# Parts Crib Database by The Walking Programmers

Rafil Yashooa, Masoud Rahguzar, Divesh Oree

Project Website: Masoud647.github.io

March 28th, 2017

Parts Crib Database

## Declaration of Joint Authorship

The work that is specified in this report is a joint project by Masoud Rahguzar, Rafil Yashooa, and Divesh Oree. The work that has been done is our own and is expressed in our own words. We have clearly defined all work cited using the APA format expressing the authors/owners and their information. All work in this project is equally divided defined in section 2.4.4.

## Approved Proposal

Technical Report for the development of Parts Crib DataBase

Prepared by Masoud Rahguzar, Divesh Oree, Rafil Yashooa

Computer Engineering Technology Student

Masoud647.github.io

## Executive Summary

As students in the Computer Engineering Technology program, We will be integrating the knowledge and skills we have learned from our program into this Internet of Things themed capstone project. This proposal requests the approval to build the hardware portion that will connect to a database as well as to a mobile device application. The internet connected hardware will include a custom PCB with sensors and actuators for Parts Crib Database. The database will store information of students who sign out items from the parts crib. The mobile device functionality will include list amount of each item in the parts crib and how many are left after being sign out and will be further detailed in the mobile application proposal. We will be collaborating with the following company/department Humber College Information Technology. In the winter semester we plan to form a group with the following students, who are also building similar hardware this term and working on the mobile application with the following group members Masoud Rahguzar, Rafil Yashooa and Divesh Oree. The hardware will be completed in CENG 317 Hardware Production Techniques independently and the application will be completed in CENG 319 Software Project.

## Background

The problem solved by project is it will improve the efficiency of students taking out items from the parts crib. This project will connect to a database and keep track of students with their full names, student numbers and items. There will be bar codes for each items in the parts crib that will be linked to the database that contains every item in the parts crib. When a student takes out items from the parts crib their card will be scanned with the items that are taken out. This will put the students in the list(database) of students who taken out items from the parts crib. This will intern keep track of all items taken out and help keep everything organized. The mobile application then will be used to list all the items available in the parts crib and will be available to all students in the campus. The mobile application will be presented in a listed form that will have detailed information of the items in the parts crib and help students know which item to take out.

We have searched for prior art via Humber's IEEE subscription selecting "My Subscribed Content" and have found and read which provides insight into similar efforts.(Bulan & Sharma, 2011; Deng et al., 2015; Muniz, Junco, & Otero, 1999)

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

- Java Docs from CENG 212 Programming Techniques In Java,
- Construction of circuits from CENG 215 Digital And Interfacing Systems,
- Rapid application development and Gantt charts from CENG 216 Intro to Software Engineering,
- Micro computing from CENG 252 Embedded Systems,
- SQL from CENG 254 Database With Java,
- Web access of databases from CENG 256 Internet Scripting; and,
- Wireless protocols such as 802.11 from TECH152 Telecom Networks.

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

## Methodology

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

Phase 1 Hardware build.

Phase 2 System integration.

Phase 3 Demonstration to future employers.

Phase 1 Hardware build

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

Phase 2 System integration

The system integration will be completed in the fall term.

Phase 3 Demonstration to future employers

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

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

| Labour Estimates | Hrs | Notes |
|---|---|---|
| Phase 1 | | |
| Writing proposal. | 9 | Tech identification quiz. |
| Creating project schedule. Initial project team meeting. | 9 | Proposal due. |
| Creating budget. Status Meeting. | 9 | Project Schedule due. |
| Acquiring components and writing progress report. | 9 | Budget due. |

