. One must download the software from a dropbox and then the software may need to be loaded several times to work properly.

The goal was to create a solution to this problem by creating a smartphone application that uses HTTP protocol to send and receive data from a server along with hardware to make the process of borrowing parts easier. The server currently that is used is the Humber Apollo Server (apollo.humber.ca) under a student's profile for testing, but can be moved elsewhere for deployment elsewhere. The mySQL database on the server has 3 dedicated tables for accounts, inventory, and transactions. Each table has one field or more that will be able to relate to another table.

The Android application is designed to interact with the database. It has a graphical user interface to allow students to create a user, sign in, request items and view previous requests. The process has the user queue the parts they require from the mobile application, which operates like an e-commerce store, which will then notify the employee at the parts crib via web application. When the user queues up parts, this will create a unique NFC code for the user to scan at the parts crib. The employee will then prepare the order for the user to pick-up. When the user arrives at the parts crib, they will scan their phone on the NFC reader which will then update the status of the unique transaction on the database that the employee can see. This will then update the order status on the database until the user scans their phone again on the NFC reader when items have been returned.

The web application will give the employee at the parts crib the ability to release orders, see the contents of the orders, and any relevant information requiring the order. The hardware includes a VCNL 4010 proximity sensor, Adafruit PN532 NFC Sensor, and a 16x2 HD44780 LCD Display. The NFC reader is responsible for receiving the information from the user's phone, the LCD display is responsible for displaying the information from the NFC reader to the user, and the proximity sensor will turn up the brightness of the LCD screen when the user puts their phone near the NFC reader. We will be designing a PCB with the intention that the sensors will be placed next to each other to reduce the size of the case. All the components will fit into a case that will be laser cut and the layout of the components will be important so it will be easier to use and understand.

1.1 Product Scope

The purpose of this project was to create an inventory management system that permits equipment provisioning for the Parts Crib that provided a friendly user experience for students and employees. The process involves interacting with a database through requests and finalizing transactions through an interaction between a user's mobile device and the NFC reader located at the crib. The system implements functionality from the current system as well as provide features for both parties to stay informed regarding items pertaining to them to ensure users are updated accordingly per use.

The current database contains user account information and inventory that range from various tools and equipment. The goal of the project was to create an easy experience for the users and intuitive enough for new users so it is simple and convenient for students to check out equipment.

The IoT capstone project implements a distributed computing model comprised of a smartphone application, and a database accessible via PHP scripts using a web browser or our NFC reader embedded system prototype with a custom PCB along with enclosure.

The project was limited to the database that was created by the team and was only tested through simulation of the process. Also, because the application will contain personal information users there are security aspects such as secure practices in the handling of our account database involving unique IDs and hashed passwords (Hasing Security, 2019) that need to be met.

1.2 Requirements and Specifications

Application

- developed for android mobile devices starting at API 21
- requires internet connection over WiFi
- mobile device must be NFC-equipped

Hardware

- Requires wall outlet for power
- Wi-Fi to communicate with database
- Raspberry Pi 3 B+ with WiFi and I2C enabled

Database

- MySQL Database to read/write/store data
- Requires use of PHP scripts and/or web browser to communicate to the database stored on student account in Humber College servers

2.0 Background

Radio frequency identification, also known as RFID, is the utilization of tags with integrated chips and coiled wire that can be activated using radio signals to retrieve their unique identification number or other information. Tags that are often passive, do not require a power source as its main source is from a reader that sends out an electromagnetic signal (Butterfield & Szymanski, 2018). These RFID tags are commonly found in everyday activities such as tapping a membership card to get into a car wash, to make a payment for food or to get a bus/train(transit).

Near Field Communication, also known as NFC, is a subset of RFID. NFC operates at a frequency of 13.56MHz and transfers data at a rate of 106k-424k bits per second (Butterfield & Szymanski, 2018). NFC capable devices can be switched to emulate passive or active states, such as smartphones to read and write to each other. To standardize the flow of information from device to device, NXP Semiconductors, Sony, Nokia and Philips have joined together to create protocols that are found under the NFC Forum name (Vanderkay, 2004)Currently, NFC Forum has 5 different specifications for tags, each unique for their own purposes. For this project, the Type 4 Forum Tag (ISO/IEC 14443) specification will be used to communicate between the hardware and an Android mobile device which. This involves 4 parts: physical characteristics, radio frequency power and signal interface, and initialization and anti-collision (Sabella, 2016). The Type 4 tag was selected due to the ability to hold large amounts of data in NFC Data Exchange Format (NDEF). This will be furthered explained in conjunction with Application Protocol Data Unit(APDU) commands specified in ISO/IEC 7816-4

(Organization, security and commands for interchange) to show how one device can access data on another device .

Modern day smartphones, capable of doing many tasks, have been very convenient for everyday users. With the ability to perform host card emulation on these Android devices, contactless payment and other data transactions are made possible through NFC from one device to another. This type of technology can be applied to modern day inventory management problems such as the Parts Crib. The Parts Crib currently uses RFID tags that are individually scanned in and out for each incoming student, costing precious time and money. Having a contactless transaction of information with a mobile device will allow the Parts Crib to operate efficiently and effectively without requiring any extra manpower. Since students carry around their phones every day, it is simpler to just tap their phones than to rummage through purses and wallets. This system will allow students to securely transfer their information to the Parts Crib for easy access.

3.0 Methodology

3.1 Required Resources

3.1.1 Parts, Components, Materials

The NFC System Management System revolves around a hardware device that senses an incoming device, scans NDEF messages containing data via NFC and displays the status of the device. The internal hardware component has 4 main devices with supporting electronics parts to deliver the functionality.

The first main component is a Raspberry Pi 3 B that revolves around a system on a chip and has GPIO pins for power and data transfer. The device's main purpose is to gather data from the sensors and display data to the effectors. To manage and handle the data, it runs scripts to interpret the data and sends it off to a database server for further processing.

The second device is the Adafruit PN532 NFC Controller that is capable of many functions but mainly acts as an NFC reader to get access to the NDEF messages sent by an Android mobile device. For the device to power up, it requires a connection to the 3.3V and ground lines from the Raspberry Pi. To communicate with the Raspberry Pi, it needs to be wired in the I2C configuration which requires connections to the Serial Data Line (SDA) and a Serial Clock Line (SCL) line.

The third device is the Adafruit HD44780 16x2 LCD Display that is used to output messages to the user based on their interaction with our system. After completing a transaction through the device above the output will either display a success or instructions/error code based on the action. The device requires a 5V and ground

connection to the Raspberry Pi. To communicate to the Pi using I2C a PCF8574P IC I/O expander chip is used to communicate to the device, this requires 3.3V and ground from the Pi as well as connections to the SDA and SCL lines. To control the contrast and backlight of the LCD there are two 10k ohm potentiometers wired to the LCD as well as a 10k and 4k7 ohm resistor for better control over the lighting options. This setup also provides protection to the LCD so less voltage is being input to the device.

The fourth device is the Adafruit VCNL 4010 proximity sensor which will be used to for user input to the device. The proximity sensor will be set up in such a way so that when the user puts their phone near the device, it will brighten the LCD screen. The VCNL 4010 has an SCL and SDA pin that are used to transfer data. The values the 4010 returns are unit less but need to be converted. The closer you get to the sensor, the higher the proximity value and the lower the ambiance. The sensor's IR led operates from 3.3 to 5 volts and is not very large as it is smaller than a quarter.

3.1.2 PCB and Enclosure Design

When the hardware components have been wired and tested on a breadboard, a printed circuit board (PCB) was designed in Fritzing with the Adafruit library. It was designed so that it can be a connecting point for all the main devices and its supporting components. It was also designed for modularity with troubleshooting in mind so that nonfunctional components can be easily accessed and replaced.

The PCB design contains holes for female header pins to be soldered on. One 2x20 40-pin female header was used to connect the PCB to the raspberry pi's GPIO pins. Also, a 4-pin female header, 9-pin female header, and a 2 5-pin header will be used for a

proximity sensor, an NFC reader and an LCD display respectively. Each header that is dedicated to each device has lines providing power and I2C communication running from the 40-pin header. To route these lines, vias were created so that the lines would overlap, allowing them to be placed both on top and bottom of the PCB. Any other supporting electrical components such as resistors and transistors has holes dedicated to them as well. For the PCB to be completely secure and held in place, 4 mounting holes around 2.5-3mm in diameter was put in so that screws can be threaded through to hold the PCB to the nylon standoffs that will be put under it. When the PCB was finished in its designed phase, the files were sent off to a local prototype lab at Humber College to be printed.

When the PCB was being sent off to be printed and the design files were then imported into AutoCAD and Inventor to design the case to surround the internal hardware. The external design was boxy in shape similar to that of a rectangular prism.

The base of the enclosure was based on the dimensions of the Raspberry Pi. Holes with standoffs were created so that the Raspberry Pi did rest on the soldered joints but rather on elevated pads. This prevented the device from overheating and possibly melting the plastic underneath it.

The sidewalls of the device have holes created to give access to the Raspberry Pi’s micro USB for power, HDMI, USB ports, Ethernet port, and an audio port. Having access to these ports allows easy access for future development and mainly for troubleshooting the device. Extra holes were put in place to allow the Raspberry Pi’s LED and the NFC reader’s LED to check on the status of other devices.

The removable top-lid of the device was designed to have holes for the LCD Screen and a very small hole for the proximity sensor. 2.5-3mm mounting holes were put in place for each device so that they do not shift around during transportation or during a transaction.

3.1.3 Tools and Facilities

The initial setup of the final hardware prototype required the integration of all devices wired on a breadboard and so that all components could be inspected when they are powered on. This was mostly done in the J232 Lab room at Humber College. Using the digital multimeter in the room was important for ensuring the right voltage levels were present in specific parts of the build. Schematics, breadboard designs, and PCB designs were created on Fritzing before the development of the PCB.

After completing the tests required for the devices to all work on the same platform together, the custom PCB (Printed Circuit Board) and enclosure were designed and created through the facilities at Humber College.

All of the past custom PCBs were created after submitting the designs to the Prototype Lab at Humber College. The past enclosures were split between submitting a design for laser-cutting and 3D printing at the same location. Through comparing the pros and cons of both methods, the plan was to use the same methods to have the final product's PCB and enclosure created at the Prototype Lab with the enclosure being a mix of 3D-printing and laser cutting. Preference was for a full laser-cut enclosure however 3D printed portions of the enclosure requiring mounted screws were with a better design.

After the PCB was created the assembly took place in the J232 Lab room. This facility provided soldering stations with fume extractors to provide user safety. This room was

an important factor towards the assembly step of the PCB as it was an easy access point to solder the PCB in safe work conditions. Along with the facility, the use of pliers and wire cutters were required to prepare fine wire, to shape components such as connectors, and to trim the smaller pieces that were soldered onto the PCB (example: resistors). Safety goggles were essential for any worker near the solder work to prevent any exposure of chemicals to the eye, and the fume extractor lowered the risk of inhaling of any chemicals.

Through past solder work, the digital multimeter was found to be important as it helps to ensure connections were made on the PCB and also to be able to test voltage levels for the devices when it was assembled. After this phase of testing was completed, the PCB was attached to the Raspberry Pi to ensure the devices worked and could be read and written to. The testing here was similar to the breadboard testing, however, it gave real size dimensions as a whole and helped determine the finalization for sizes and designs towards the enclosure. Lastly, the enclosure assembly required mainly screws, heat-set inserts and the use of a screwdriver to bring the pieces of the enclosure together with PCB and the RaspberryPi.

3.1.4 Shipping, duty, taxes

The main components, PN532 NFC, Raspberry Pi, HDF8574P LCD screen, and the VCNL 4010 proximity sensor, were all obtained mainly from amazon or directly from the manufacturer. The majority of these places do apply taxes but do not charge for shipping or duty. The individual costs of the parts are the following: Raspberry pi was $45.75, the HD44780 LCD screen was $9.95, the VCNL 4010 proximity sensor was $7.50 and the PN532 NFC reader was $71.42. All of these components combined came

out to be $134.62 CAD. With taxes, this brings the total to $152.12 CAD which is $17.50 in taxes. The other smaller components include: the nylon stand-off kit, which was $13.99, the PCF8574P Remote 8-bit I/O expander IC which was $9.95, a 10K resistor which was $0.15, and a 4k resistor which was also $0.15. With these components added it then brings the grand total to $161.86 and $179.51after tax. This means that the taxes paid be $20.65 for all the materials. The shipping stated that the parts would arrive in 5-7 business days however, in the past, orders were delayed and adjustments had to be made to the project plan in order to ensure that the project would be done on time. For example, when Colin ordered his blue pill, it got delayed by a week and he had to make adjustments to his schedule to ensure he finished it on time. The team was prepared to make changes to the schedule in case unforeseen events like delayed shipments take place.

3.1.5 Time expenditure

Working time for the project was mostly done outside of the allocated lab time. The exception to this was for soldering the PCB, printing and assembling the case. The plan was to assemble most of the hardware outside of the lab as the workload was divided and set dates needed to be followed. This made it easier and more efficient as more work could be done in a shorter period of time since all of the group members will be contributing to different parts of the project at the same time. The plan was to use most of the lab time as a meet up to troubleshoot any issues and to make plans for future steps. The majority of the time was used to polish the mobile application, develop the web application, and create the hardware firmware. The lead time consists of brainstorming, troubleshooting, and discussing future plans and any possible changes

to the schedule or overall direction. Working time was used to test and debug any issues that might occur during the development of this project. Since most of the development was done outside of lab time, communication was done over online chat. Following this work ethic is the best possible way to approach this project in terms of working time and lead time.

3.2 Development Platform

The mobile application was developed using java with Android Studio version 3.5 with API 21. The app coexists and communicates with the webserver that handles requests through PHP to an SQL database.

3.2.1 Mobile Application

The app is currently working as intended and communicates with the server with no issues. Going further into the semester Colin will be working to polish the GUI to make it more appealing and easier to follow since that was one of the goals. Since the major functions of the app are now operational, polishing the GUI should not be too time-consuming since it just requires fixing the XML files. The mobile application is due March the 14th 2020 and there should be no issues getting it done by that date.

The mobile application was made in android studio using API level 21. The app was developed with the intention to make it easy and straight forward to use. The project was divided into three parts; the GUI, the database, and the back end. These parts were allocated to the three group members and work was completed on time. The app has a total of 11 screens and was designed to be easy to navigate. The main activity that will load after the splash screen is the login screen.

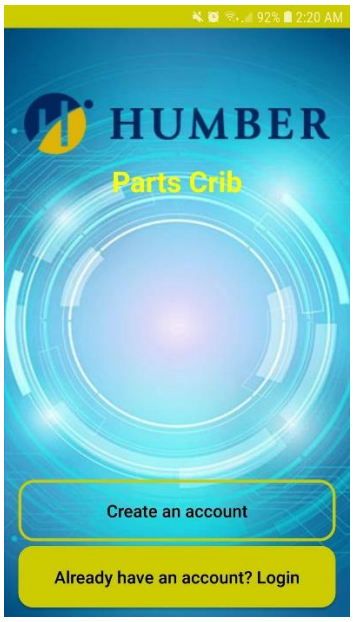

Figure 1. Main Activity

The main activity (Figure 1) was made using two custom buttons and a simple image placed at the top of the screen. The two buttons lead to the two corresponding activities.

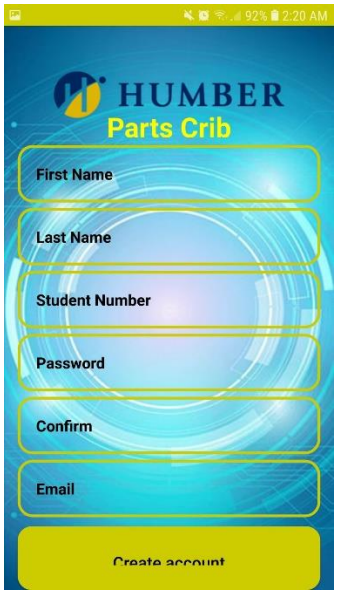

Figure 2. Create Account Screen

The create account (Figure 2) activity was made with custom input boxes and a custom button. This screen was made to mirror other create account pages or screen. In order

for an account to be created, all of the fields must have valid information in them. If the user tries to make an account while having no information entered, it will alert the user that they need to fill in the information. This was crucial when creating this screen since the database would be full of incomplete information.

Figure 3. Login Screen

The login screen (Figure 3) was made to also be simple and straight forward. This screen is also where the admin can log in once the web application is completed. This screen was made with two custom input boxes, a custom button, a remember me checkbox, that saves the user's login information on the phone's memory, and an admin login.

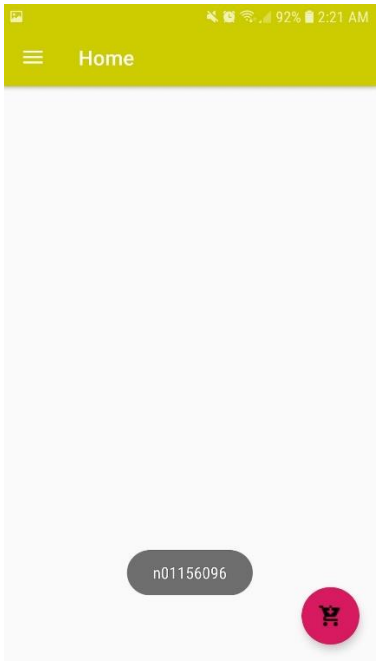

Figure 4. Home Screen

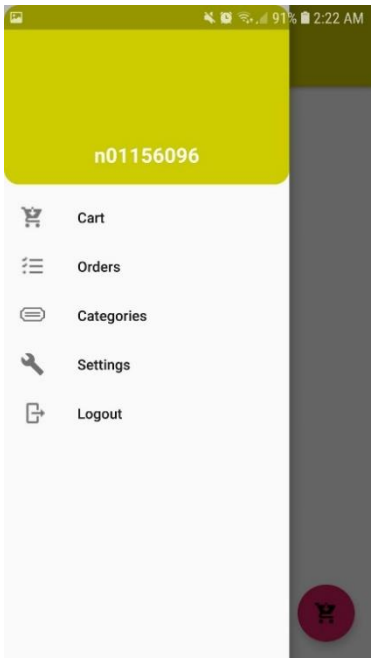

Figure 5. Navigation Drawer

The home activity (Figure 4) was made by using a navigation drawer (Figure 5) that lead to the other main activities. This screen will also have notifications from the parts

crib. The notifications feature will be implemented when the web application is completed.

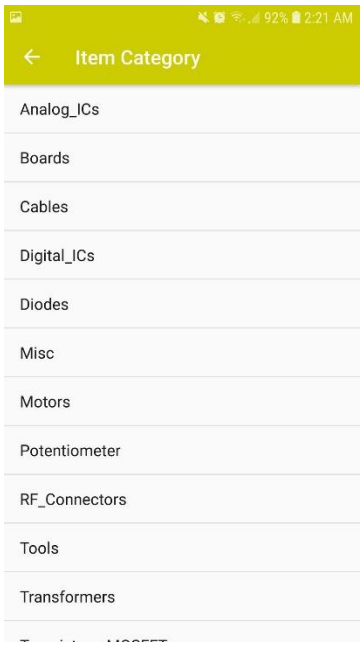

Figure 6. Items Category Screen

The items category (Figure 6) uses a list view that pulls data from the SQL database to display the current items available at the parts crib that the user can borrow from the parts crib. All of the screens from the navigation drawer also have a back button on the top left to make navigation easier. This was done by making separate processes in the manifest file.