| | | |
|---|---|---|
| Mechanical assembly and writing progress report. Status Meeting. | 9 | Progress Report due (components acquired milestone). |
| PCB fabrication. | 9 | Progress Report due (Mechanical Assembly milestone). |
| Interface wiring, Placard design, Status Meeting. | 9 | PCB Due (power up milestone). |
| Preparing for demonstration. | 9 | Placard due. |
| Writing progress report and demonstrating project. | 9 | Progress Report due (Demonstrations at Open House Saturday, November 7, 2015 from 10 a.m. - 2 p.m.). |
| Editing build video. | 9 | Peer grading of demonstrations due. |
| Incorporation of feedback from demonstration and writing progress report. Status Meeting. | 9 | 30 second build video due. |
| Practice presentations | 9 | Progress Report due. |
| 1st round of Presentations, Collaborators present. | 9 | Presentation PowerPoint file due. |
| 2nd round of Presentations | 9 | Build instructions up due. |
| Project videos, Status Meeting. | 9 | 30 second script due. |
| Phase 1 Total | 135 | |
| Phase 2 | | |
| Meet with collaborators | 9 | Status Meeting |
| Initial integration. | 9 | Progress Report |
| Meet with collaborators | 9 | Status Meeting |
| Testing. | 9 | Progress Report |
| Meet with collaborators | 9 | Status Meeting |
| Meet with collaborators | 9 | Status Meeting |
| Incorporation of feedback. | 9 | Progress Report |
| Meet with collaborators | 9 | Status Meeting |

| | | |
|---|---|---|
| Testing. | 9 | Progress Report |
| Meet with collaborators | 9 | Status Meeting |
| Prepare for demonstration. | 9 | Progress Report |
| Complete presentation. | 9 | Demonstration at Open House Saturday, April 9, 2016 10 a.m. to 2 p.m. |
| Complete final report. 1st round of Presentations. | 9 | Presentation PowerPoint file due. |
| Write video script. 2nd round of Presentations, delivery of project. | 9 | Final written report including final budget and record of expenditures, covering both this semester and the previous semester. |
| Project videos. | 9 | Video script due |
| Phase 2 Total | 135 | |
| Phase 3 | | |
| Interviews | TBD | |
| Phase 3 Total | TBD | |
| Material Estimates | Cost | Notes |
| Phase 1 | | |
| Raspberry Pi 3.0(Kit), HDMI, Webcam, LED lights, PCB, Speaker Peripherals with cables Sensors | 189 | |
| Phase 1 Total | >$200.00 | |
| Phase 2 | | |
| Materials to improve functionality, fit, and finish of project. | | |
| Phase 2 Total | TBD | |

| | | |
|---|---|---|
| Phase 3 | | |
| Off campus colocation | N/A | N/A |
| Shipping | 25 | |
| Tax | 18 | |
| Duty | N/A | |
| Phase 3 Total | 43 | |

## Concluding remarks

This proposal presents a plan for providing an IoT solution for Parts Crib. This is an opportunity to integrate the knowledge and skills developed in our program to create a collaborative IoT capstone project demonstrating my ability to learn how to support projects such as the initiative described by our group, Humber parts crib. We request approval of this project.

## Abstract

The reason for this project is to develop a new parts crib system that is efficient and reliable, instead of the present way of using paper slips that are used to sign out items. The parts crib is an area that is used for technology students at Humber who can sign out hardware items that can be useful for labs and projects which is very useful. Our project will consist of three major hardware parts the Raspberry pi, webcam and barcodes. Raspberry Pi is the brain of the operation computing and processing all the actives in the parts crib. The webcam is the eyes of the operation by scanning the barcodes. Lastly, the barcodes hold the information to identify each item in the parts crib and students who take out items. All information will be fetched to a database that will categorized using firebase. The database will be working along with our app and website, these applications will be used for showing the inventory status of the parts crib. In addition, the applications will also be used for the administrator (individual who works at the parts crib.) whom will have additional options to manipulate the inventory status and the students who took out items. This is brief description of our project.

## Table of Contents

# Illustration/Diagrams

# 1. Introduction/Overview

The main point of this project is to create a new parts crib system that is more efficient and more advanced than just using paper slips to sign out item from the crib. What we have in mind for the project is to create a barcode scanner using either a web cam or either a laser scanner (if we can afford one) to scan student numbers and items that will be checked out by the students.

So, when a student comes to the parts crib for an item, they will be assigned a barcode will their student number if it's their first time coming and then they'll be ready for item take out. All the administrator has to do then is to scan the student number and then start scanning then items that that user requested as they will each have a barcode. After the scanning is done, the user is ready to go and their student number will be stored in the local database on the raspberry pi.

The barcode scanned will be processed by the Raspberry Pi which will store it on the SQL database and then the application fetches from the database and displays the information. Note: only administrators will be able to view the student numbers, regular users will only be able to view the inventory count of items in the crib.

# 2. System Requirements Specifications

The goal of this project is to create a better system for the Humber part's crib system. It is a more organized way of signing in and out items. This application is going to receive data from a Raspberry Pi that will be used as our hardware device located in the parts crib. The user has to first scan the student id and that is recognized by the first 'n' character. And then scan the part number which will be identified by a 'p' character. Every item that the user will scan will go under the student's name for check out which will go to a database stored in the raspberry pi.

This application is designed for an android device. This project also cannot work offline due to the database being a huge part of the hardware aspect. However, internet availability in the college is very static and will no need to be offline at anytime

### 2.1.1 Purpose

This project is a collaboration of the Humber Parts Crib which will allow students to sign off items for their labs on regular basis. Our main objective and goal is to make the Humber Parts Crib more effective by using cutting edge technologies.

Our first priority is to have students sign off and sign in parts as fast as possible in order to prevent crowds in the hall ways of the Parts Crib. Our project will also be providing the Parts Crib administrators with statistics on what days and hours the Part Crib is mostly occupied at. These statistics will be helpful by providing the administrators with ahead of time precreation for each class.

Our main goal is to provide as much guidance to the administrator as we can. Our project consists of 3 main parts, a web application interface hosted on the munro servers, a raspberry Pi with a webcam which will be used as a scanner, and a Android application which will be used as a inventory monitor for Humber's applied technology students.

### 2.1.2 Product Overview

The main product that we will be using is a Raspberry Pi 3, a Microsoft LifeCam Studio webcam with 720p video, and 30 frames per second, as well as a well-designed PCB with an LED indicator and an

audio amplifier circuit which will be connected to a speaker that will provide an indicator every time an item is scanned.

All of these components will be assembled inside an acrylic case designed to fit our project obligations, also for safety and provide a better presentation of our project.

The web interface is going to act as a root to control all the components of this project, the hardware branch is going to be sending information to the web interface, which is going to be stored in a database and then send availability information to the Android application inventory.

### 2.1.3 Targeted Audience Group

The main targeted audience will be the technology students at Humber College. This is because mostly technology students at Humber College take out parts at the parts crib. The administrator that will be using this design will have to be a working staff member at Humber College as privacy of student information will be presented to them.

## 2.2 Product Information

### 2.2.1 Main Functionality

The functionality is pretty simple. Basically, we will have a camera that will be used to scan the barcode strips on the student card and on the parts. The barcode information then, will be processed by the raspberry pi and will get input electronically in the system. On the new custom made PCB, we will mount a speaker and the LEDs which will allow the user to recognize if the barcode got scanned successfully or not.

### 2.2.2 Extra Requirements

In order to be able to use the device and have access to the database web interface, it is important that we have internet connection as everything will be connected on a server. As an extra requirement, we might use an RF scanner (we are planning on discussing with our Professor first before moving forward) in order to speed up the scanning of barcodes.

### 2.2.3 Best Performance

As stated above, we will mount both the LEDs and the speaker on a newly designed PCB instead of having them connected separately. The LEDs will toggle a green light if scanning is successful and then red if unsuccessful. The speaker will produce a "beep" sound which will allow the admin to recognize whether the barcode got scanned or not. We also decided that we will consider designing, building a stand and attaching it to the case which will hold the student card in order to provide a best performance when it comes to scanning the barcode. In addition scanning the barcodes should take matter of seconds to be efficient at the parts crib.

## 2.3 Overall Description

### 2.3.1 Database

For our capstone project this term, we are building a PartCrib system which hopefully be successful enough to be used by the Humber's parts crib department. This project consists of 2 major parts, hardware and software. And in order to have the hardware interact with the software, we need them both to be connected to the same database so that one can send data and the other receive data, and vice-versa.

The hardware part of this project is going to act as a scanner to scan barcode items and student id barcode. After it scan them successfully, the database would create a table for each student id it scans with a row of the exact time stamp and part number. Students would be able to sign out multiple items at once since the table can handle a max of 12 parts under it. Error checking will also be done as each part scanned will not be valid unless it passes the analyzing phase. The analyzing phase will be an array full of part numbers and part names which will assign each part to its part number, and if the scanned part id doesn't match any of the parts in that array, it will be considered as an invalid part number. This is to keep all the parts in identifying order.