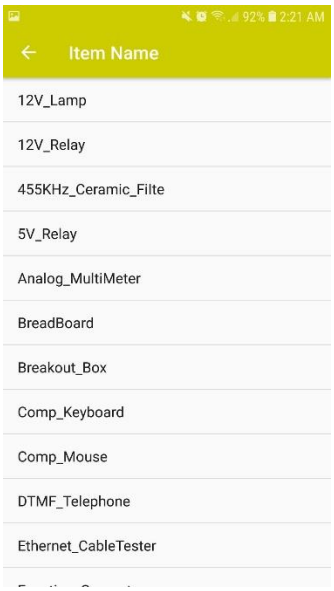

Figure 7. Items Name Screen

The items name screen (Figure 7) is similar to the item's category screen (Figure 6) as they both use a list view and pull data from the SQL database to display to the user.

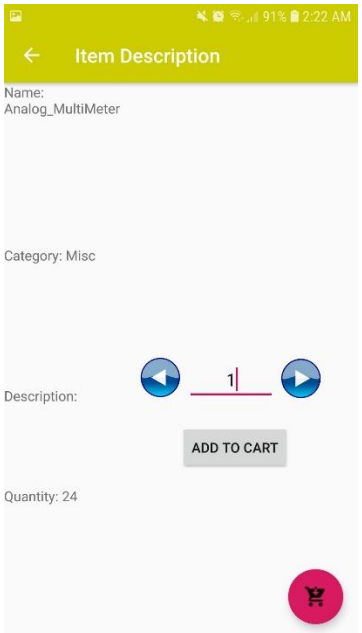

Figure 8. Item Description Screen

The item description screen (Figure 8) can is accessed from pressing on the desired item and it pulls the corresponding information from the SQL server to display to the user.

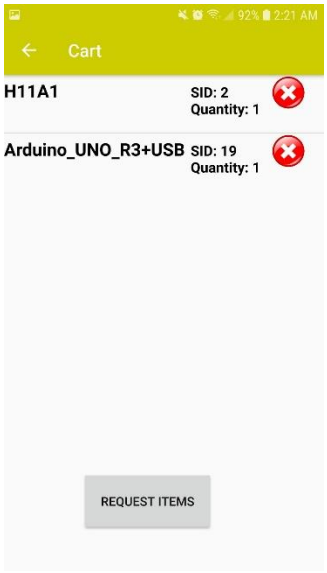

Figure 9. Cart Screen

The cart screen (Figure 9) is another list view that gives the user an overview of what they are currently planning to request. The cart screen can be accessed from the navigation drawer (Figure 5) or the floating action button on the home screen (Figure 4). Once the user presses the request button, the parts crib will receive a notification the web application to prepare the order.

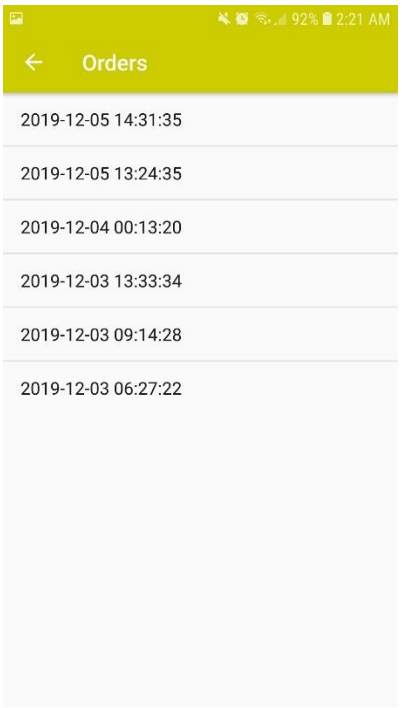

Figure 10. Orders Screen

The orders screen (Figure 10) displays all the previous orders that the user has made. This is just a list view that pulls order details from the SQL server. Pressing on the order will display what was in the order and if it has been returned or not.

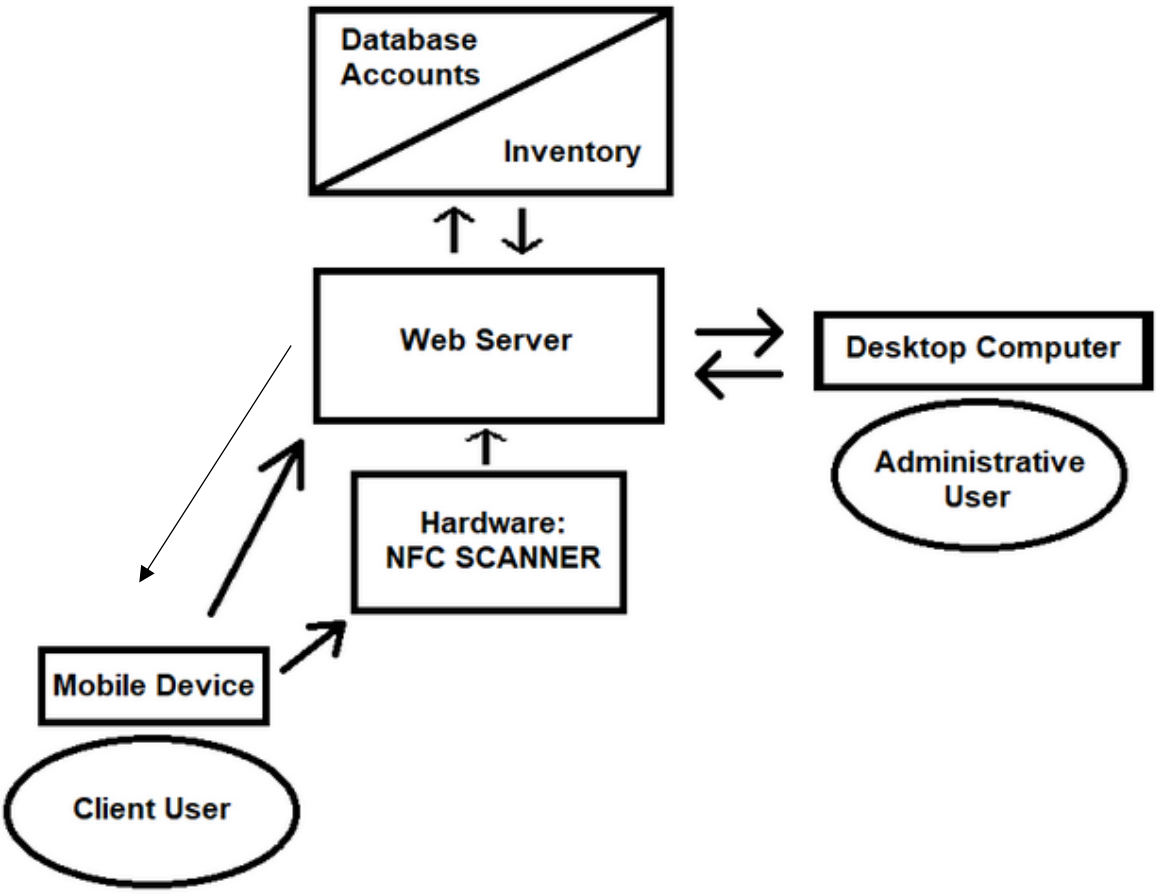

Figure 11. Data Visualization

The data in the app mainly flows from the client's mobile device to the webserver. The webserver also communicated with the database in order to pull information to display to the user. For example, when the user wants to view their older orders, the data is sent from the user and then the server pulls the information from the database to display to the user's mobile device. The user's mobile device is also going to communicate with the NFC reader at the parts crib. Once the user is ready to pick up or return their order, they tap a button in the app to get into an NFC state to transfer transaction ID to the NFC reader. This transaction ID will be used to look up the order in the database and update the status of the order as picked up/returned.

3.2.2 Image/firmware

The Raspberry Pi uses the current up-to-date image of Raspbian (Buster) operating system found at their main website. The image was flashed onto the 16GB SanDisk SD card that is used with the Pi. The process was done using an application called ‘balenaEtcher’, this application is used for flashing operating systems onto SD cards and USB drives.

The firmware used for the hardware prototype was compiled through three different methods used by each member to initially implement their sensor/effector from the hardware projects last semester. Each device was setup to communicate with the Pi via I2C.

The VCNL4010 Proximity Sensor was rewritten in C with the source code provided by Adafruit repository for the sensor. The main code used reads the proximity value given by the sensor every 3ms, when the value surpasses 2300 units the program will initiate the set of code for the NFC portion of the task.

The NFC reader PN532 utilizes example code provided by nfc-tools.org as well as their wiringPi libraries and is written in C. Upon activation the NFC sensor begins a process by first detecting the nearby device then selecting the application ID to receive an NDEF (NFC Data Exchange Format) message that contains the transaction ID of the client from their mobile device. If the device does not have the corresponding application ID the process will terminate. When a NDEF message is successfully the program will initiate the script for the LCD with the information given through a system call.

Using a guide provided by Rototron.info as well as the libraries provided by Adafruit the HD44780 LCD screen is able to display strings written to its respective address from the NFC function. The script used to display strings on the LCD was written in Python.

The goal was to achieve communication between the hardware prototype and the PHP script that handles the database. The plan involves communication over HTML using GET and POST to update the server and complete the transaction process at the Parts Crib when the client has tapped their mobile device. Documentation of the resources used to achieve this are provided at the repository for reference.

3.2.3 Connectivity

The Raspberry Pi when setup and in Wi-Fi range can be accessed remotely from another terminal using RealVNC Server/Client. Provided in the wpa_supplicant.conf file on the Pi are the credentials for access to the Humber College Wi-Fi under a student account. With this enabled any user logged into the RealVNC team on the client application can access the device remotely. Internet connection can also be established by connecting an Ethernet cable between the Pi and any networking device that has connection to the internet.

If internet connection cannot be established the Pi can also be communicated to by connecting the Pi to another device via Ethernet and using the network application Bonjour or any similar functioning tool. Bonjour is the application selected by the team and it is used to help setup a network between connected devices. This allows the connected device to assign an IP to the Pi so that it can be remotely connected to.

If neither of these options are a viable option for the scenario then peripherals can be connected directly to the Pi and interacted with.

3.2.4 Breadboard/Independent PCBs

The hardware seen in this system were all designed in the previous semester. The LCD, the proximity/luminosity sensor and the NFC Controller were all carefully tested using the I2C interface before the integration on two separate breadboards. Due to the large amounts of components and required space, the LCD needed a dedicated breadboard while the rest shared a single breadboard.

The LCD, in general, requires a lot of wire when it used in a parallel interface. Though it is almost impossible to run all those connections to a Raspberry Pi GPIO, the LCD uses a PCF8574 IC to use the I2C interface to convert the parallel interface into a serial one. To run in an I2C interface, the PCF8574 IC needs 3V, SDA, SCL and ground to be connected from the Raspberry Pi while the remaining connections are meant to be used to control the LCD. Using the potentiometers with the 4k7 and 10k resistors tied to them allows control over the backlight of the display and the light of the text (the output) during the testing phase of the device. The potentiometer will later be replaced with fixed value resistors to keep the device more compact.

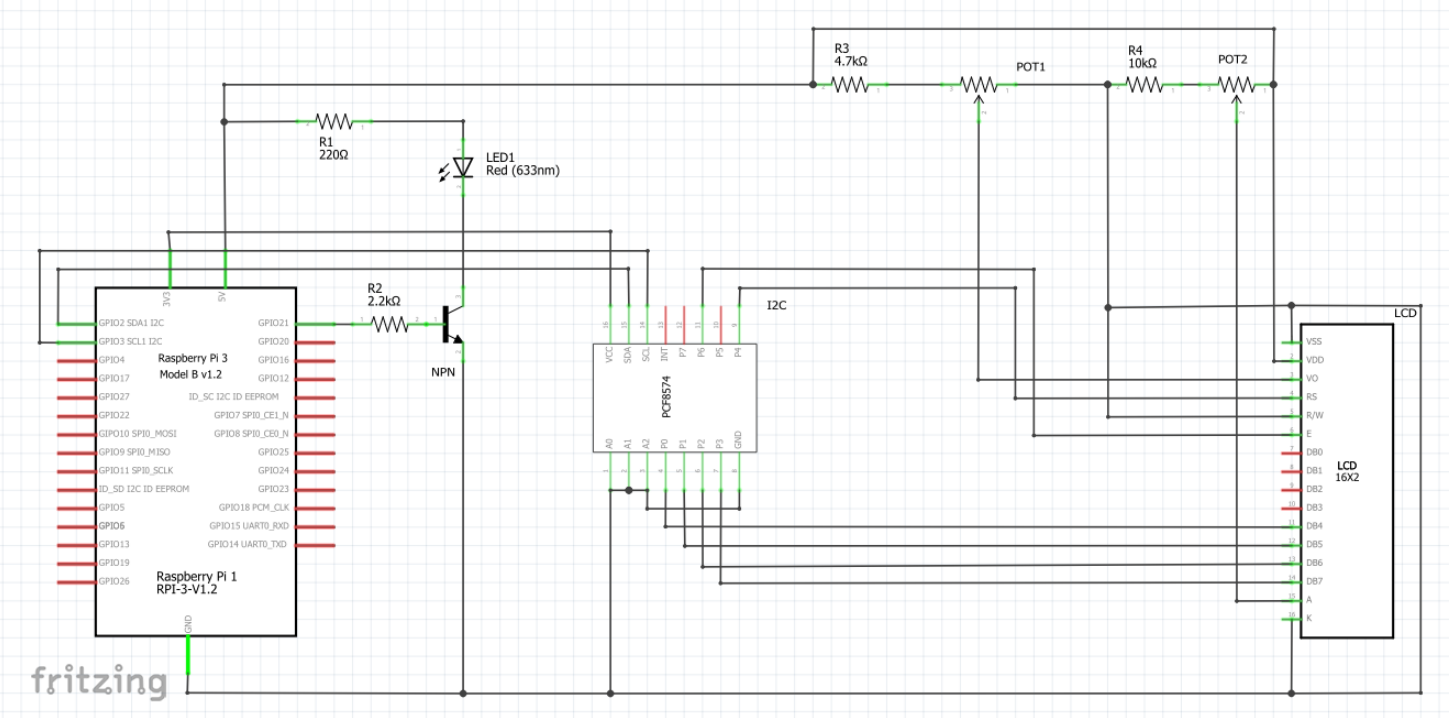

Figure 12. LCD Schematic

The VCNL 4010 proximity/luminosity sensor, previously tested on an STM32 V2 link Blue Pill (an Arduino alternative that uses its IDE), also uses the I2C interface. It just needs to be daisy chained off of the LCD's connections which are the: 3V, SDA, SCL, and ground. To connect to the device through I2C its address is 0x13.

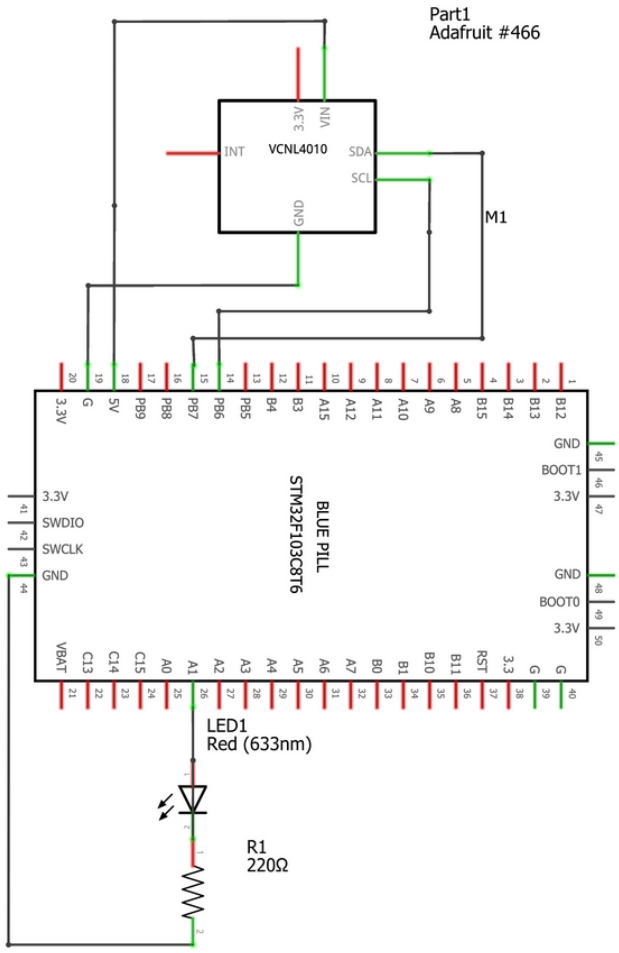

Figure 13. Proximity/Luminosity Sensor

The PN532 controller also uses the I2C interface which is daisy-chained off of the VCNL 4010 using the 3V, SDA, SCL and ground connections. The hardware itself requires a jumper on two sets of pins. Located on the board is labeled SE0 and SE1 where they have 3 pins each. There are several interfaces that can be selected on this NFC controller which include: I2C, UART, and SPI. In this case, I2C is being used so SE0 has to have the jumpers on the ON position while SE1 is switched off. To communicate with the device, it uses a I2C address of 0x24.

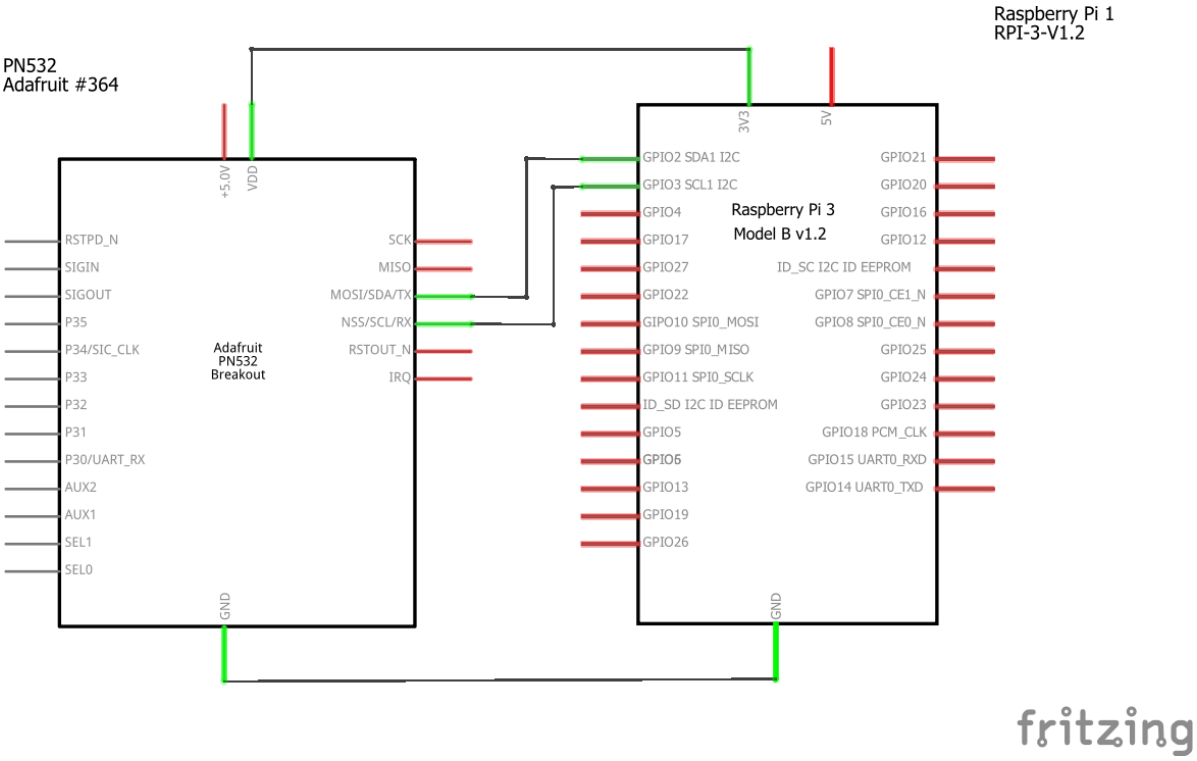

Figure 14. NFC Controller Schematic

Even though the device as a whole currently is operational, there are still some technical difficulties that need to be addressed. The PN532 is the core of the project and is currently experiencing some issues. It is currently reading corrupted data when there is an interruption with an ongoing data transaction. When it is unable to complete a data transaction, it will begin to corrupt the next incoming data transaction as well. Upon further inspection, there is a reset pin on the device. The reset pin can be attached to a GPIO pin on the Raspberry PI to control when to reset the device, but more reading has to be done on the device before any changes are made. Also, the group had also noticed that the backlight did not need to be on all the time especially when no one is using it because it was a waste of energy. The idea of a transistor was brought up to control the backlight and will be later implemented into the design of the project.