The software part would be the project's web application interface which is going to be used as a control unit for Humber's staff administrators which displays the full database with the student numbers and part numbers. The android application is going to retrieve data from the database and display it live to the application for inventory tracking purposes.

For the database, we are going to be using a Google based service, the database we have up and running right now is firebase which is what we are going to be using for adding and retrieving data. Both the web application and the Android application are designed for based on firebase's libraries.

### 2.3.2 Hardware

The Humber parts crib is a project that will be able to keep track of students who take out and return items from the crib. This will occur because of three main hardware components; the raspberry pi, a camera, and barcodes. The raspberry pi is basically single-board computer which will be the brains of the project by computing and keeping track of all the student's activity in the parts crib by using the code/program implement by us the developers. Secondly, the camera, an important hardware component because it will be used to scan items in the parts crib which then will be sent to the raspberry pi. Lastly, the paper barcodes will be used to identify the items in the parts crib when scanned by the camera. Identified by a 'p' character. Every item that the user will scan will go under the student's name for check out which will go to a database stored in the raspberry pi. In addition, the Humber parts crib project will have additional hardware components such as LED lights to notify when an item is scanned or when it is unable to be scanned and there will also be a sound bar implementing the LED characteristics but with sound. The LED and audio amplifier will be placed in a single circuit board working cohesively together.

This project is designed for an web and android applications. This project also cannot work offline due to the database being a huge part of the hardware aspect. However, internet availability in the college is very static and will no need to be offline at anytime

### 2.3.3 Application

The application for the Humber parts crib is an interactive, simple and user-friendly app (available only on android) that has the potential of being very useful for students at Humber. The app will consist of two types of users, the first being the student user and the second being the administrator. The administrator users will have a username and password that will give them access to everything in the app such as inventory status, database, add items and delete items. Our main focus for the app will be the inventory. The way, the app will continuously update the inventory status is by the use of a database program called

firebase. The inventory will decrement the number of stocks of every single parts which are being signed out from the inventory and display it on the mobile app.

### 2.3.4 Database with Web Interface

For our Humber Part Crib project, our web interface will be designed using a two-tier architecture where the administrator will communicate and interact with a server. The web application will be based on a Java platform where it will communicate directly with a firebase database by using the appropriate Java Database Connectivity API. The latter will use his credentials that is his email address and password in order to log in. Inside the database, the administrator will have the privilege to check all the items available in the inventory, remove a student once the items have been returned, and also add a new student with the item parts which are being signed out from the crib. We will be using the web interface in order to decrement the parts from the stock. The web interface will be connected to the mobile application through the inventory firebase. The user will login, navigate to the sign out page and then he can either enter the student number and the part number manually or get it scanned by using the camera. Once the part is signed out, the inventory will automatically get decremented from the inventory firebase. The user can then use the mobile application in order to see which part got decremented (signed out).

As the database, will be connected on the Humber Server, we expect the wireless internet connection to be fast enough and is connected to the right access point so that the administrator can perform his daily task without any difficulty and hence making this whole project an effective one.

## 2.4 Future Considerations

### 2.4.1 Operating Environment

In order to be able to use this mechanism at Humber College, you will need a reliable internet source as internet connectivity is highly needed in order to send and retrieve data from the firebase database.

### 2.4.2 Safety Considerations

Safety precautions to take into consideration:

- Protect the device from any liquid exposure and always wear safety glasses
- Check your connections first before powering the raspberry Pi
- Always have your username and password , in order to have access to the database web interface
- Remember to sign students off after they return the parts in order to avoid database stack overflow

### 2.4.3 Future Additions

We might consider on designing and cutting out a new acrylic case as the previous one that we made was a bit too big. Also, as mentioned above regarding the RF scanner, we will make the stand first and see how it works.

### 2.4.4 Workload Breakdown

The work break down will consist of three sections the first is the database, the second is the hardware and the last section is software. Rafil Yashooa will be responsible for the database which he will connected to the website application and the hardware to work seamlessly together. Masoud Rahguzar will be responsible for the hardware, printing a PCB which will connect the speaker to the Raspberry Pi, Acrylic case using corel draw and many other aspects in the hardware. Lastly, Divesh Oree will be updating and improving the mobile application for inventory use.

## 2.5 Build Instructions

### 2.5.1 Introduction

This build log instruction will help students replicate our project for future use, we will describe and show step by step instructions on how to do so. The group members who participated to create this

project are Masoud Rahguzar, Rafil Yashooa, Divesh Oree and felt like it was their duty to make a better system to take out parts from the parts crib.

The main purpose of this project is to create a new parts crib system that is more efficient and more advanced than just using paper slips to sign out items from the crib. We created a system where each student can have up to 12 parts under her/his name and each part will be a time stamp indicating when the part was signed out. The administrator will also be altered when a student is overdue of parts as the web interface is designed to programmatically highlight what parts are overdue with in 3 days.

When a student comes to the parts crib for an item, if its their first time coming they will be assigned a barcode which will have their student number printed on a barcode, after that, they'll be ready for item take out. The administrator has to scan the student number and then start scanning then items that that user requested as they will each have a barcode. After the scanning is done, the user is ready to go and their student number will be stored in an online database.

### 2.5.2 System Diagram

The humber parts crib database project functionality is to be able to scan barcodes on student id cards and the associated part items from the parts crib and then be placed in a database. Now understanding what the concept of the humber part crib database project is, the system diagram can be easily interpreted.

### 2.5.3 Bill of Materials/Budget

The main materials/components required for our project are a raspberry pi, webcam and barcodes. Added features/materials are PCB for light indicator, sound bar and acrylic box. The PCB and acrylic box were both provided by the school. In addition, the raspberry pi requires a monitor, keyboard and mouse to be setup. Our budget was fairly simple and not expensive except for the raspberry pi kit. Our full excel version of our budget is provided in our GitHub page.

| Item | Site/Provider | Price (Including Tax) |
|---|---|---|
| Raspberry Pi 3.0 (Kit) | Amazon.ca | $135.99 |

| | | |
|---|---|---|
| HDMI | Bestbuy.ca | $15.99 |
| Web-camera | Amazon.ca | $19.99 |
| Web-camera 2 | Canada Computers | $76.01 |
| Barcode Strips | Barcodesinc.com | $4.99 |
| Acrylic Box | Humber College | $15.00 |
| PCB | Humber College | N/A |
| LEDS | Amazon.ca | $2.00 |
| | Total: | $264.97 |

~ Budget

Image 2: This picture displays the budget in a excel format

### 2.5.4 Time Commitment

The time commitment for our project took about 15 weeks to complete in its entirety. Firstly, the project began with ordering parts through websites like amazon, eBay and all sorts of technology base websites. Once ordered, the delivery took about week and a half (week 2 & 3) to come in. As everything arrived, we then began to setup our Raspberry Pi and all of its components which took about 1 hour to setup. As week 5 approached we printed out the PCB and started soldering everything. The soldering approximately took me about 2 hours to complete at school. Afterward, at week 6 and week 7 we started to test the raspberry pi and its components which took us about 6 hours in its entirety. Firstly, we tested if the PCB by implementing the code given by the teacher which tests if the light and the sensors work. Then we started implementing the code in which will scan barcodes with our webcam which then again took me about 5 hours to complete. Next, during week 9 we created the remote desktop with the raspberry pi and laptop so we could connect it remotely without using an external monitor, keyboard and mouse. Lastly, during week 12 and 13 we began and created our acrylic box, the box design took about 1 hour to complete and the lazar cut took about 20 minutes. Here at the bottom is all the task we completed and will be easier to understand.