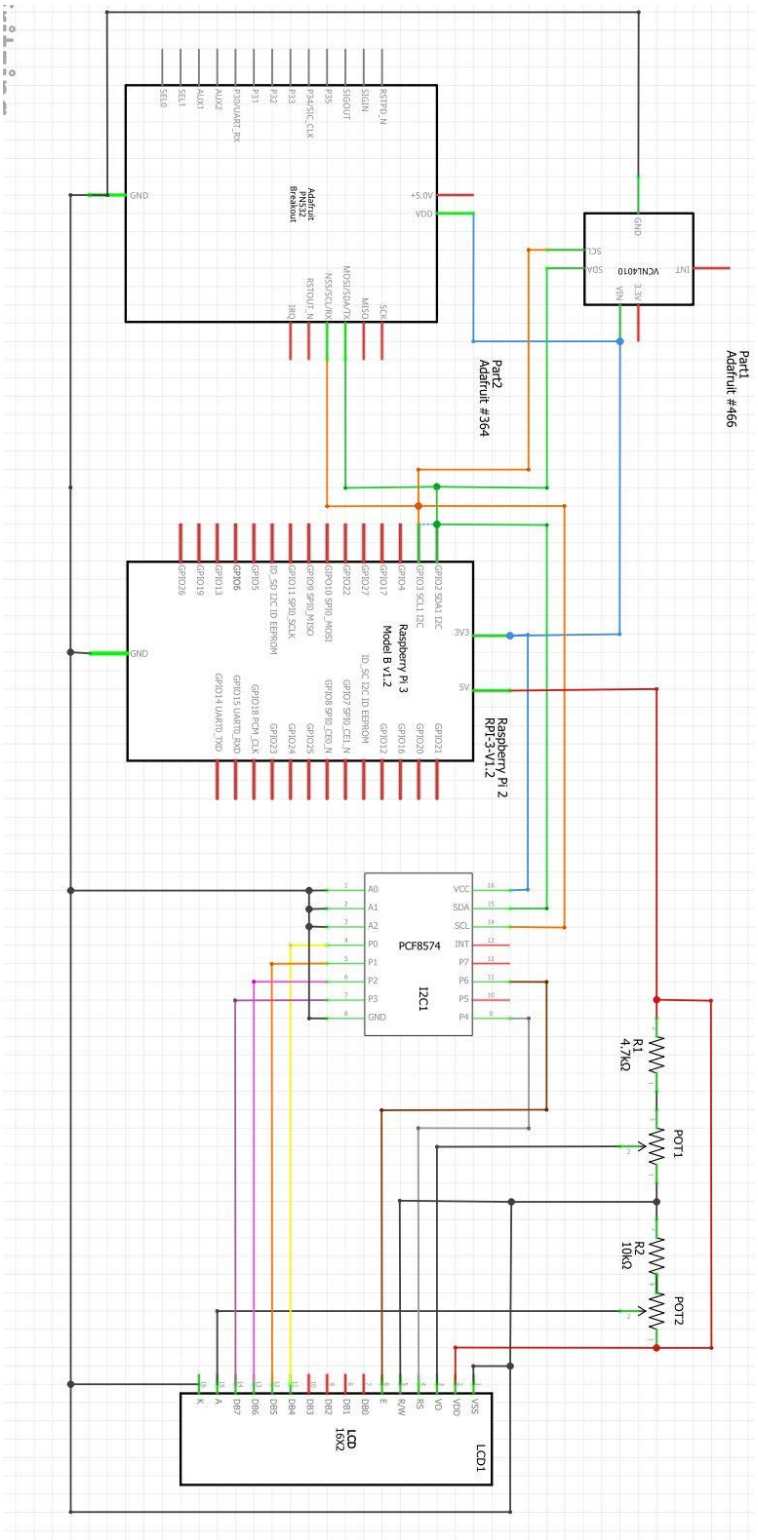

Figure 15. Schematic for all devices wired together

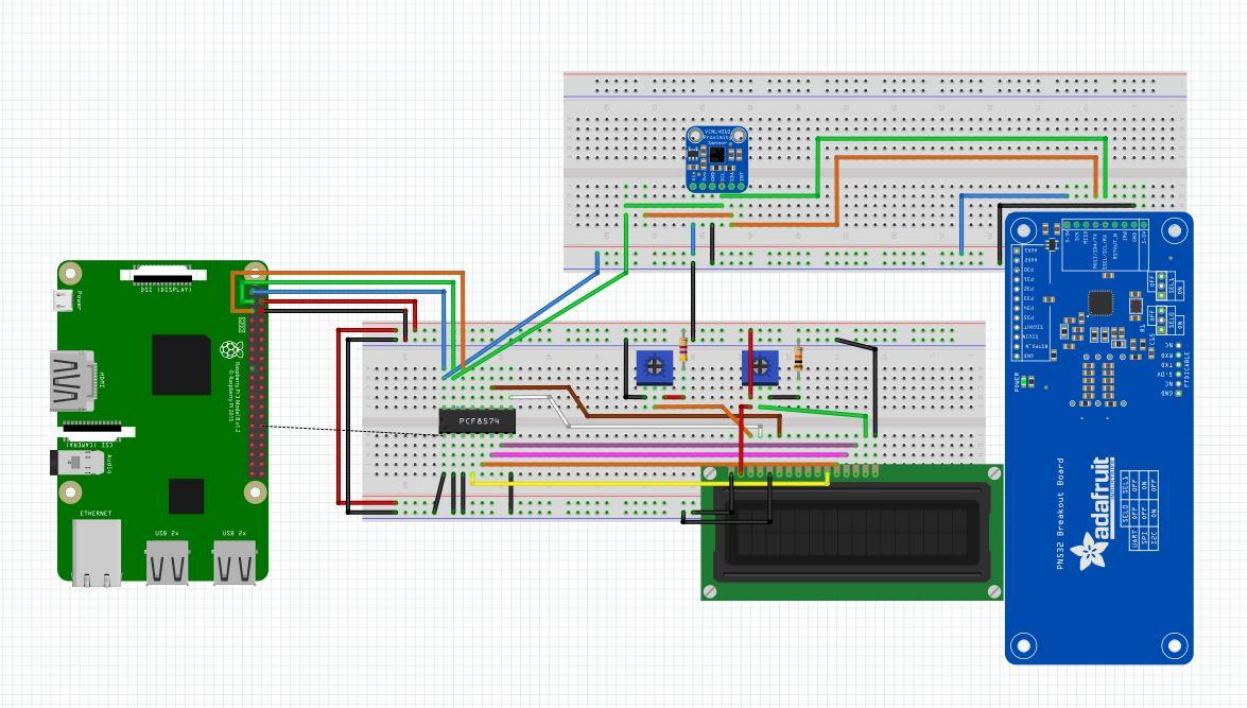

Figure 16. Frizing Visual Layout

3.2.5 Printed Circuit Board

This section documents the PCB created for the prototype. Outlined will be the process as well as figures to explain how the prototype PCB was created.

The first step in the process was taking a look over each of the independent PCBs from the previous semester and removing the components that would not be required for the combined prototype (LEDs, potentiometers). After close inspection, the remaining components were two sensors, the LCD display, the IC chip, and a few resistors to handle the lighting for the LCD. With the initial schematic for the LCD, the back and foreground lights were adjustable with potentiometers but were later replaced with resistors and a 2N4124 NPN transistor that is controlled by a GPIO pin, so that the backlight can be controlled by software. The last step of the planning process was deciding the size of the PCB that would affect the enclosure and the prototype as a whole. The size was determined by the largest component the NFC sensor with the LCD display and proximity sensor sitting above the NFC sensor and the Raspberry Pi underneath to seat the PCB itself.

After the initial preparation, the PCB design was later designed in Fritzing. The wiring necessary for each component was designed to lead into connectors that would have pin connections for only the pins required for each component. Due to the size of the NFC sensor the decision was made to have a wired connection from the connectors to the LCD display and proximity sensor, then the two components are to be seated within the enclosure. This decision was made to prevent the dimensions of the enclosure from being too large.

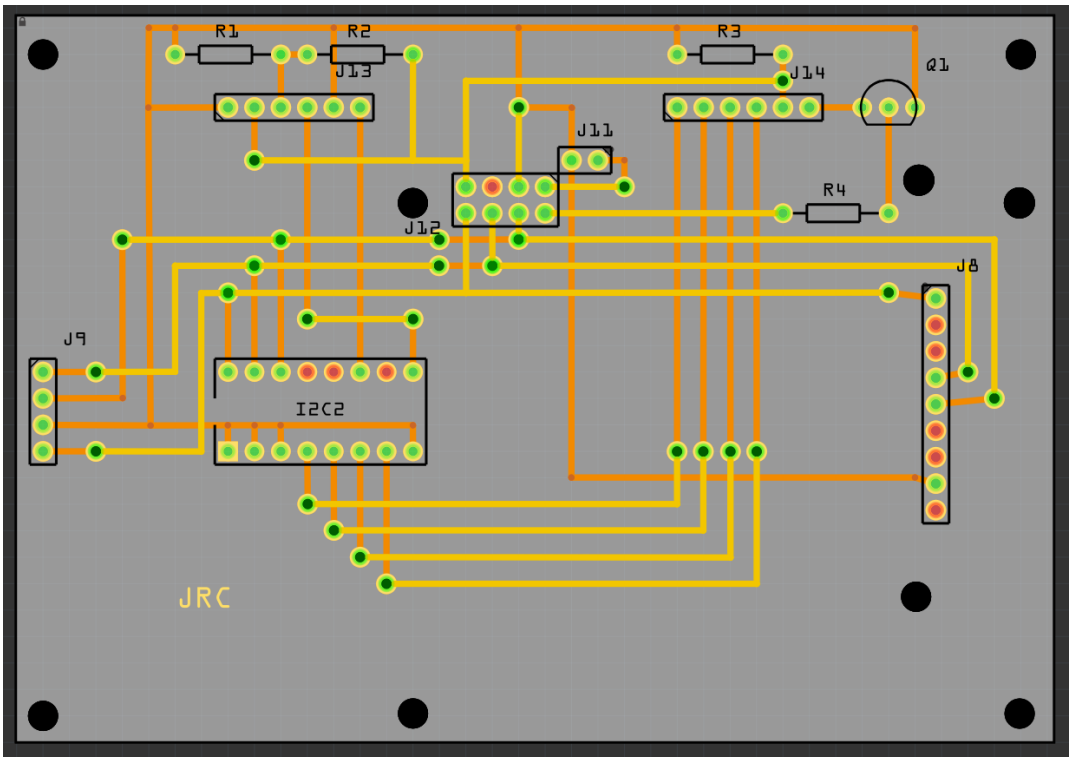

Figure 17. PCB Design

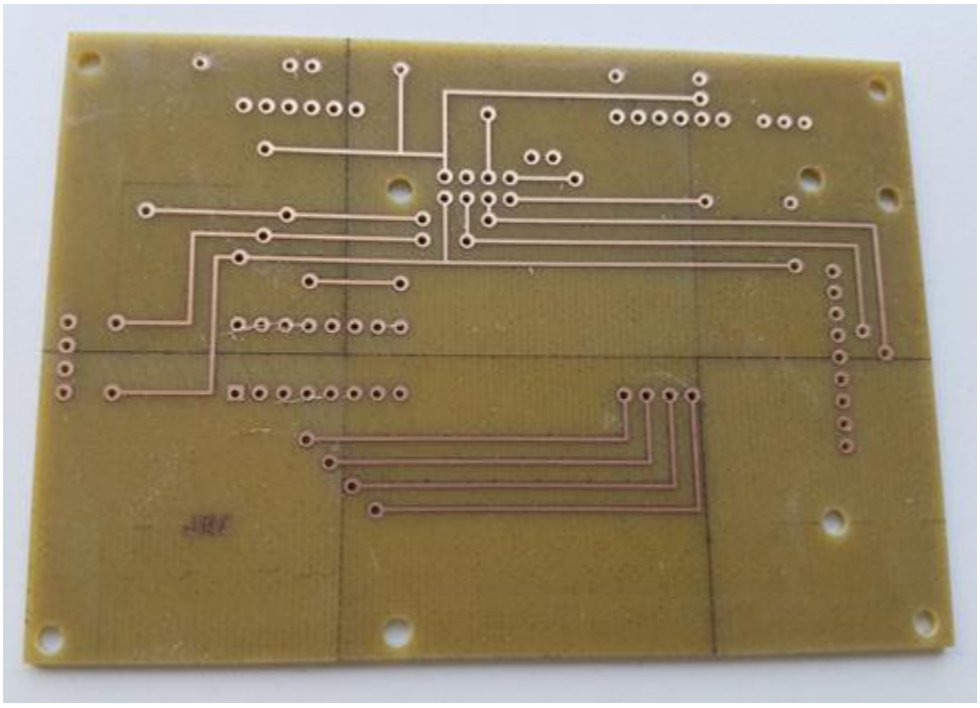

Figure 18 Top of PCB

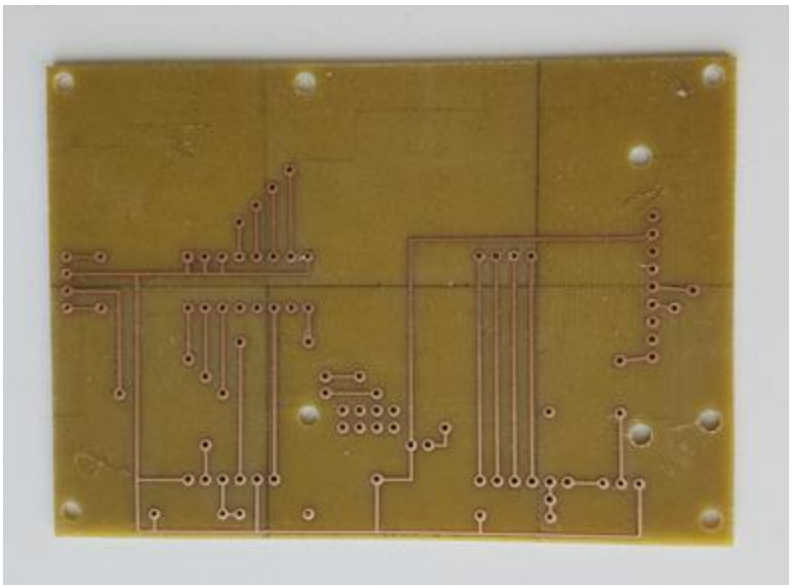

Figure 19 Bottom of PCB

The file that was designed in Fritzing was then submitted to the Humber Prototype lab for the PCB to be developed. The soldering of the connectors, resistors as well as completing the wires through the VIAs onto the PCB took less than two hours to put together.

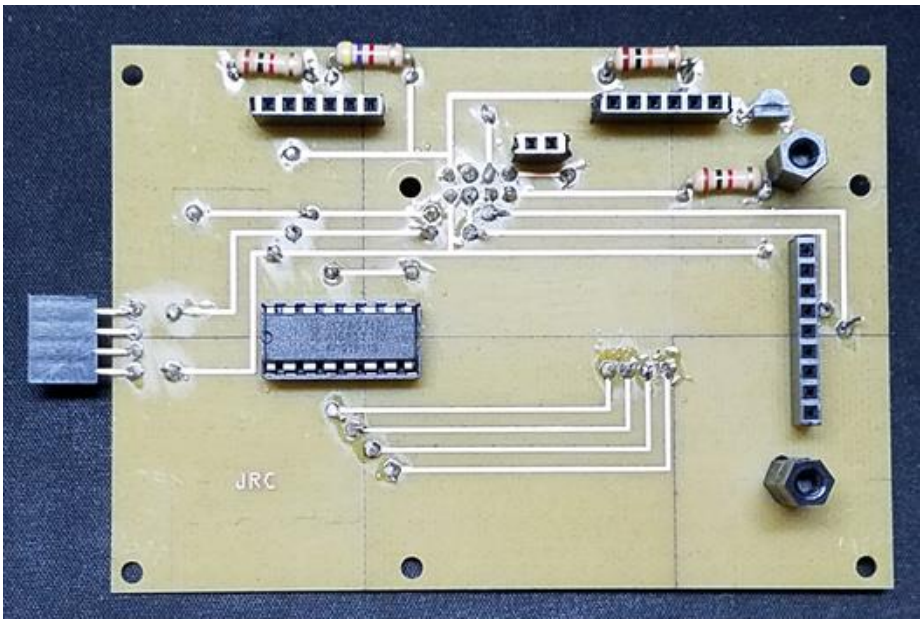

Figure 20 Top of PCB after soldering

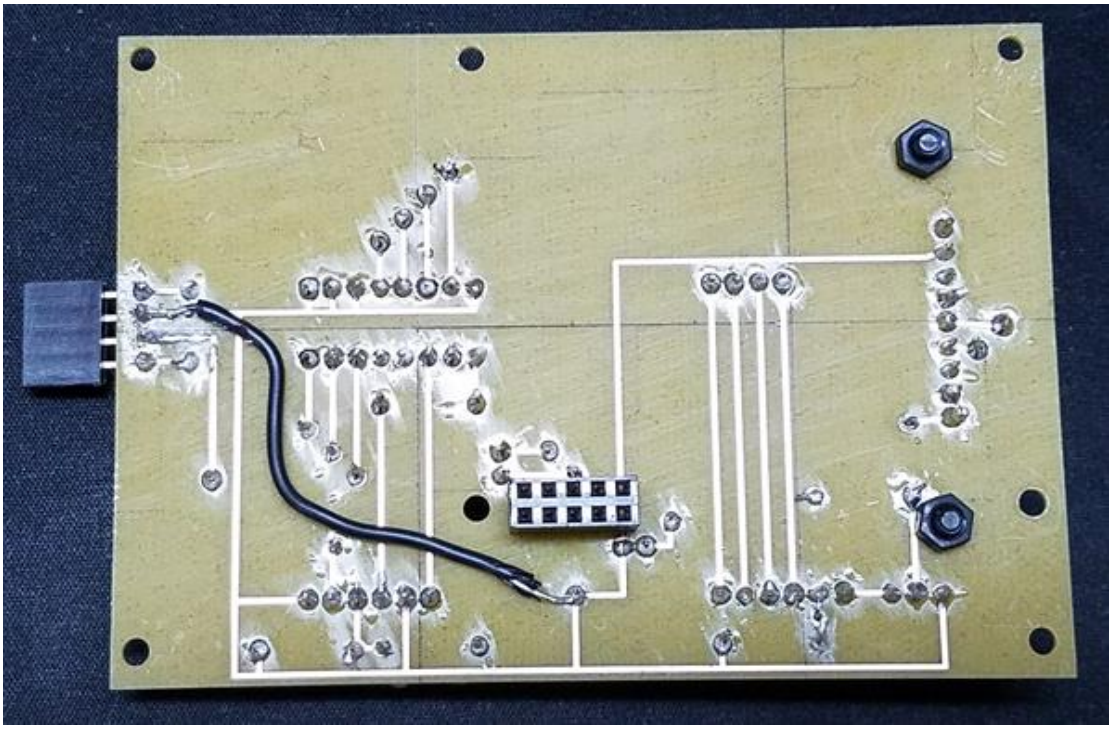

Figure 21. Bottom of PCB after soldering

The testing of the PCB was conducted with a multi-meter to ensure all the copper lines were connected, correct voltages were being delivered and to ensure there were no short circuits. With the success of this testing, the next step was to test the PCB with the main components plugged into the connectors. At this stage, there was one issue that required troubleshooting. The IC chip used for the setup was running at a high temperature. To resolve this issue, a wire was ran from the grounding point near the IC directly to the ground of the female header that is connected to the Raspberry Pi. Further testing showed that all devices had valid I2C addresses shown to the Pi and all components could communicate through the software.

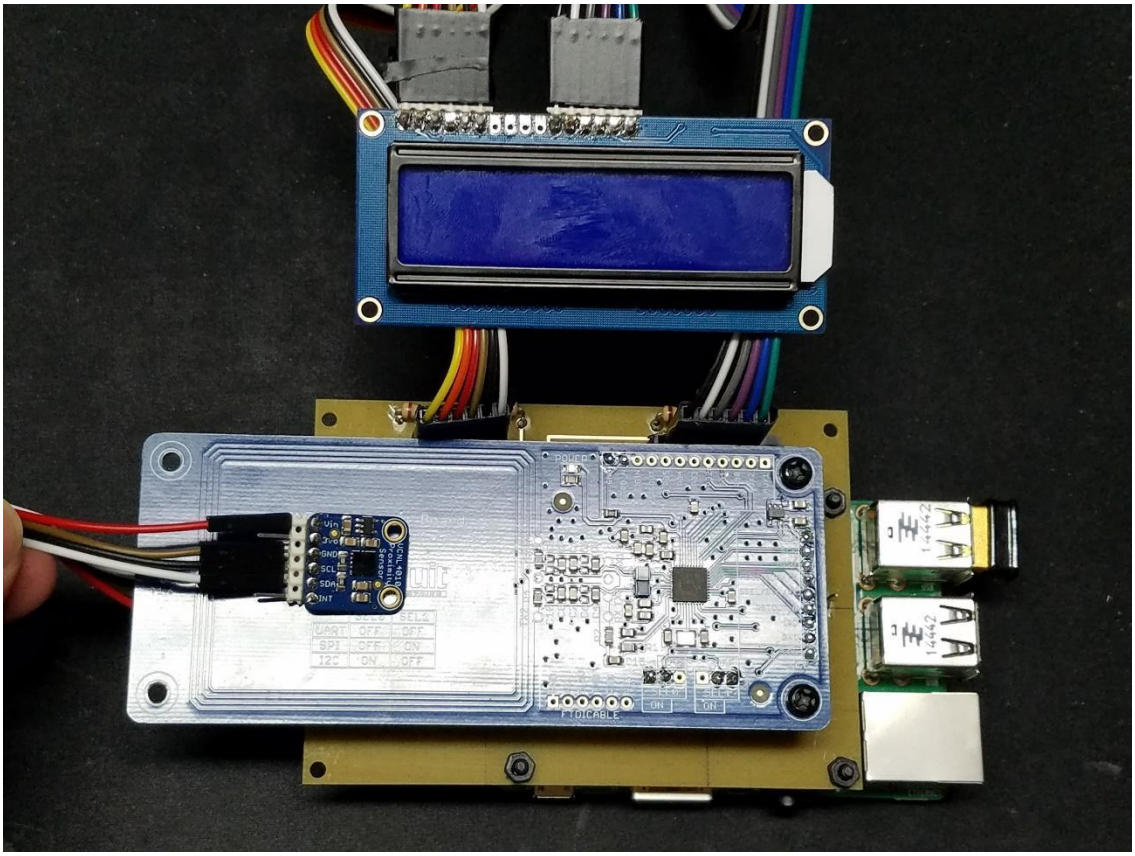

Figure 22. Completed PCB with connected components

3.2.5 Enclosure

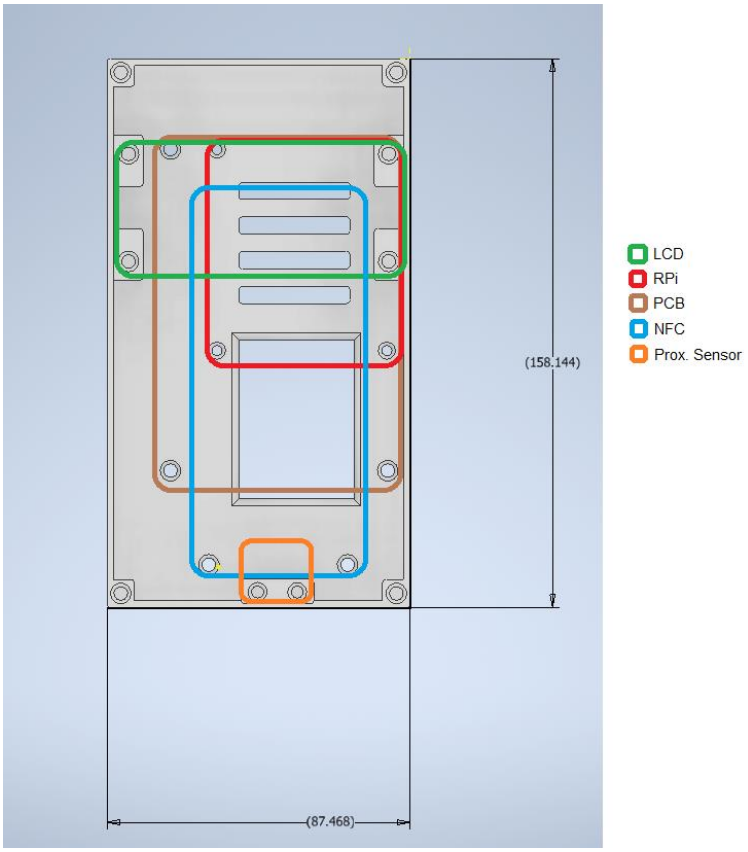

Figure 23. Enclosure Layout

The main base of the enclosure that mounts all the hardware components was 3D printed using ABS plastic while the top lid portion was laser cut using acrylic. The length, width and height (mm) are 158.1, 87.5, 61.0 respectively. The enclosure was designed to be as compact as possible but still leave enough room to insert screws, manage wires and mount devices on top of each other.

The process to develop the device initially required careful measurements to retain functionalities such as when a phone is being sensed or scanned from a distance. After the measurements were taken, the work was then done on the computer to create a 2D layout of the enclosure. The Fritzing PCB diagram that was created before was

exported as a PDF and then modified in AutoCad. Here the measurements that were taken, were drawn and the holes from the PDF were redrawn to a specific size to accommodate heat set threaded inserts. The measurements for the holes were 3mm for the 2.5mm inserts and 4mm for the 3mm inserts. The CAD file was then exported into AutoDesk Inventor where base and walls of the enclosure were extruded. 2 areas were then cut out to expose the I/O ports and micro USB port to make them more accessible. After, standoffs were created for the LCD(4mm), the proximity sensor(3mm) and the top lid(4mm). These standoffs were design with angles less than 45 degrees from the sidewalls. Anything over 45 degrees will cause the 3D printer to start printer supports for the component when it does not have enough time to cool down and hold its shape. When the standoffs were completed, the file was then exported as an STL and sent off to the Prototype Lab to be printed. It took approximately 4-5 hours to be printed using a Stratasys printer and was picked up the next day.

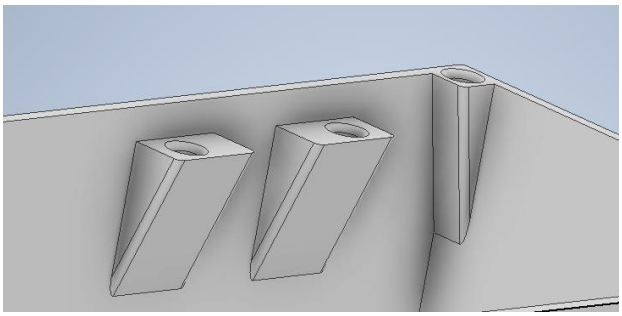

Figure 24. Closeup of standoffs for an LCD and the lid.

Heat-set threaded inserts were then placed into their required holes using a soldering iron with a temperature of 750 degrees Celsius. When the inserts were almost seated, an object with a straight flat surface was used to push the insert flush to the plastic so that screws can be threaded straight. Once all the inserts were in, the hardware

components were then briefly mounted and were only held with several screws to check the fitment of the enclosure.

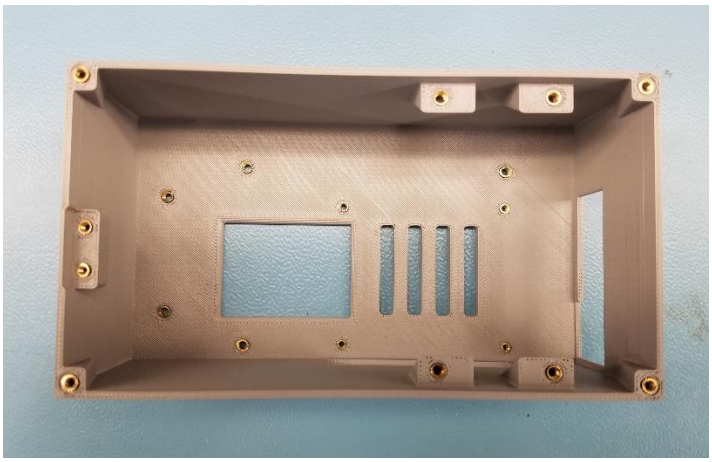

Figure 25. 3D Printed Enclosure with heat set inserts

When everything looked adequate, a lid was then quickly designed in Inventor with the base of the enclosure as reference. The lid was designed to have (4) 3.5mm holes for the 3mmx8mm screws to secure the lid to the standoffs that were created on the walls of the enclosure. It also has a cut out for the LCD to protrude and a small square for the proximity sensor so that it does not fully detect the acrylic that will be laid on top of it. Once completed, only the sketch was exported to a DXF file and sent to the Prototype Lab for laser cutting on a piece of clear acrylic that was 3mm thick. The laser cutting was done in approximately 10 minutes and the enclosure with the components were taken apart for reassembly and wire management.

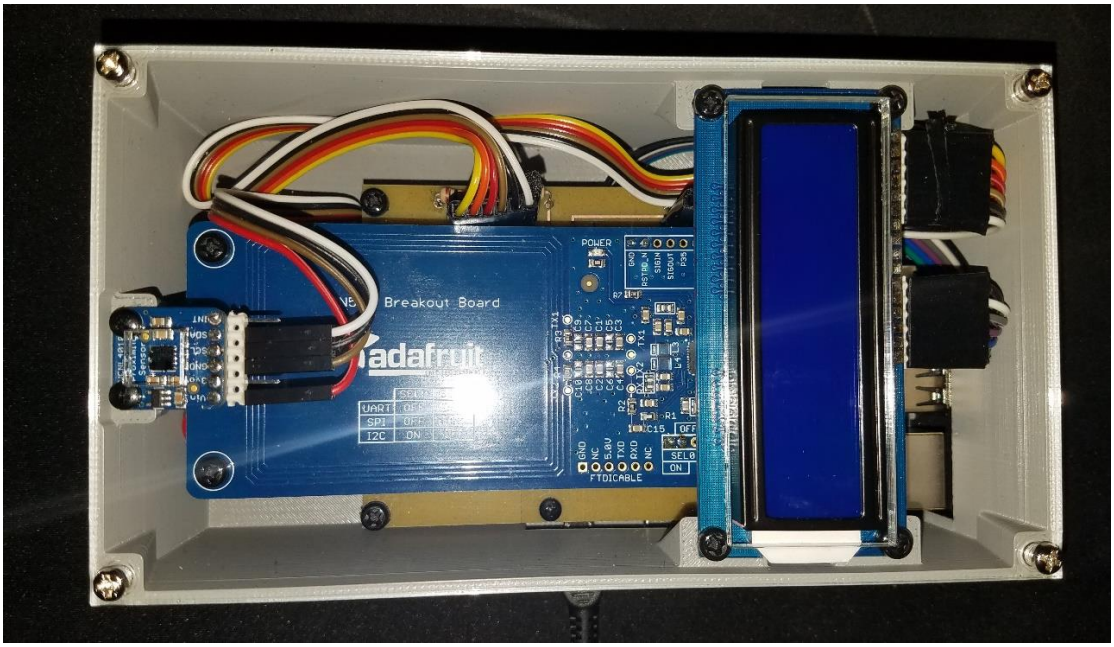

Figure 26. Assembled enclosure with wired hardware components.

The assembly of the enclosure included the following materials as seen in Figure 23.

Hardware	**Materials**	**Notes**
Raspberry Pi 3 B+	4 M2.5*3mm Inserts 4 M2.5*12+6mm standoffs 3 M2.5 nuts	The standoffs are attached to the Raspberry Pi with the nuts. With the remaining threads extending out from the nuts, these are then threaded into the heat set inserts of the enclosure.
PCB	2x M3*10+6mm standoffs 2x M3*6+6mm standoffs 2x M3 nuts 3 2 M3*3mm Inserts 3 M3x15+6mm standoffs 3 M3 screws 4 M2.5+6mm screws	One M3*10 and one M3*6 are threaded together to create two M3*16 standoffs. These standoffs are attached to the PCB with the M3 nuts so that the NFC reader can later be mounted. The M3*15 standoffs are threaded into the heat set inserts of the enclosure. The PCB is then mounted onto the Raspberry Pi and the standoffs that are mounted to enclosure. The PCB is then attached with 4 M2.5 screws to keep them in place.

PN532(NFC Reader)	2 M3*3mm Inserts 2x M3*20+6mm standoffs 2x M3*12+6mm standoffs 4x M3+6mm screws	One M3*20 and one M3*12 are threaded together to create two M3*32 standoffs. These are then threaded into the heat set inserts of the enclosure. The NFC reader is then mounted to the 2 standoffs on the PCB and the 2 standoffs mounted to the enclosure
VNCL 4010 Proximity Sensor	2 M2.5*3mm Inserts 2 M2.5+6mm screws	2 M2.5 screws are used to mount the device to the heat set inserts of the enclosure.
Top (Lid)	4 M3+8mm screws	4 M3+8mm screws are used to mount the top lid onto the enclosure.

3.3 Integration

3.3.1 Database

The database is stored in a MySQL database and currently stored on Humber College's Apollo Linux server. The tables themselves are all created with SQL scripts that can be moved to another active Linux Server if necessary. The database consists of two main tables with three subsidiary tables that create entries based on the different functionality of the system.

The main tables are 'accountsJRC' used for storing account information of each user and 'Inventory' for holding data for each item registered in the system. Every account has a unique character string (UID) tied to each entry acting as the primary key distinguishing each account along with personal data for administrative reference and stored encrypted credentials. All the inventory has a unique number (SID) to identify each item as the primary key, information to identify each item in a human readable

form is also available in each entry as well as a quantity value to maintain availability of each item.

The first subsidiary table is ‘Transaction’ used to maintain every order created by user interaction. Each order has a generated transaction ID (TID) tied to each individual order and gains foreign keys from the two main table’s primary keys as foreign keys to identify the user the order is tied to and every item associated with it and the quantity of each. The other fields contain a timestamp for logging interactions and a status field to execute the desired functionality of the system.

The next table is ‘ListHead’ that is used to save lists so users can use for efficient functionality when creating orders. Each entry has a unique ID (TID) to identify in the system with the rest of the fields storing detailed information about the list being stored. The primary key from ‘accountsJRC’ is used to keep track of the creator of each entry - this field and the ‘course’ field can be used to organize lists for more efficient searching by clients.

The last table is ‘ListItems’ used to store the items and quantities associated in each entry of saved list. They are grouped by the TID received from ‘ListHead’ to organize all items in one list. Each item is stored by the SID value associated with the inventory and a quantity desired in the list.

All the tables are interacted through multiple PHP scripts that perform all the functions required for the system’s functionality. They are only accessible when certain fields are given to each particular script. This is done so the fields can be filtered server side to prevent database injections and to control the type of interaction that can be done with the database.

3.3.2 Unit Testing and Production Testing

Initial testing requires every PHP script to execute its intended functions. Each file is tested with web browser and access to the server through SSH connection to monitor the results and output. Afterwards the devices interacting with the scripts (mobile device, NFC Reader, web application) all require the ability to post requests and then ensure each device can post or retrieve the necessary information from each function.

Testing the entire system requires the devices involved to be able to temporarily store the ID of the transaction. Upon interaction with each other it will change the state of each order to communicate to both client and administrator the actions required during each state. Every device interacts with the orders differently through each state. The mobile application puts the transaction into a “P” state - which stands for a pending order. This allows the employee to double check their inventory beforehand and prep the order before the person arrives at the point of service’. The hardware prototype has 2 states: “Q” - which stands for “Queued” and “QR” - which stands for “Queued for Return”. This is activated when the person arrives at the point of service, taps their phone with the application open and sends the NDEF message containing the transaction ID. If the message starts with an ‘R’ that means the person who scanned their phone intends to return their items. To finalize the transaction, the employee at the point of service will need to double check any items coming in or out. Here a web application will be used to set the state of the transaction to “O” - which stands for “Out” and “R” - which stands for returned.

The hardware prototype created for this system contains a check sequence to ensure that the components connected via I2C are active as well as the Wi-Fi/internet

connection allowing the device to communicate to the database. If there are issues, a notification would be displayed in the terminal or displayed on the LCD. The check sequence is important to reduce the amount of time for technical troubleshooting and can be used during production testing to speed up the process.

The mobile application is tested through ensuring that it can interact smoothly with the tables in the database and post/receive required information. All functionality in the application is tested individually first, then with each other to ensure data is being controlled as intended. Finally, the application is tested interacting with the hardware to complete the process.