Time Commitment Schedule:

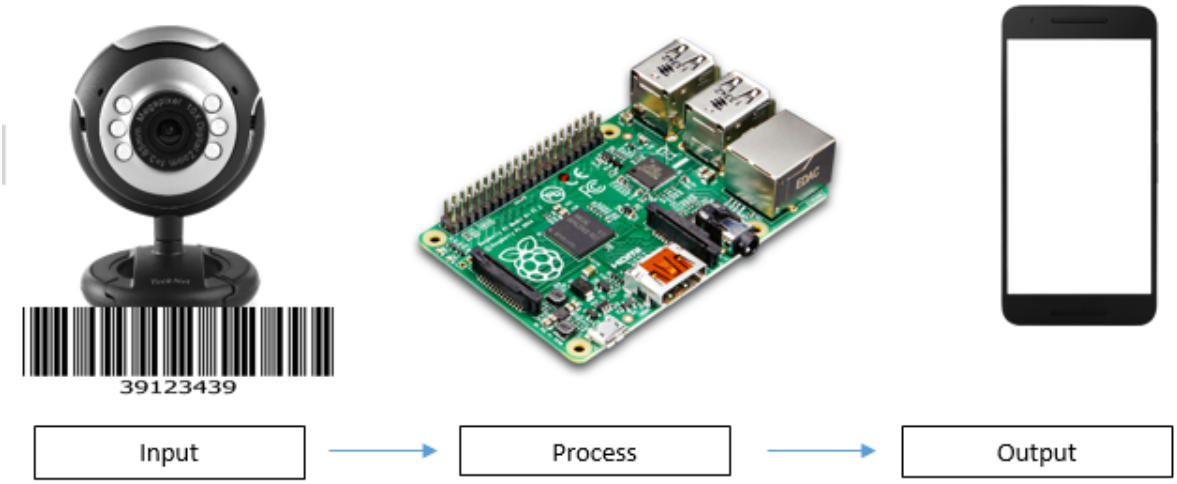


Figure 1: System Diagram

| Tasks | Time Required |
|---|---|
| Ordering Parts & Delivery | 2 Weeks |
| Raspberry Pi Setup | 1 hour |
| PCB Soldering | 2 hours |
| Testing PCB | 1 hour |
| Coding | 5 hours in span of 4 days |
| Remote Desktop | 1 hour |
| Mechanical Assembly | 15 minutes |
| Designing acrylic box | 2 hours |
| Laser Cutting | 15-30 minutes |
| Web-Application Interface | 3 Weeks (Approx. 150 hours) |
| Mobile Application | 6 Weeks (Approx. 50 hours) |

Table 1: Display time commitment in a table format

### 2.5.5 Mechanical Assembly & Setup

The assembly of our project is fairly simple, first connect the PCB on top of the raspberry pi using the GPIO pins on the raspberry pi and PCB. Secondly, proceed and connect the webcam to the USB port on the raspberry pi and then lastly connect the raspberry pi to a power outlet. And that's how to Assembly all the parts of our project to get the Humber Parts Crib working properly.

```
sudo apt-get install python-pip
```

Next, you'll need to install pillow:

```
sudo apt-get install python-pip
```

```
sudo apt-get install python-httplib2
```

After, go ahead and download the zbar library from this github account:Here

Click on "Clone Or download" and you should see it start downloading

After that, use the unzip command to extract the folder and the cd into the folder.

After you're in the folder, execute the following command: python setup.py install –user

If everything was done right, you should get no errors and the files should be extracted.

After that, use the code that that is provided in the downloads bellow and download all the downloads as they all link to each other.

After downloading all the files, you'll need to compile the red_light.c, green_light.c and also softTone.c

Compiling them using: gcc –Wall –o executable name c_file.c – lwiringPi

Where 'executable name' is the name of the executable you'll want them to be, 'c_file.c' is the source file. NOTE: make sure the executable names are the same as the ones in the bar_code.py file as they will be called from there.

If everything is done right, you should be able to run and scan a barcode, when it successfully scans the barcode, you should notice the light turning green and the frequency beep audio sound coming from the speaker, after it scans it, you should be able to see the created file called student_number.txt which will contain the stored student numbers.

### 2.5.6 Modular Sense Hat & Audio Amplifier / Library Installation/ Soldering

Humber college provided us with a PCB which is called the Modular Sensor Hat. The PCB contains 20 pin GPIO header, two 4 pin header, 5 pin header, a couple of resistors, transistors and a light. Before soldering, we took care of safety by wearing safety glasses and make sure the workplace is clear. The soldering was fairly simple to complete because of the schematic which outlined where everything goes.