3.3.3 Network and Security

For each device their connection to networks is crucial to setup specifically to control security and ensure no unintended breaches can be made. Currently the hardware is provided with a WiFi certificate in the configuration files to ensure secure connection to the Humber College network. This is important for setup per product to keep the log-in credentials encrypted specifically to each business. Increasing secure network access is essential involving any database that stores data for businesses or personal information. Currently through testing the hardware is assigned a reserved IP address when the MAC address is identified and used to troubleshoot issues via SSH.

Every interaction from clients must be limited to specifically control access to the database to only what is required. Account information and every transaction is identified with uniquely generated character strings. No access is permitted unless an account is recognized with their corresponding hashed password that is generated and recreated with the PHP scripts stored on the server. To use this product in a business,

heavy consideration over security is essential to protect all data used by the respective business.

Currently, the server is only utilizing http which is unsafe as anyone with access to applications similar to Wireshark can see what data is flowing from the hardware prototype or from the mobile application. To increase security for businesses, HTTPS with a SSL certificate must be used to encrypt the data being transferred so that it adds a layer of security that prevents anyone from seeing the data over a network.

4.0 Results and Discussion

Over several months of development, troubleshooting, and testing, the final product had to be demonstrated. Even though the basic concept and the components of the system have been completed there are certain things that still need to be addressed. Due to unforeseen circumstances, the project was unable to reach its full potential. The current hardware prototype is operational but was only tested within known conditions. Due to the general experience of the team and lack of facilities to test in, there may have been certain factors that may have been missed with the hardware. Also, since the team had only several months of experience with Android Visual Studio, and troubleshooting NFC protocols had taken several weeks to create a stable working application without it completely crashing. The app does crash but it is a rare issue that has yet to be solved because the cause is still unknown. With all the troubleshooting that had to be done between the hardware and the Android application, there was a limited amount of time that had remained for the web application which was not a requirement for the course. Only simple aspects of the web application were done to display the communication of data between the hardware, the mobile application, and the webserver /database. Regardless of its current flaws, it is able to demonstrate a basic concept of how the components of the NFC inventory Management System interact with each other through requesting items, tapping the mobile device to a built hardware prototype with an NFC reader, and the data within the database being managed over a web application. The app still maintains its core functions of singing in and out parts.

5.0 Conclusions

To make this solution a more viable one and applicable in the industry, the size must be reduced because the size in its current state would seem very unappealing. More research and development would be required to put all components on to a single PCB. This would require understanding how the PN532 NFC reader and VCNL410 proximity sensor chips operate to eliminate unnecessary interfaces that are not required for this solution such as SPI and UART. Also, developing coiled wires for the NFC reader is essential for the PCB because it will determine how far the mobile device can be picked up when it is being scanned. When all of these things are put onto a single PCB it will reduce the number of wires, headers, and solder which will also reduce the price to create the prototype when components are bought in bulk. In addition, with the compactness of the PCB, a smaller enclosure will need to be designed and tested so that it operates similarly to the current prototype. Following these steps will help in the production of the new prototype in much larger volumes.

With the app in its final stages of troubleshooting and polishing, other applications of the NFC Inventory Management System need to be found. Even though the project overall revolves around handling inventory, adjustments can be made to the system so that it can be applied to any industry that involves cards and/or memberships. To further the improvement of this system, it will be made open-sourced for anyone to contribute and/or use for their implementations of the system. Making it open source will allow the concept to gain popularity and be a more common concept in the near future.

6.0 References

Butterfield, A., & Szymanski, J. (2018). *In A Dictionary of Electronics and Electrical Engineering.* Oxford University Press.

Hasing Security. (2019, June 5). Retrieved from Crack Station: https://crackstation.net/hashing-security.htm

OACETT. (2017, March). *I need to Complete a Technology Report*. Retrieved from The Ontario Association of Certified Engineering Technicians and Technologists: https://www.oacett.org/Membership/Technology-Report-and-Seminar

Sabella, R. (2016, April 11). *NFC for dummies*. Retrieved from dummies: https://www.dummies.com/consumer-electronics/nfc-standard-isoiec-14443/

Vanderkay, J. (2004, March 18). *Nokia, Philips And Sony Establish The Near Field Communication (NFC) Forum*. Retrieved from NFC Forum: https://nfc-forum.org/nokia-philips-and-sony-establish-the-near-field-communication-nfc-forum/

7.0 Appendix

7.1 Firmware code

7.1.1 main.c File Reference

Main Program that will run the VNCL 4010, PN532 and LCD.

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <unistd.h>
#include <nfc/nfc.h>
#include "Presence.h"
#include "NDEFData.h"
```

Functions

- int main (void)
  Function used to initialize Presence and setup the NFC reader.

Detailed Description

Main Program that will run the VNCL 4010, PN532 and LCD.

Author

Robert Dinh

Date

01MAR2020

Definition in file main.c.

Function Documentation

int main (void)

Function used to initialize Presence and setup the NFC reader.

The function sets up the VNCL4010 and PN532 using the Wiring Pi library through I2C. It then sets up the registers of the devices so that data can be retrieved. When a presence is detected by the VNCL4010 it triggers the NFC reader into a read state to to grab an NDEF encoded message from a mobile device.

Definition at line 27 of file main.c.

main.c

```
00001
00008 #include <stdio.h>
00009 #include <stdlib.h>
00010 #include <signal.h>
00011 #include <string.h>
00012 #include <unistd.h>
00013 #include <nfc/nfc.h>
00014 #include "Presence.h"
00015 #include "NDEFData.h"
00016
00017
00027 int main (void)
00028 {
00029     initNFC();
00030     int fd = initPresence();
00031
00032     int present = 0;
00033
00034     fprintf(stdout,"%d \n",fd);
00035     present = detectPresence(fd);
00036     if (present)
00037         getNdef();
00038
00039     stopFunction(2);
00040     return 0;
00041 }
```

7.1.2 NDEFData.c File Reference

Various Functions to read an NDEF message from a mobile device.

```
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <signal.h>
#include <nfc/nfc.h>
```

Functions

- void stopFunction (int sig)
  Function used to stop the NFC reader in the case of a System Interrupt.

- void initNFC (void)
  Function used to initialize the PN532 NFC Reader.

- int CardTransmit (nfc_device *pnd, uint8_t *capdu, size_t capdulen, uint8_t *rapdu, size_t *rapdulen)
  Function used to send APDU commands.

- void getNdef (void)
  Function used to initialize Presence.

Variables

- nfc_device * **pnd**
- nfc_context * **context**

Detailed Description

Various Functions to read an NDEF message from a mobile device.

Author

Robert Dinh

Date

01MAR2020

The following does contain non-original code but was modified to suit certain needs. Original code was developed by NFC - TOOLS. http://nfc-tools.org/index.php/Libnfc:APDU_example

Definition in file NDEFData.c.

Function Documentation

*int CardTransmit (nfc_device * pnd, uint8_t * capdu, size_t capdulen, uint8_t * rapdu, size_t * rapdulen)*

Function used to send APDU commands.

Definition at line 70 of file NDEFData.c.

void getNdef (void)

Function used to initialize Presence.

Definition at line 106 of file NDEFData.c.

void initNFC (void)

Function used to initialize the PN532 NFC Reader.

Definition at line 42 of file NDEFData.c.

void stopFunction (int sig)

Function used to stop the NFC reader in the case of a System Interrupt.

Definition at line 28 of file NDEFData.c.

NDEFData.c

```
00001
00013 #include <stdlib.h>
00014 #include <string.h>
00015 #include <unistd.h>
00016 #include <signal.h>
00017 #include <nfc/nfc.h>
00018
00019 //NFC Variables
00020  nfc_device *pnd;
00021  nfc_context *context;
00022
00023
00028 void stopFunction(int sig)
00029 {
00030
00031     printf("Stopping now.\n");
00032     nfc_close(pnd);
00033     nfc_exit(context);
00034     exit(EXIT_SUCCESS);
00035
00036 }
00037
00042 void initNFC(void)
```

```
00043 {
00044     signal(SIGINT,stopFunction);
00045     nfc_init(&context);
00046 
00047     if (context == NULL) {
00048         printf("Unable to init libnfc (malloc)\n");
00049         exit(EXIT_FAILURE);
00050     }
00051 
00052     pnd = nfc_open(context, NULL);
00053 
00054     if (pnd == NULL) {
00055         printf("ERROR: %s", "Unable to open NFC device.");
00056         exit(EXIT_FAILURE);
00057     }
00058 
00059     if (nfc_initiator_init(pnd) < 0) {
00060         nfc_perror(pnd, "nfc_initiator_init");
00061         exit(EXIT_FAILURE);
00062     }
00063 
00064 }
00065 
00070 int CardTransmit(nfc_device *pnd, uint8_t * capdu, size_t capdulen, uint8_t *
rapdu, size_t * rapdulen)
00071 {
00072     int res;
00073     size_t  szPos;
00074 
00075     printf("=> ");
00076     for (szPos = 0; szPos < capdulen; szPos++)
00077     {
00078         printf("%02x ", capdu[szPos]);
00079     }
00080     printf("\n");
00081 
00082     if ((res = nfc_initiator_transceive_bytes(pnd, capdu, capdulen, rapdu,
*rapdulen, 500)) < 0)
00083     {
00084         return -1;
00085     }
00086     else
00087     {
00088         *rapdulen = (size_t) res;
00089         printf("<= ");
00090 
00091         for (szPos = 0; szPos < *rapdulen; szPos++)
00092         {
00093           printf("%02x ", rapdu[szPos]);
00094         }
00095 
00096     printf("\n");
00097     return 0;
00098     }
00099 }
00100 
00101 
00106 void getNdef(void)
00107 {
00108     nfc_target nt;
00109 
00110 
00111     const nfc_modulation nmMifare = {
00112     .nmt = NMT_ISO14443A,
00113     .nbr = NBR_106,
00114     };
00115 
00116     printf("Polling for target...\n");
00117 
00118     while (nfc_initiator_select_passive_target(pnd, nmMifare, NULL, 0, &nt) <=
0);
00119     printf("Target detected!\n");
00120 
00121     uint8_t capdu[264];
00122     size_t capdulen;
00123     uint8_t rapdu[264];
00124     size_t rapdulen;
```

```
00125
00126
00127     // Select application
00128     memcpy(capdu, "\x00\xA4\x04\x00\x07\xF0\x39\x41\x48\x14\x81\x00\x00", 13);
00129     capdulen=13;
00130     rapdulen=sizeof(rapdu);
00131     if (CardTransmit(pnd, capdu, capdulen, rapdu, &rapdulen) < 0)
00132         exit(EXIT_FAILURE);
00133     if (rapdulen < 2 || rapdu[rapdulen-2] != 0x90 || rapdu[rapdulen-1] != 0x00)
00134         exit(EXIT_FAILURE);
00135
00136     printf("Application selected!\n");
00137
00138     // Select Capability Container
00139     memcpy(capdu, "\x00\xa4\x00\x0c\x02\xe1\x03", 7);
00140     capdulen=7;
00141     rapdulen=sizeof(rapdu);
00142     if (CardTransmit(pnd, capdu, capdulen, rapdu, &rapdulen) < 0)
00143         exit(EXIT_FAILURE);
00144     if (rapdulen < 2 || rapdu[rapdulen-2] != 0x90 || rapdu[rapdulen-1] != 0x00)
{
00145         capdu[3]='\x00'; // Maybe an older Tag4 ?
00146     if (CardTransmit(pnd, capdu, capdulen, rapdu, &rapdulen) < 0)
00147         exit(EXIT_FAILURE);
00148     }
00149
00150     printf("Capability Container selected!\n");
00151
00152     // Read Capability Container
00153     memcpy(capdu, "\x00\xb0\x00\x00\x0f", 5);
00154     capdulen=5;
00155     rapdulen=sizeof(rapdu);
00156     if (CardTransmit(pnd, capdu, capdulen, rapdu, &rapdulen) < 0)
00157         exit(EXIT_FAILURE);
00158     if (rapdulen < 2 || rapdu[rapdulen-2] != 0x90 || rapdu[rapdulen-1] != 0x00)
00159         exit(EXIT_FAILURE);
00160
00161     printf("Capability Container header:\n");
00162
00163     size_t  szPos;
00164     for (szPos = 0; szPos < rapdulen-2; szPos++)
00165     {
00166         printf("%02x ", rapdu[szPos]);
00167     }
00168     printf("\n");
00169
00170     // NDEF SELECT
00171     memcpy(capdu, "\x00\xA4\x00\x0C\x02\xE1\x04", 7);
00172     capdulen=7;
00173     rapdulen=sizeof(rapdu);
00174     if (CardTransmit(pnd, capdu, capdulen, rapdu, &rapdulen) < 0)
00175         exit(EXIT_FAILURE);
00176     if (rapdulen < 2 || rapdu[rapdulen-2] != 0x90 || rapdu[rapdulen-1] != 0x00)
00177         exit(EXIT_FAILURE);
00178
00179     printf("NDEF SELECTED!\n");
00180
00181     // NDEF Read Binary
00182     memcpy(capdu, "\x00\xb0\x00\x00\x02", 5);
00183     capdulen=5;
00184     rapdulen=sizeof(rapdu);
00185     if (CardTransmit(pnd, capdu, capdulen, rapdu, &rapdulen) < 0)
00186         exit(EXIT_FAILURE);
00187     if (rapdulen < 2 || rapdu[rapdulen-2] != 0x90 || rapdu[rapdulen-1] != 0x00)
00188         exit(EXIT_FAILURE);
00189     printf("NDEF Read Binary NLEN!\n");
00190
00191     //READING NDEF DATA
00192     memcpy(capdu, "\x00\xb0\x00\x00\x0f", 5);
00193     capdulen=5;
00194     rapdulen=sizeof(rapdu);
00195     if (CardTransmit(pnd, capdu, capdulen, rapdu, &rapdulen) < 0)
00196         exit(EXIT_FAILURE);
00197     if (rapdulen < 2 || rapdu[rapdulen-2] != 0x90 || rapdu[rapdulen-1] != 0x00)
00198         exit(EXIT_FAILURE);
00199
00200     printf("NDEF DATA ! \n\n");
```

```
00201     size_t  szPos2;
00202     char ndefMsg[100] = "";
00203     char chr[1] = "\0";
00204     int counter = 0;
00205 
00206     for (szPos2 = 8; szPos2 < rapdulen-2; szPos2++) {
00207         ndefMsg[counter]=(char)rapdu[szPos2];
00208         counter++;
00209     }
00210 
00211     printf("%s",ndefMsg);
00212 
00213     char cmdS[100] = "python lcd.py Phone Scanned\n";
00214 
00215 
00216     system(cmdS);
00217     sleep(1);
00218     printf("\nDone...\n");
00219 }
```

NDEFData.h File Reference

Function prototypes for NDEFData.

Detailed Description

Function prototypes for NDEFData.

Author

Robert Dinh

Date

01MAR2020

Definition in file NDEFData.h.

NDEFData.h

```
00001
00008 #ifndef NDEFData_H
00009 #define NDEFData_H
00010
00011
00013 // Function Prototypes
00014 void stopFunction(int sig);
00015 void initNFC(void);
00016 int CardTransmit(nfc_device *pnd, uint8_t * capdu, size_t capdulen, uint8_t *
rapdu, size_t * rapdulen);
00017 void getNdef(void);
00019
00020
00021
00022 #endif
```

7.1.3 Presence.c File Reference

Various functions to aid in Presence Detection with the VNCL4010.

```
#include <stdio.h>
#include <wiringPiI2C.h>
#include <unistd.h>
#include "Presence.h"
```

Functions

- int initPresence (void)
  Function used to initialize Presence.

- int detectPresence (int fd)
  Function used to detect the presence of a person.

- int getProximity (int fd)
  Function used get the proximity of the sensor.

- int getLuminosity (int fd)
  Function used get the luminosity of the sensor.

Detailed Description

Various functions to aid in Presence Detection with the VNCL4010.

Author

Robert Dinh

Date

01MAR2020

Definition in file Presence.c.

Function Documentation

int detectPresence (int fd)

Function used to detect the presence of a person.

Definition at line 44 of file Presence.c.

int getLuminosity (int fd)

Function used get the luminosity of the sensor.

Definition at line 80 of file Presence.c.

int getProximity (int fd)

Function used get the proximity of the sensor.

Definition at line 65 of file Presence.c.

int initPresence (void)

Function used to initialize Presence.

The fuction sets up the VNCL4010 using the Wiring Pi library through I2C. It then sets up the registers of the device so that data can be retrieved.

Definition at line 20 of file Presence.c.

Presence.c

```
00001
00008 #include <stdio.h>
00009 #include <wiringPiI2C.h>
00010 #include <unistd.h>
00011 #include "Presence.h"
00012
00013
00020 int initPresence(void)
00021 {
00022     //Setup Wiring Pi thorugh I2C
00023     int fd = wiringPiI2CSetup(0x13);
00024
00025     //Select Command Register
00026     wiringPiI2CWriteReg8(fd,0x80,0xFF);
00027
00028     //Select Prox Register
00029     wiringPiI2CWriteReg8(fd,0x82,0x03);
00030
00031     //Select Current for IR LEDto 200ma
00032     wiringPiI2CWriteReg8(fd,0x83,0x14);
00033
00034     //Select ALS Register
00035     wiringPiI2CWriteReg8(fd,0x84,0x9D);
00036
00037     return fd;
00038 }
00039
00044 int detectPresence(int fd)
00045 {
00046         presenceD pData  = {0};
00047         while (1)
00048         {
00049             pData.proximity = getProximity(fd);
00050             pData.luminosity = getLuminosity(fd);
00051
00052             fprintf(stdout,"Proximity:\t%d\t
Luminosity:\t%d\n",pData.proximity,pData.luminosity);
00053             if (pData.proximity > 2300)
```

```
00054                 return pData.proximity;
00055
00056             usleep(300000);
00057         }
00058
00059 }
00060
00065 int getProximity(int fd)
00066 {
00067     int prox1= 0;
00068     int prox2= 0;
00069
00070     prox1 = wiringPiI2CReadReg8(fd,0x87) *256; //Upper Byte
00071     prox2 = wiringPiI2CReadReg8(fd,0x88); //Lower Byte
00072
00073     return prox1+prox2;
00074 }
00075
00080 int getLuminosity(int fd)
00081 {
00082     int lux1= 0;
00083     int lux2= 0;
00084
00085     lux1 = wiringPiI2CReadReg8(fd,0x85) *256; //Upper Byte
00086     lux2 = wiringPiI2CReadReg8(fd,0x86); //Lower Byte
00087
00088     return lux1 + lux2;
00089 }
```

Presence.h File Reference

Constants, structures, function prototypes for Presence.

```
#include <wiringPiI2C.h>
```

Data Structures

- struct PresenceData

Typedefs

- typedef struct PresenceData **presenceD**

Detailed Description

Constants, structures, function prototypes for Presence.

Author

Robert Dinh

Date

01MAR2020

Definition in file Presence.h.

Presence.h

```
00001
00008 #ifndef PRESENCE_H
00009 #define PRESENCE_H
00010
00011 #include <wiringPiI2C.h>
00012
00013 // Structures
00014 typedef struct PresenceData
00015 {
00016         int proximity;
00017         int luminosity;
00018 }presenceD;
00019
00021 // Function Prototypes
00022 int initPresence(void);
00023 int detectPresence(int fd);
00024 int getProximity(int fd);
00025 int getLuminosity(int fd);
00027
00028
00029
00030 #endif
```

7.1.4 Lcd.py File Reference

```
from Adafruit_CharLCD import Adafruit_CharLCD
import Adafruit_GPIO.PCF8574 as PCF
import wiringpi as wiringpi
import time
from time import sleep
import sys


#Initialization of the Wiring Pi Library
wiringpi.wiringPiSetup()
wiringpi.wiringPiSetupGpio()
wiringpi.wiringPiSetupPhys()
wiringpi.pinMode(7, 1)


#Getting the I2C address of the LCD
GPIO = PCF.PCF8574(address=0x38)

#Define PCF pins connected to the LCD
lcd_rs          = 4
lcd_en          = 6
d4,d5,d6,d7     = 0,1,2,3
cols,lines      = 16,2

#instantiate LCD display
lcd = Adafruit_CharLCD(lcd_rs, lcd_en, d4, d5, d6, d7,
        cols, lines, gpio=GPIO)
lcd.clear()

test = ' '.join(sys.argv[1:])

if len(test)>16:
        test = test[0:15] + "\n" + test[15:]

#Display the Contents to the LCD. Turn on Backlight and turn off.
print str(test)
print "LCD ON"
wiringpi.digitalWrite(7,1)
lcd.message(str(test))
sleep(5)
wiringpi.digitalWrite(7,0)
print "LCD OFF"
```

7.2 Application code

Cart Activity

```
public class CartActivity extends AppCompatActivity {
    private RequestQueue mQueue;
    public static final String SHARED_PREFS = Prevalent.CurrentOnlineUser.getStudent_Number();
    private ArrayList<ItemHandler> cart;

    private ListView mCartList;
    private SlidrInterface slidr;

    SimpleAdapter adapter;
    List<HashMap<String,String>> listItems = new ArrayList<>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cart);
        getSupportActionBar().setTitle(R.string.cart);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        Button mButtonRequest = (Button) findViewById(R.id.requestBtn);
        final Button mButtonRemove = (Button) findViewById(R.id.removeBtn);
        final ListView itemCartListView = (ListView) findViewById(R.id.cart_ListView);

        HashMap<String,String>  itemQuanity = new HashMap<>();
        mQueue = Volley.newRequestQueue(this);
        slidr = Slidr.attach(this);
        mCartList = (ListView) findViewById(R.id.cart_ListView);

        loadData();

        for(int i=0;i<cart.size();i++){
            itemQuanity.put(cart.get(i).getName(),"SID: "+cart.get(i).getSid()+"    Quantity: "+cart.get(i).getQuantity());
        }

        Iterator it = itemQuanity.entrySet().iterator();
        while(it.hasNext()){
            HashMap<String,String> resultMap = new HashMap<>();
            Map.Entry pair = (Map.Entry) it.next();
            resultMap.put("Name",pair.getKey().toString());
            resultMap.put("Quantity",pair.getValue().toString());
            listItems.add(resultMap);
        }

        adapter = new SimpleAdapter(this, listItems,R.layout.list_item_cart,
                            new String[]{"Name","Quantity"},
                            new int[]{R.id.text_ItemName,R.id.text_itemQuantity})
        {
            @Override
            public View getView (final int position, View convertView, ViewGroup parent) {
                View v = super.getView(position, convertView, parent);

                Button removeBtn=(Button)v.findViewById(R.id.removeBtn);
                removeBtn.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View view) {

                        AlertDialog.Builder confirm = new AlertDialog.Builder(CartActivity.this);
```

```
                confirm.setTitle(R.string.remove);
                confirm.setMessage(R.string.are_you_sure);
                confirm.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {
                        cart.remove(position);
                        saveData();
                        listItems.remove(position);
                        adapter.notifyDataSetChanged();
                    }
                });
                confirm.setNegativeButton("No", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {
                        Toast.makeText(CartActivity.this,R.string.cancel,Toast.LENGTH_SHORT).show();
                    }
                });
                AlertDialog dialog = confirm.create();
                dialog.show();
            }
        });
        return v;
    }
};

itemCartListView.setAdapter(adapter);


mButtonRequest.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        if (Paper.book().read(Prevalent.CurrentTID).equals(getString(R.string.NoID))) {
            makeRequest();
        }
        else {
            Toast.makeText(CartActivity.this,"You currently have a transaction.",Toast.LENGTH_SHORT).show();
        }
    }
});
}


public void makeRequest(){

    final String url = "http://apollo.humber.ca/~n01156096/CENG319/request.php";

    StringRequest postRequest = new StringRequest(Request.Method.POST, url,
            new Response.Listener<String>()
            {
                @Override
                public void onResponse(String response) {
                    // response
                    Toast.makeText(CartActivity.this,response,Toast.LENGTH_SHORT).show();
                    Paper.book().write(Prevalent.CurrentTID, response);
                        SharedPreferences sharedPreferences =
getSharedPreferences(SHARED_PREFS,MODE_PRIVATE);
                        SharedPreferences.Editor editor = sharedPreferences.edit();
                        editor.remove("cart").commit();
                        goHome();
                }
            },
            new Response.ErrorListener()
            {
```

```
                @Override
                public void onErrorResponse(VolleyError error) {
                    // error
                    Log.d("Error.Response", error.toString());
                }
            }
        ) {
            @Override
            protected Map<String, String> getParams()
            {
                Map<String, String>  params = new HashMap<String, String>();
                String sJson="";
                sJson="{\"uid\":\""+Prevalent.CurrentOnlineUser.getStudent_Number()+"\",\"items\":[";

                int count = 1;
                for(ItemHandler item: cart){
                    sJson+="{\"sid\":"+item.getSid()+",\"quantity\":"+item.getQuantity()+"},";
                }
                sJson = sJson.substring(0,sJson.length()-1)+"]}";
                Log.d("Test",sJson);
                params.put("request", sJson);

                return params;
            }
        };
        mQueue.add(postRequest);

    }


    public void loadData(){
        SharedPreferences sharedPreferences = getSharedPreferences(SHARED_PREFS,MODE_PRIVATE);
        String json = sharedPreferences.getString("cart","");
        if (json.equals("")){
            this.cart = new ArrayList<ItemHandler>();
        }
        else{
            Gson gson = new Gson();
            Type itemCartType = new TypeToken<ArrayList<ItemHandler>>() {}.getType();
            this.cart = gson.fromJson(json,itemCartType);
        }

    }

    public void saveData(){
        SharedPreferences sharedPreferences = getSharedPreferences(SHARED_PREFS,MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPreferences.edit();
        Gson gson = new Gson();
        Type itemCartType = new TypeToken<ArrayList<ItemHandler>>() {}.getType();
        String json = gson.toJson(this.cart,itemCartType);
        editor.putString("cart",json);
        editor.commit();

    }

    public void goHome() {
        Intent intent = new Intent (this, HomeActivity.class);
        startActivity(intent);
    }
}
```

Home Activity

```
public class HomeActivity extends AppCompatActivity implements
NavigationView.OnNavigationItemSelectedListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);
        Toolbar toolbar = findViewById(R.id.toolbar);
        toolbar.setTitle(R.string.menu_home);
        setSupportActionBar(toolbar);
        Paper.init(this);

        DrawerLayout drawer  = findViewById(R.id.drawer_layout);
        NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
        View headerView = navigationView.getHeaderView(0);
        TextView userNameTextView = headerView.findViewById(R.id.user_profile_name);

        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this, drawer, toolbar,
R.string.navigation_drawer_open, R.string.navigation_drawer_close);
        drawer.addDrawerListener(toggle);
        toggle.syncState();

        userNameTextView.setText(Prevalent.CurrentOnlineUser.getStudent_Number());
        Toast.makeText(HomeActivity.this, Prevalent.CurrentOnlineUser.getStudent_Number(),
Toast.LENGTH_SHORT).show();

        navigationView.setNavigationItemSelectedListener(this);



        FloatingActionButton fab = findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(HomeActivity.this,ScanActivity.class);
                startActivity(intent);
            }
        });
    }

    @Override
    public void onBackPressed(){
        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        if(drawer.isDrawerOpen(GravityCompat.START)){
            drawer.closeDrawer(GravityCompat.START);
        }else{
            super.onBackPressed();
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.home, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item){
        int id = item.getItemId();
```

```
        if(id == R.id.nav_settings){
            return true;
        }
        return super.onOptionsItemSelected(item);
    }

    @SuppressWarnings("StatementWithEmptyBoady")
    @Override
    public boolean onNavigationItemSelected(MenuItem item){
        int id = item.getItemId();

        if(id == R.id.nav_orders){
            Intent intent = new Intent(HomeActivity.this,OrderActivity.class);
            startActivity(intent);
        }
        else if (id == R.id.nav_categories) {
            Intent intent = new Intent(HomeActivity.this,ItemMainActivity.class);
            startActivity(intent);

        }
        else if (id == R.id.nav_settings){
            Intent intent = new Intent(HomeActivity.this,SettingsActivity.class);
            startActivity(intent);
        }
        else if (id == R.id.nav_cart){
            Intent intent = new Intent(HomeActivity.this,CartActivity.class);
            startActivity(intent);
        }
        else if (id == R.id.nav_logout){
            Paper.book().destroy();

            Intent intent = new Intent (HomeActivity.this, MainActivity.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
            startActivity(intent);
            finish();

        }
        DrawerLayout drawer = findViewById(R.id.drawer_layout);
        drawer.closeDrawer(GravityCompat.START);
        return true;
    }
}
```

Item Activity

```
public class ItemActivity extends AppCompatActivity{

    public static final String SHARED_PREFS = Prevalent.CurrentOnlineUser.getStudent_Number();;
    private RequestQueue mQueue;
    private String selectedItem;
    private String itemCategory;
    private String itemName;
    private String itemDescription;
    private String itemQuantity;
    private String itemImage;

    private TextView mItemNameTV;
    private TextView mItemCategoryTV;
    private TextView mItemDescriptionTV;
    private TextView mItemQuantityTV;
    private EditText mQuantity;

    private Button mButtonAdd;
    private Button mButtonPlus;
    private Button mButtonMinus;

    private ArrayList<ItemHandler> cart;
    private ItemHandler newItem;
    private SlidrInterface slidr;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_item);
        Intent intent = getIntent();

        getSupportActionBar().setTitle(R.string.item_description);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        selectedItem = intent.getStringExtra("Item");

        mItemNameTV = (TextView) findViewById(R.id.Name_TV);
        mItemCategoryTV = (TextView) findViewById(R.id.Category_TV);
        mItemDescriptionTV = (TextView) findViewById(R.id.Description_TV);
        mItemQuantityTV = (TextView) findViewById(R.id.Quantity_TV);

        mButtonAdd = (Button) findViewById(R.id.addToCartBtn);
        mButtonPlus = (Button) findViewById(R.id.addBtn);
        mButtonMinus = (Button) findViewById(R.id.minusBtn);
        mQuantity = (EditText) findViewById(R.id.quantityField);

        mQueue = Volley.newRequestQueue(this);
        slidr = Slidr.attach(this);

        jsonParse();
        loadData();

        mButtonAdd.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                ItemHandler newItem = new
ItemHandler(itemName,Integer.parseInt(selectedItem),Integer.parseInt(mQuantity.getText().toString()));
                createNewItem(newItem);
            }
```

```
    });

    mButtonPlus.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            int quantity = Integer.parseInt(mQuantity.getText().toString())+1;
            if(quantity<10) {
                mQuantity.setText(Integer.toString(quantity));
            }
        }
    });

    mButtonMinus.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            int quantity = Integer.parseInt(mQuantity.getText().toString());
            if(quantity>1) {
                quantity = quantity - 1;
                mQuantity.setText(Integer.toString(quantity));
            }
        }
    });

    FloatingActionButton fab = findViewById(R.id.fab);
    fab.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(ItemActivity.this,CartActivity.class);
            startActivity(intent);
        }
    });
}

private void jsonParse(){

    String url = "http://apollo.humber.ca/~n01156096/CENG319/query.php?item="+selectedItem;

    JsonObjectRequest request =  new JsonObjectRequest(Request.Method.GET, url, null,
            new Response.Listener<JSONObject>() {
                @Override
                public void onResponse(JSONObject response) {
                    try {
                        itemCategory = response.getString("Category");
                        itemName = response.getString("Name");
                        itemDescription = response.getString("Description");
                        itemQuantity = response.getString("Quantity");
                        itemImage = response.getString("Image");
                        Toast.makeText(ItemActivity.this,itemImage,Toast.LENGTH_SHORT).show();

                        updateTextView();
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                }
            }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            error.printStackTrace();
        }
    });
    mQueue.add(request);
}
```

```
private void updateTextView(){
    mItemNameTV.setText("Name: "+itemName);
    mItemCategoryTV.setText("Category: "+itemCategory);
    mItemDescriptionTV.setText("Description: "+itemDescription);
    mItemQuantityTV.setText("Quantity: "+itemQuantity);
}

public void loadData(){
    SharedPreferences sharedPreferences = getSharedPreferences(SHARED_PREFS,MODE_PRIVATE);
    String json = sharedPreferences.getString("cart","");
    if (json.equals("")){
        this.cart = new ArrayList<ItemHandler>();
    }
    else{
        Gson gson = new Gson();
        Type itemCartType = new TypeToken<ArrayList<ItemHandler>>() {}.getType();
        this.cart = gson.fromJson(json,itemCartType);
    }

}
public void saveData(){
    SharedPreferences sharedPreferences = getSharedPreferences(SHARED_PREFS,MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    Gson gson = new Gson();
    Type itemCartType = new TypeToken<ArrayList<ItemHandler>>() {}.getType();
    String json = gson.toJson(this.cart,itemCartType);
    editor.putString("cart",json);
    editor.commit();
}

public void createNewItem(final ItemHandler item){
    if (cart.isEmpty()) {
        this.cart.add(item);
        saveData();
        Toast.makeText(ItemActivity.this,R.string.add_yes,Toast.LENGTH_SHORT).show();
    } else {
        for (int i=0; i< cart.size(); i++) {
            if (cart.get(i).getSid() == item.getSid()) {

                final int cartSpot = i;

                AlertDialog.Builder confirm = new AlertDialog.Builder(ItemActivity.this);
                confirm.setTitle(R.string.order_exists);
                confirm.setMessage(R.string.add_prompt);
                confirm.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {

                        int initQuantity = cart.get(cartSpot).getQuantity();
                        int addThis = item.getQuantity();
                        cart.get(cartSpot).setQuantity(initQuantity + addThis);
                        saveData();
                        Toast.makeText(ItemActivity.this,R.string.add_yes,Toast.LENGTH_SHORT).show();

                    }
                });
                confirm.setNegativeButton("No", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {
                        Toast.makeText(ItemActivity.this,R.string.cancel,Toast.LENGTH_SHORT).show();
                    }
```

```
                });
                AlertDialog dialog = confirm.create();
                dialog.show();

                return;
            }
        }
            this.cart.add(item);
            saveData();
            Toast.makeText(ItemActivity.this,R.string.add_yes,Toast.LENGTH_SHORT).show();
    }
  }
}
```

Item Handler

```
public class ItemHandler {
    private String name;
    private int sid;
    private int quantity;

    public ItemHandler(String name, int sid, int quantity){
        this.name=name;
        this.sid=sid;
        this.quantity=quantity;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getSid() {
        return sid;
    }

    public void setSid(int sid) {
        this.sid = sid;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    @Override
    public String toString() {
        return "ItemHandler{" +
                "name='" + name + '\'' +
                ", sid=" + sid +
                ", quantity=" + quantity +
                '}';
    }
}
```

Item List Activity

```
public class ItemListActivity extends AppCompatActivity{
    private RequestQueue mQueue;
    private ArrayList<String> itemList = new ArrayList<>();
    private ArrayList<String> sidList = new ArrayList<>();
    private ListView mListItems;
    private ArrayAdapter arrayAdapter;
    private String selectedCat;
    private SlidrInterface slidr;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_item_list);
        Intent intent = getIntent();
        getSupportActionBar().setTitle(R.string.item_name);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        selectedCat = intent.getStringExtra("Category");
        mListItems = (ListView)findViewById(R.id.listViewItems);
        slidr = Slidr.attach(this);
        mQueue = Volley.newRequestQueue(this);

        jsonParse();
        arrayAdapter = new ArrayAdapter(this,android.R.layout.simple_list_item_1);

        mListItems.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
                String selectedItem = sidList.get(i);
                Intent newIntent = new Intent(ItemListActivity.this,ItemActivity.class);
                newIntent.putExtra("Item",selectedItem);
                ItemListActivity.this.startActivity(newIntent);
            }
        });

        Toast.makeText(ItemListActivity.this,"Going to: "+ selectedCat,Toast.LENGTH_SHORT).show();
    }

    private void jsonParse(){

        String url = "http://apollo.humber.ca/~n01156096/CENG319/query.php?category="+selectedCat;

        JsonObjectRequest request =  new JsonObjectRequest(Request.Method.GET, url, null,
                new Response.Listener<JSONObject>() {
                    @Override
                    public void onResponse(JSONObject response) {
                        try {
                            JSONArray jsonArray = response.getJSONArray("Items");
                            for (int i = 0; i< jsonArray.length(); i++){
                                //Append to array list to create list view
                                itemList.add(String.valueOf(jsonArray.get(i)));

                                mListItems.setAdapter(arrayAdapter);
                                arrayAdapter.clear();
                                arrayAdapter.addAll(itemList);
                                arrayAdapter.notifyDataSetChanged();


                            }
```

```
                    JSONArray jsonArray2 = response.getJSONArray("SID");
                    for (int i = 0; i< jsonArray.length(); i++){
                        sidList.add(String.valueOf(jsonArray2.get(i)));
                    }
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            error.printStackTrace();
        }
    });
    mQueue.add(request);

}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu, menu);

    MenuItem searchItem = menu.findItem(R.id.item_search);
    SearchView searchView = (SearchView) MenuItemCompat.getActionView(searchItem);

    searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
        @Override
        public boolean onQueryTextSubmit(String query) {

            return false;
        }

        @Override
        public boolean onQueryTextChange(String newText) {
            ArrayList<String> usersList = new ArrayList<>();

            for (String user : itemList){
                if (user.toLowerCase().contains(newText.toLowerCase())){
                    usersList.add(user);
                }
            }
            ArrayAdapter<String> adapter = new
ArrayAdapter<String>(ItemListActivity.this,android.R.layout.simple_list_item_1, usersList);
            mListItems.setAdapter(adapter);


            return true;
        }
    });


    return super.onCreateOptionsMenu(menu);
}
}
```

Item Main Activity

```
public class ItemMainActivity extends AppCompatActivity{

    private RequestQueue mQueue;
    private ArrayList <String> categoryList = new ArrayList<>();
    private ListView mListCategories;
    private ArrayAdapter arrayAdapter;
    private SlidrInterface slidr;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_item_main);

        getSupportActionBar().setTitle(R.string.item_category);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        mListCategories = (ListView)findViewById(R.id.listViewCategories);
        mQueue = Volley.newRequestQueue(this);
        slidr = Slidr.attach(this);

        jsonParse();
        arrayAdapter = new ArrayAdapter(this,android.R.layout.simple_list_item_1);

        mListCategories.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {

                String selectedCat = categoryList.get(i);
                Intent newIntent = new Intent(ItemMainActivity.this, ItemListActivity.class);
                newIntent.putExtra("Category",selectedCat);
                ItemMainActivity.this.startActivity(newIntent);
            }
        });
    }

    private void jsonParse(){

        String url = "http://apollo.humber.ca/~n01156096/CENG319/query.php";
        JsonObjectRequest request =  new JsonObjectRequest(Request.Method.GET, url, null,
                new Response.Listener<JSONObject>() {
                    @Override
                    public void onResponse(JSONObject response) {
                        try {
                            JSONArray jsonArray = response.getJSONArray("Category");
                            for (int i = 0; i< jsonArray.length(); i++){
                                //Append to array list to create list view
                                categoryList.add(String.valueOf(jsonArray.get(i)));
                                mListCategories.setAdapter(arrayAdapter);
                                arrayAdapter.clear();
                                arrayAdapter.addAll(categoryList);
                                arrayAdapter.notifyDataSetChanged();
                            }
                        } catch (JSONException e) {
                            e.printStackTrace();
                        }
                    }
                }, new Response.ErrorListener() {
            @Override
```

```
            public void onErrorResponse(VolleyError error) {
                error.printStackTrace();
            }
        });
        mQueue.add(request);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menu, menu);

        MenuItem searchItem = menu.findItem(R.id.item_search);
        SearchView searchView = (SearchView) MenuItemCompat.getActionView(searchItem);

        searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String query) {

                return false;
            }

            @Override
            public boolean onQueryTextChange(String newText) {
                ArrayList<String> usersList = new ArrayList<>();

                for (String user : categoryList){
                    if (user.toLowerCase().contains(newText.toLowerCase())){
                        usersList.add(user);
                    }
                }
                ArrayAdapter<String> adapter = new
ArrayAdapter<String>(ItemMainActivity.this,android.R.layout.simple_list_item_1, usersList);
                mListCategories.setAdapter(adapter);


                return true;
            }
        });


        return super.onCreateOptionsMenu(menu);
    }
}
```

JSON Parser

```
public class JSONParser {

    static InputStream is = null;
    static JSONObject jObj = null;
    static JSONArray jArr = null;
    static String json = "";
    static String error = "";

    // constructor
    public JSONParser() {

    }

    // function get json from url
    // by making HTTP POST or GET mehtod
    public JSONObject makeHttpRequest(String url, String method,
                                      ArrayList params) {

        // Making HTTP request
        try {

            // check for request method
            if(method.equals("POST")){
                // request method is POST
                // defaultHttpClient
                HttpClient httpClient = new DefaultHttpClient();
                HttpPost httpPost = new HttpPost(url);
                httpPost.setEntity(new UrlEncodedFormEntity(params));
                try {
                    Log.e("API123", " " +convertStreamToString(httpPost.getEntity().getContent()));
                    Log.e("API123",httpPost.getURI().toString());
                } catch (Exception e) {
                    e.printStackTrace();
                }

                HttpResponse httpResponse = httpClient.execute(httpPost);
                Log.e("API123",""+httpResponse.getStatusLine().getStatusCode());
                error= String.valueOf(httpResponse.getStatusLine().getStatusCode());
                HttpEntity httpEntity = httpResponse.getEntity();
                is = httpEntity.getContent();

            }else if(method.equals("GET")){
                // request method is GET
                DefaultHttpClient httpClient = new DefaultHttpClient();
                String paramString = URLEncodedUtils.format(params, "utf-8");
                url += "?" + paramString;
                HttpGet httpGet = new HttpGet(url);

                HttpResponse httpResponse = httpClient.execute(httpGet);
                HttpEntity httpEntity = httpResponse.getEntity();
                is = httpEntity.getContent();
            }

        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        } catch (ClientProtocolException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
```

```
        }

        try {
            BufferedReader reader = new BufferedReader(new InputStreamReader(
                    is, "iso-8859-1"), 8);
            StringBuilder sb = new StringBuilder();
            String line = null;
            while ((line = reader.readLine()) != null) {
                sb.append(line + "\n");
            }
            is.close();
            json = sb.toString();
            Log.d("API123",json);
        } catch (Exception e) {
            Log.e("Buffer Error", "Error converting result " + e.toString());
        }

        // try to parse the string to a JSON object
        try {
            jObj = new JSONObject(json);
            jObj.put("error_code",error);
        } catch (JSONException e) {
            Log.e("JSON Parser", "Error parsing data " + e.toString());
        }

        // return JSON String
        return jObj;

    }

    private String convertStreamToString(InputStream is) throws Exception {
        BufferedReader reader = new BufferedReader(new InputStreamReader(is));
        StringBuilder sb = new StringBuilder();
        String line = null;
        while ((line = reader.readLine()) != null) {
            sb.append(line);
        }
        is.close();
        return sb.toString();
    }
}
```

Login Activity

```
public class LoginActivity extends AppCompatActivity {

    private EditText InputStudentNumber, InputPassword;
    private Button LoginButton;
    private ProgressDialog loadingBar;
    private CheckBox chkBoxRemeberMe;

    final private int REQUEST_CODE_ASK_PERMISSIONS = 200;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        getSupportActionBar().hide();

        LoginButton = (Button) findViewById(R.id.login_btn);
        InputStudentNumber = (EditText) findViewById(R.id.login_student_number_input);
        InputPassword = (EditText) findViewById(R.id.login_password_input);
        loadingBar = new ProgressDialog(this);

        chkBoxRemeberMe = (CheckBox) findViewById(R.id.remember_me_chkb);
        Paper.init(this);

        LoginButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (ContextCompat.checkSelfPermission(LoginActivity.this, Manifest.permission.custom) ==
PackageManager.PERMISSION_GRANTED) {
                    LoginUser();
                } else { ActivityCompat.requestPermissions(LoginActivity.this, new String[] {Manifest.permission.custom},
REQUEST_CODE_ASK_PERMISSIONS); }
            }
        });
    }

    @Override
    protected void onStop() {
        loadingBar.dismiss();
        super.onStop();
    }

    public void LoginUser() {
        String studentNumber = InputStudentNumber.getText().toString();
        String password = InputPassword.getText().toString();

        if(TextUtils.isEmpty(studentNumber)){
            Toast.makeText(this, R.string.enter_s_num, Toast.LENGTH_SHORT).show();
        }
        else if(TextUtils.isEmpty(password)){
            Toast.makeText(this, R.string.enter_pass, Toast.LENGTH_SHORT).show();
        }
        else{
            if(chkBoxRemeberMe.isChecked()){
                Paper.book().write(Prevalent.UserStudentNumberKey, studentNumber);
                Paper.book().write(Prevalent.UserPasswordKey, password);
            } else {
                Paper.book().destroy();
            }
```

```
        loadingBar.setTitle(R.string.logging);
        loadingBar.setMessage("Please wait");
        loadingBar.setCanceledOnTouchOutside(false);
        loadingBar.show();

        AttemptLogin attemptLogin = new AttemptLogin();
        attemptLogin.execute(studentNumber,password);

    }
}

private class AttemptLogin extends AsyncTask<String, String, JSONObject> {

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }

    @Override
    protected JSONObject doInBackground(String... args) {

        String name = args[0];
        String password = args[1];
        String url = "http://apollo.humber.ca/~n01156096/CENG319/account.php";
        JSONParser jsonParser = new JSONParser();

        ArrayList params = new ArrayList();
        params.add(new BasicNameValuePair("username",name));
        params.add(new BasicNameValuePair("password",password));

        JSONObject json = jsonParser.makeHttpRequest(url, "POST", params);
        return json;
    }

    protected void onPostExecute(JSONObject result) {
        try {
            if(result!=null) {
                if(result.getInt("success")==1) {
                    //CurrentOnlineUser data is saved here, alter if necessary
                    Users userData = new Users(result.getString("id"),InputPassword.getText().toString(),
                            result.getString("name"),result.getString("uid"),result.getString("email"));
                    Prevalent.CurrentOnlineUser = userData;

                    //Function call to go to HomeActivity
                    success();
                } else {
                    Toast.makeText(LoginActivity.this, result.getString("message"), Toast.LENGTH_SHORT).show();
                    loadingBar.dismiss();
                }
            } else {
                Toast.makeText(getApplicationContext(),R.string.db_error, Toast.LENGTH_SHORT).show();
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}

public void success() {
    Intent intent = new Intent (this, HomeActivity.class);
    startActivity(intent);
}
```

```
}
```

Main Activity

```
public class MainActivity extends AppCompatActivity {

    private Button joinNowButton, loginButton;
    private ProgressDialog loadingBar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ProcessPreference();
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        getSupportActionBar().hide();

        joinNowButton = (Button) findViewById(R.id.main_join_now_btn);
        loginButton = (Button) findViewById(R.id.main_login_btn);
        loadingBar = new ProgressDialog(this);

        Paper.init(this);

        String UserStudentNumberKey  = Paper.book().read(Prevalent.UserStudentNumberKey);
        String UserPasswordKey  = Paper.book().read(Prevalent.UserPasswordKey);

        loginButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick (View view){
                Intent intent = new Intent(MainActivity.this,LoginActivity.class);
                startActivity(intent);
            }
        });

        joinNowButton.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View view)
            {
                Intent intent = new Intent(MainActivity.this,RegisterActivity.class);
                startActivity(intent);
            }

        });

        if(UserStudentNumberKey != "" && UserPasswordKey != ""){
            if(!TextUtils.isEmpty(UserStudentNumberKey) && !TextUtils.isEmpty(UserPasswordKey)){
                loadingBar.setTitle(R.string.already_logged);
                loadingBar.setMessage("Please Wait");
                loadingBar.setCanceledOnTouchOutside(false);
                loadingBar.show();

                AttemptResignin attemptLogin = new AttemptResignin();
                attemptLogin.execute(UserStudentNumberKey,UserPasswordKey);
            }
        }
    }

    protected void onStop() {
        loadingBar.dismiss();
        super.onStop();
    }
```

```
private void ProcessPreference() {
    SharedPreferences settings = getSharedPreferences("SettingsActivity",MODE_PRIVATE);
    Boolean language = settings.getBoolean("language",false);

    if (language) {
        setAppLocale("fr");
    } else {
        setAppLocale("en");
    }
}

private void setAppLocale(String localeCode) {
    Resources resources = getResources();
    DisplayMetrics dm = resources.getDisplayMetrics();
    Configuration config = resources.getConfiguration();

    if (Build.VERSION.SDK_INT>=Build.VERSION_CODES.JELLY_BEAN_MR1) {
        config.setLocale(new Locale(localeCode.toLowerCase()));
    } else {
        config.locale = new Locale(localeCode.toLowerCase());
    }
    resources.updateConfiguration(config,dm);
}

private class AttemptResignin extends AsyncTask<String, String, JSONObject> {

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }

    @Override
    protected JSONObject doInBackground(String... args) {

        String name = args[0];
        String password = args[1];
        String url = "http://apollo.humber.ca/~n01267335/CENG319/account.php";
        JSONParser jsonParser = new JSONParser();

        ArrayList params = new ArrayList();
        params.add(new BasicNameValuePair("username",name));
        params.add(new BasicNameValuePair("password",password));

        JSONObject json = jsonParser.makeHttpRequest(url, "POST", params);
        return json;
    }

    protected void onPostExecute(JSONObject result) {
        try {
            if(result!=null) {
                //Pass string to HomeActivity page
                if(result.getInt("success")==1) {
                    Users userData = new Users(result.getString("id"),UserPasswordKey,
                            result.getString("name"),result.getString("uid"),result.getString("email"));
                    Prevalent.CurrentOnlineUser = userData;
                    success();
                } else {
                    Toast.makeText(MainActivity.this,result.getString("message"), Toast.LENGTH_SHORT).show();
                    loadingBar.dismiss();
                }
            } else {
```

```
                Toast.makeText(getApplicationContext(),R.string.db_error, Toast.LENGTH_SHORT).show();
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}

public void success() {
    Intent intent = new Intent (this, HomeActivity.class);
    startActivity(intent);
}


}
```

Order Activity

```
public class OrderActivity extends AppCompatActivity {

    private RequestQueue mQueue;
    private ArrayList <String> orderList = new ArrayList<>();
    private ArrayList<String> tidList = new ArrayList<>();
    private ListView mListCategories;
    private ArrayAdapter arrayAdapter;
    private String CurrentUser  = Prevalent.CurrentOnlineUser.getStudent_Number();
    private SlidrInterface slidr;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_order);

        getSupportActionBar().setTitle(R.string.orders);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        mListCategories = (ListView)findViewById(R.id.order_ListView);

        mQueue = Volley.newRequestQueue(this);
        slidr = Slidr.attach(this);
        jsonParse();
        arrayAdapter = new ArrayAdapter(this,android.R.layout.simple_list_item_1);

        mListCategories.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {

                String selectedOrder = tidList.get(i);
                Intent newIntent = new Intent(OrderActivity.this, OrderDetailActivity.class);
                newIntent.putExtra("Order",selectedOrder);
                startActivity(newIntent);
            }
        });
    }

    private void jsonParse(){

        String url = "http://apollo.humber.ca/~n01156096/CENG319/order.php?username=" + CurrentUser;
        JsonObjectRequest request =  new JsonObjectRequest(Request.Method.GET, url, null,
            new Response.Listener<JSONObject>() {
                @Override
                public void onResponse(JSONObject response) {
                    try {
                        JSONArray jsonArray = response.getJSONArray("TOut");
                        for (int i = 0; i< jsonArray.length(); i++){
                            //Append to array list to create list view
                            orderList.add(String.valueOf(jsonArray.get(i)));
                            mListCategories.setAdapter(arrayAdapter);
                            arrayAdapter.clear();
                            arrayAdapter.addAll(orderList);
                            arrayAdapter.notifyDataSetChanged();
                        }

                        JSONArray jsonArray2 = response.getJSONArray("TID");
                        for (int i = 0; i< jsonArray.length(); i++){
                            tidList.add(String.valueOf(jsonArray2.get(i)));
                        }
```

```
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            error.printStackTrace();
        }
    });
    mQueue.add(request);
  }
}
```

Order Details Activity

```
public class OrderDetailActivity extends AppCompatActivity {

    private String selectedOrder;
    private String CurrentUser  = Prevalent.CurrentOnlineUser.getStudent_Number();
    private RequestQueue mQueue;

    private ArrayList<String> orderList = new ArrayList<>();

    private ListView mOrderDetail;
    private ArrayAdapter arrayAdapter;
    private SlidrInterface slidr;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_order_detail);

        Intent intent = getIntent();

        getSupportActionBar().setTitle(R.string.orders_details);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        selectedOrder = intent.getStringExtra("Order");
        mOrderDetail = (ListView)findViewById(R.id.order_ListView);
        mQueue = Volley.newRequestQueue(this);
        slidr = Slidr.attach(this);
        jsonParse();

        arrayAdapter = new ArrayAdapter(this,android.R.layout.simple_list_item_1);
    }

    private void jsonParse(){

        String url = "http://apollo.humber.ca/~n01156096/CENG319/order.php?TID=" + selectedOrder + "&username="
+ CurrentUser;
        JsonObjectRequest request =  new JsonObjectRequest(Request.Method.GET, url, null,
                new Response.Listener<JSONObject>() {
                    @Override
                    public void onResponse(JSONObject response) {
                        try {
                            JSONArray jsonArray = response.getJSONArray("Category");
                            JSONArray jsonArray2 = response.getJSONArray("Name");
                            JSONArray jsonArray3 = response.getJSONArray("SID");
                            JSONArray jsonArray4 = response.getJSONArray("Quantity");
                            for (int i = 0; i< jsonArray.length(); i++){
                                //Append to array list to create list view
                                orderList.add(jsonArray.get(i)+": "+jsonArray2.get(i)+"\nSID:"+jsonArray3.get(i)+"
Quantity:"+jsonArray4.get(i));
                                mOrderDetail.setAdapter(arrayAdapter);
                                arrayAdapter.clear();
                                arrayAdapter.addAll(orderList);
                                arrayAdapter.notifyDataSetChanged();
                            }
                        } catch (JSONException e) {
                            e.printStackTrace();
                        }
                    }
                }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
```

```
                error.printStackTrace();
            }
        });
        mQueue.add(request);
    }
}
```

Register Activity

```
public class RegisterActivity extends AppCompatActivity {

    private Button CreateAccountButton;
    private EditText InputFName, InputLName, InputStudentNumber, InputPassword, InputConfirmPassword,
InputEmail;
    private ProgressDialog loadingBar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
        getSupportActionBar().hide();

        CreateAccountButton = (Button) findViewById(R.id.register_btn);
        InputFName = (EditText) findViewById(R.id.register_FName);
        InputLName = (EditText) findViewById(R.id.register_LName);
        InputStudentNumber = (EditText) findViewById(R.id.register_student_number_input);
        InputPassword = (EditText) findViewById(R.id.register_password_input);
        InputConfirmPassword = (EditText) findViewById(R.id.register_confirm_password_input);
        InputEmail = (EditText) findViewById(R.id.register_email_input);
        loadingBar = new ProgressDialog(this);

        CreateAccountButton.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View view){
                CreateAccount();
            }
        });
    }

    private void CreateAccount() {
        String FName = InputFName.getText().toString();
        String LName = InputLName.getText().toString();
        String StudentNumber = InputStudentNumber.getText().toString();
        String Password = InputPassword.getText().toString();
        String ConfirmPassword = InputConfirmPassword.getText().toString();
        String Email = InputEmail.getText().toString();

        if(TextUtils.isEmpty(FName)){
            Toast.makeText(this, "Please Enter your first Name", Toast.LENGTH_SHORT).show();
        }
        else if(TextUtils.isEmpty(LName)){
            Toast.makeText(this, "Please Enter your Last Name", Toast.LENGTH_SHORT).show();
        }
        else if(TextUtils.isEmpty(StudentNumber)){
            Toast.makeText(this, R.string.enter_s_num, Toast.LENGTH_SHORT).show();
        }
        else if(TextUtils.isEmpty(Password)){
            Toast.makeText(this, R.string.enter_pass, Toast.LENGTH_SHORT).show();
        }
       else if(TextUtils.isEmpty(Email)){
            Toast.makeText(this, "Please Enter a Email", Toast.LENGTH_SHORT).show();
        }
       else if(Password.compareTo (ConfirmPassword) == 0){
            loadingBar.setTitle("Creating Account");
            loadingBar.setMessage("Please Wait");
            loadingBar.setCanceledOnTouchOutside(false);
            loadingBar.show();
```

```
            ValidateStudentNumber(FName, LName,  StudentNumber, Email, Password);
        }
       else{
            Toast.makeText(this,"Passwords do not match", Toast.LENGTH_SHORT).show();
        }
    }

    private void ValidateStudentNumber(String FName, String LName, String StudentNumber, String Email, String
Password) {
        RegisterAccount register = new RegisterAccount();
        register.execute(StudentNumber,Password,FName,LName,Email);
    }

    private class RegisterAccount extends AsyncTask<String, String, JSONObject> {
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }

        @Override
        protected JSONObject doInBackground(String... args) {

            String id = args[0];
            String password = args[1];
            String name = args[2];
            String last = args[3];
            String email = args[4];
            String url = "http://apollo.humber.ca/~n01267335/CENG319/account.php";

            JSONParser jsonParser = new JSONParser();
            ArrayList params = new ArrayList();
            params.add(new BasicNameValuePair("username",id));
            params.add(new BasicNameValuePair("password",password));
            params.add(new BasicNameValuePair("name",name));
            params.add(new BasicNameValuePair("last",last));
            params.add(new BasicNameValuePair("email",email));


            JSONObject json = jsonParser.makeHttpRequest(url, "POST", params);
            return json;
        }

        protected void onPostExecute(JSONObject result) {
            try {
                if(result!=null) {
                    if(result.getInt("success")==1) {
                        loadingBar.dismiss();
                        Intent intent = new Intent(RegisterActivity.this,LoginActivity.class);
                        startActivity(intent);
                        Toast.makeText(RegisterActivity.this, result.getString("message"), Toast.LENGTH_SHORT).show();
                    } else {
                        loadingBar.dismiss();
                        Toast.makeText(RegisterActivity.this, result.getString("message"), Toast.LENGTH_SHORT).show();
                    }
                } else {
                    Toast.makeText(getApplicationContext(), R.string.db_error, Toast.LENGTH_SHORT).show();
                }
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    }
```

```
}
```

Settings Activity

```
public class SettingsActivity extends AppCompatActivity {

    private Button confirm_btn;
    private Switch language_sw;
    private EditText currentPass, newPass, confirmNewPass;
    private CheckBox checkBox;
    private SlidrInterface slidr;
    SharedPreferences sharedPref;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_settings);

        getSupportActionBar().setTitle(R.string.settings);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        confirm_btn = (Button) findViewById(R.id.confirm_btn);
        currentPass = (EditText) findViewById(R.id.CurrentPassword);
        newPass = (EditText) findViewById(R.id.NewPassword);
        confirmNewPass = (EditText) findViewById(R.id.ConfirmNewPassword);
        checkBox = (CheckBox) findViewById(R.id.checkBox);
        language_sw = (Switch) findViewById(R.id.language_sw);
        sharedPref = this.getPreferences(Context.MODE_PRIVATE);

        slidr = Slidr.attach(this);
        LoadPreference();

        confirm_btn.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View view){
                validate();
            }
        });

        checkBox.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton compoundButton, boolean isChecked) {
                showPass(isChecked);
            }
        });

        language_sw.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (language_sw.isChecked()) {
                    setAppLocale("fr");
                    saveSharedPreference();
                } else if (!language_sw.isChecked()){
                    setAppLocale("en");
                    saveSharedPreference();
                }
                finish();
                startActivity(getIntent());
            }
        });
//        Toast.makeText(this, Prevalent.CurrentOnlineUser.getEmail(), Toast.LENGTH_SHORT).show();
    }
```

```
@Override
public void onBackPressed() {
    saveSharedPreference();
    super.onBackPressed();
}

private void setAppLocale(String localeCode) {
    Resources resources = getResources();
    DisplayMetrics dm = resources.getDisplayMetrics();
    Configuration config = resources.getConfiguration();

    if (Build.VERSION.SDK_INT>=Build.VERSION_CODES.JELLY_BEAN_MR1) {
        config.setLocale(new Locale(localeCode.toLowerCase()));
    } else {
        config.locale = new Locale(localeCode.toLowerCase());
    }
    resources.updateConfiguration(config,dm);
}

private void saveSharedPreference() {
    SharedPreferences.Editor editor = sharedPref.edit();
    editor.putBoolean("language",language_sw.isChecked());
    editor.commit();
}

private void LoadPreference() {
    Boolean checked = sharedPref.getBoolean("language",false);
    language_sw.setChecked(checked);
}

private void validate() {
    String currentPassword = currentPass.getText().toString();
    String newPassword = newPass.getText().toString();
    String confirmNewPassword = confirmNewPass.getText().toString();

    if(TextUtils.isEmpty(currentPassword)){
        Toast.makeText(this, "Please Enter your current password", Toast.LENGTH_SHORT).show();
    }
    else if(TextUtils.isEmpty(newPassword)){
        Toast.makeText(this, "Please Enter a newPassword", Toast.LENGTH_SHORT).show();
    }

    else if(newPassword.compareTo (confirmNewPassword) == 0){
        //New Password added
    }
    else{
        Toast.makeText(this, "New passwords do not match", Toast.LENGTH_SHORT).show();
    }

    ChangePassword changePass = new ChangePassword();
    changePass.execute(Prevalent.CurrentOnlineUser.getEmail(),currentPassword,newPassword);
}

private void showPass(boolean isChecked) {
    if(isChecked) {
        currentPass.setTransformationMethod(HideReturnsTransformationMethod.getInstance());
        newPass.setTransformationMethod(HideReturnsTransformationMethod.getInstance());
        confirmNewPass.setTransformationMethod(HideReturnsTransformationMethod.getInstance());
    } else {
        currentPass.setTransformationMethod(PasswordTransformationMethod.getInstance());
        newPass.setTransformationMethod(PasswordTransformationMethod.getInstance());
        confirmNewPass.setTransformationMethod(PasswordTransformationMethod.getInstance());
```

```
        }
    }

    private class ChangePassword extends AsyncTask<String, String, JSONObject> {

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }

        @Override
        protected JSONObject doInBackground(String... args) {

            String email = args[0];
            String password = args[1];
            String passwordNew = args[2];
//            String url = "http://apollo.humber.ca/~n01156096/CENG319/accCon4.php";
            String url = "http://apollo.humber.ca/~n01156096/CENG319/account.php";

            JSONParser jsonParser = new JSONParser();
            ArrayList params = new ArrayList();
            params.add(new BasicNameValuePair("email", email));
            params.add(new BasicNameValuePair("password", password));
            params.add(new BasicNameValuePair("passwordNew", passwordNew));

            JSONObject json = jsonParser.makeHttpRequest(url, "POST", params);
            return json;
        }

        protected void onPostExecute(JSONObject result) {
            try {
                if (result != null) {
                    if (result.getInt("success") == 1) {
                        Toast.makeText(SettingsActivity.this, result.getString("message"), Toast.LENGTH_SHORT).show();
                    }
                } else {
                    Toast.makeText(getApplicationContext(), "Unable to retrieve data from server",
Toast.LENGTH_SHORT).show();
                }
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Splash Activity

```
public class SplashActivity extends AppCompatActivity {
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
        finish();
    }

}
```

Credits to Justin Riberio for the following.

myHostApduService

```
/**
 * Created by justin.ribeiro on 10/27/2014.
 *
 * The following definitions are based on two things:
 *   1. NFC Forum Type 4 Tag Operation Technical Specification, version 3.0 2014-07-30
 *   2. APDU example in libnfc: http://nfc-tools.org/index.php?title=Libnfc:APDU_example
 *
 */
public class myHostApduService extends HostApduService {

    private static final String TAG = "JDR HostApduService";

    //
    // We use the default AID from the HCE Android documentation
    // https://developer.android.com/guide/topics/connectivity/nfc/hce.html
    //
    // Ala... <aid-filter android:name="F0394148148100" />
    //
    private static final byte[] APDU_SELECT = {
        (byte)0x00, // CLA      - Class - Class of instruction
        (byte)0xA4, // INS      - Instruction - Instruction code
        (byte)0x04, // P1       - Parameter 1 - Instruction parameter 1
        (byte)0x00, // P2       - Parameter 2 - Instruction parameter 2
        (byte)0x07, // Lc field  - Number of bytes present in the data field of the command
        (byte)0xF0, (byte)0x39, (byte)0x41, (byte)0x48, (byte)0x14, (byte)0x81, (byte)0x00, // NDEF Tag Application
name
        (byte)0x00  // Le field  - Maximum number of bytes expected in the data field of the response to the command
    };

    private static final byte[] CAPABILITY_CONTAINER = {
            (byte)0x00, // CLA  - Class - Class of instruction
            (byte)0xa4, // INS   - Instruction - Instruction code
            (byte)0x00, // P1    - Parameter 1 - Instruction parameter 1
            (byte)0x0c, // P2    - Parameter 2 - Instruction parameter 2
            (byte)0x02, // Lc field           - Number of bytes present in the data field of the command
            (byte)0xe1, (byte)0x03 // file identifier of the CC file
    };
```

```
private static final byte[] READ_CAPABILITY_CONTAINER = {
        (byte)0x00, // CLA  - Class - Class of instruction
        (byte)0xb0, // INS   - Instruction - Instruction code
        (byte)0x00, // P1    - Parameter 1 - Instruction parameter 1
        (byte)0x00, // P2    - Parameter 2 - Instruction parameter 2
        (byte)0x0f  // Lc field            - Number of bytes present in the data field of the command
};

// In the scenario that we have done a CC read, the same byte[] match
// for ReadBinary would trigger and we don't want that in succession
private boolean READ_CAPABILITY_CONTAINER_CHECK = false;

private static final byte[] READ_CAPABILITY_CONTAINER_RESPONSE = {
        (byte)0x00, (byte)0x0F, // CCLEN length of the CC file
        (byte)0x20, // Mapping Version 2.0
        (byte)0x00, (byte)0x3B, // MLe maximum 59 bytes R-APDU data size
        (byte)0x00, (byte)0x34, // MLc maximum 52 bytes C-APDU data size
        (byte)0x04, // T field of the NDEF File Control TLV
        (byte)0x06, // L field of the NDEF File Control TLV
        (byte)0xE1, (byte)0x04, // File Identifier of NDEF file
        (byte)0x00, (byte)0x32, // Maximum NDEF file size of 50 bytes
        (byte)0x00, // Read access without any security
        (byte)0x00, // Write access without any security
        (byte)0x90, (byte)0x00 // A_OKAY
};

private static final byte[] NDEF_SELECT = {
        (byte)0x00, // CLA  - Class - Class of instruction
        (byte)0xa4, // Instruction byte (INS) for Select command
        (byte)0x00, // Parameter byte (P1), select by identifier
        (byte)0x0c, // Parameter byte (P1), select by identifier
        (byte)0x02, // Lc field            - Number of bytes present in the data field of the command
        (byte)0xE1, (byte)0x04 // file identifier of the NDEF file retrieved from the CC file
};

private static final byte[] NDEF_READ_BINARY_NLEN = {
        (byte)0x00, // Class byte (CLA)
        (byte)0xb0, // Instruction byte (INS) for ReadBinary command
        (byte)0x00, (byte)0x00, // Parameter byte (P1, P2), offset inside the CC file
        (byte)0x02  // Le field
};

private static final byte[] NDEF_READ_BINARY_GET_NDEF = {
        (byte)0x00, // Class byte (CLA)
        (byte)0xb0, // Instruction byte (INS) for ReadBinary command
        (byte)0x00, (byte)0x00, // Parameter byte (P1, P2), offset inside the CC file
        (byte)0x0f  //  Le field
};

private static final byte[] A_OKAY = {
        (byte)0x90,  // SW1 Status byte 1 - Command processing status
        (byte)0x00   // SW2 Status byte 2 - Command processing qualifier
};

private static final byte[] NDEF_ID = {
        (byte)0xE1,
        (byte)0x04
};

private NdefRecord NDEF_URI = new NdefRecord(
        NdefRecord.TNF_WELL_KNOWN,
```

```
        NdefRecord.RTD_TEXT,
        NDEF_ID,
        "Hello world!123".getBytes(Charset.forName("UTF-8"))
);
private byte[] NDEF_URI_BYTES = NDEF_URI.toByteArray();
private byte[] NDEF_URI_LEN = BigInteger.valueOf(NDEF_URI_BYTES.length).toByteArray();

@Override
public int onStartCommand(Intent intent, int flags, int startId) {

    if (intent.hasExtra("ndefMessage")) {
        NDEF_URI = new NdefRecord(
                NdefRecord.TNF_WELL_KNOWN,
                NdefRecord.RTD_TEXT,
                NDEF_ID,
                intent.getStringExtra("ndefMessage").getBytes(Charset.forName("UTF-8"))
        );

        NDEF_URI_BYTES = NDEF_URI.toByteArray();
        NDEF_URI_LEN = BigInteger.valueOf(NDEF_URI_BYTES.length).toByteArray();

        Context context = getApplicationContext();
        CharSequence text = "Your NDEF text has been set!";
        int duration = Toast.LENGTH_SHORT;
        Toast toast = Toast.makeText(context, text, duration);
        toast.setGravity(Gravity.CENTER, 0, 0);
        toast.show();
    }

    Log.i(TAG, "onStartCommand() | NDEF" + NDEF_URI.toString());

    return 0;
}

@Override
public byte[] processCommandApdu(byte[] commandApdu, Bundle extras) {

    //
    // The following flow is based on Appendix E "Example of Mapping Version 2.0 Command Flow"
    // in the NFC Forum specification
    //
    Log.i(TAG, "processCommandApdu() | incoming commandApdu: " + utils.bytesToHex(commandApdu));

    //
    // First command: NDEF Tag Application select (Section 5.5.2 in NFC Forum spec)
    //
    if (utils.isEqual(APDU_SELECT, commandApdu)) {
        Log.i(TAG, "APDU_SELECT triggered. Our Response: " + utils.bytesToHex(A_OKAY));
        return A_OKAY;
    }

    //
    // Second command: Capability Container select (Section 5.5.3 in NFC Forum spec)
    //
    if (utils.isEqual(CAPABILITY_CONTAINER, commandApdu)) {
        Log.i(TAG, "CAPABILITY_CONTAINER triggered. Our Response: " + utils.bytesToHex(A_OKAY));
        return A_OKAY;
    }

    //
    // Third command: ReadBinary data from CC file (Section 5.5.4 in NFC Forum spec)
    //
```

```
        if (utils.isEqual(READ_CAPABILITY_CONTAINER, commandApdu) &&
!READ_CAPABILITY_CONTAINER_CHECK) {
            Log.i(TAG, "READ_CAPABILITY_CONTAINER triggered. Our Response: " +
utils.bytesToHex(READ_CAPABILITY_CONTAINER_RESPONSE));
            READ_CAPABILITY_CONTAINER_CHECK = true;
            return READ_CAPABILITY_CONTAINER_RESPONSE;
        }

        //
        // Fourth command: NDEF Select command (Section 5.5.5 in NFC Forum spec)
        //
        if (utils.isEqual(NDEF_SELECT, commandApdu)) {
            Log.i(TAG, "NDEF_SELECT triggered. Our Response: " + utils.bytesToHex(A_OKAY));
            return A_OKAY;
        }

        //
        // Fifth command:  ReadBinary, read NLEN field
        //
        if (utils.isEqual(NDEF_READ_BINARY_NLEN, commandApdu)) {

            byte[] start = {
                    (byte)0x00
            };

            // Build our response
            byte[] response = new byte[start.length + NDEF_URI_LEN.length + A_OKAY.length];

            System.arraycopy(start, 0, response, 0, start.length);
            System.arraycopy(NDEF_URI_LEN, 0, response, start.length, NDEF_URI_LEN.length);
            System.arraycopy(A_OKAY, 0, response, start.length + NDEF_URI_LEN.length, A_OKAY.length);

            Log.i(TAG, response.toString());
            Log.i(TAG, "NDEF_READ_BINARY_NLEN triggered. Our Response: " + utils.bytesToHex(response));

            return response;
        }

        //
        // Sixth command: ReadBinary, get NDEF data
        //
        if (utils.isEqual(NDEF_READ_BINARY_GET_NDEF, commandApdu)) {
            Log.i(TAG, "processCommandApdu() | NDEF_READ_BINARY_GET_NDEF triggered");

            byte[] start = {
                    (byte)0x00
            };

            // Build our response
            byte[] response = new byte[start.length + NDEF_URI_LEN.length + NDEF_URI_BYTES.length +
A_OKAY.length];

            System.arraycopy(start, 0, response, 0, start.length);
            System.arraycopy(NDEF_URI_LEN, 0, response, start.length, NDEF_URI_LEN.length);
            System.arraycopy(NDEF_URI_BYTES, 0, response, start.length + NDEF_URI_LEN.length,
NDEF_URI_BYTES.length);
            System.arraycopy(A_OKAY, 0, response, start.length + NDEF_URI_LEN.length + NDEF_URI_BYTES.length,
A_OKAY.length);

            Log.i(TAG, NDEF_URI.toString());
            Log.i(TAG, "NDEF_READ_BINARY_GET_NDEF triggered. Our Response: " + utils.bytesToHex(response));
```

```
            Context context = getApplicationContext();
            CharSequence text = "NDEF text has been sent to the reader!";
            int duration = Toast.LENGTH_SHORT;
            Toast toast = Toast.makeText(context, text, duration);
            toast.setGravity(Gravity.CENTER, 0, 0);
            toast.show();

            READ_CAPABILITY_CONTAINER_CHECK = false;
            return response;
        }

        //
        // We're doing something outside our scope
        //
        Log.wtf(TAG, "processCommandApdu() | I don't know what's going on!!!.");
        return "Can I help you?".getBytes();
    }

    @Override
    public void onDeactivated(int reason) {
        Log.i(TAG, "onDeactivated() Fired! Reason: " + reason);
    }
}
```

Scan Activity

```
public class ScanActivity extends AppCompatActivity {

    private TextView currentTID;
    private Button cancelBtn;
    private SlidrInterface slidr;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_scan);

        currentTID = (TextView) findViewById(R.id.textView);
        cancelBtn = (Button) findViewById(R.id.button);
        slidr = Slidr.attach(this);

        Paper.init(this);
        String TID = getString(R.string.NoID);
        TID = Paper.book().read(Prevalent.CurrentTID);
        currentTID.setText(TID);

        Intent intent = new Intent(this, myHostApduService.class);
        intent.putExtra("ndefMessage",TID);
        startService(intent);

        cancelBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                AlertDialog.Builder confirm = new AlertDialog.Builder(ScanActivity.this);
                confirm.setTitle(R.string.remove);
                confirm.setMessage(R.string.are_you_sure);
                confirm.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {
                        Paper.book().write(Prevalent.CurrentTID, getString(R.string.NoID));
                        finish();
                    }
                });
                confirm.setNegativeButton("No", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {
                        Toast.makeText(ScanActivity.this,R.string.cancel,Toast.LENGTH_SHORT).show();
                    }
                });
                AlertDialog dialog = confirm.create();
                dialog.show();
            }
        });

    }
}
```

Utils Activity

```
public class utils {

    final protected static char[] hexArray = "0123456789ABCDEF".toCharArray();

    /**
     * Simple way to output byte[] to hex (my readable preference)
     * This version quite speedy; originally from: http://stackoverflow.com/a/9855338
     *
     * @param bytes yourByteArray
     * @return string
     *
     */
    public static String bytesToHex(byte[] bytes) {
        char[] hexChars = new char[bytes.length * 2];
        for ( int j = 0; j < bytes.length; j++ ) {
            int v = bytes[j] & 0xFF;
            hexChars[j * 2] = hexArray[v >>> 4];
            hexChars[j * 2 + 1] = hexArray[v & 0x0F];
        }
        return new String(hexChars);
    }

    /**
     * Constant-time Byte Array Comparison
     * Less overheard, safer. Originally from: http://codahale.com/a-lesson-in-timing-attacks/
     *
     * @param bytes yourByteArrayA
     * @param bytes yourByteArrayB
     * @return boolean
     *
     */
    public static boolean isEqual(byte[] a, byte[] b) {
        if (a.length != b.length) {
            return false;
        }

        int result = 0;
        for (int i = 0; i < a.length; i++) {
            result |= a[i] ^ b[i];
        }
        return result == 0;
    }
}
```