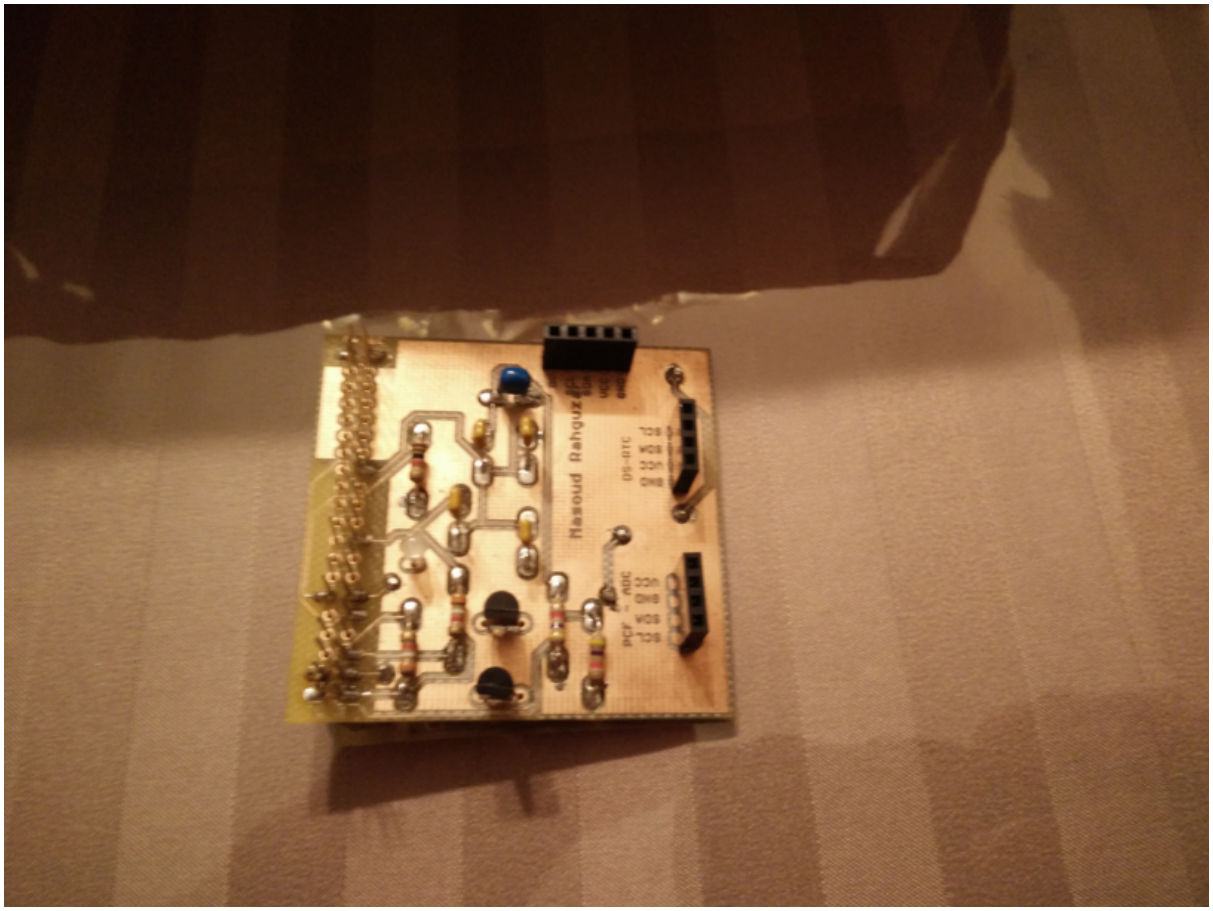

Figure 2: PCB

Also, if you don't want to use the modular sense hat, you can create a audio amplifier with an additional LED included on the same circuit board. The audio amplifier is used to use actually audio sound instead of frequency sound, the modular sense hat provides. The way this works is there will be a power cable, that will be part of the circuit board in which will provide optimal power to get the right flow of sound. Also, there will be a 3.5 mm ear phone cable that will be placed to the 3.5 mm audio out jack where the sound will be coming out from.

The audio amplifier/LED consist of a LM386N-1 IC chip, 20 pin GPIO header, capacitors, resistors, LED, power cable and potentiometer. The soldering is fairly easy compared to the modular sense hat because of the less amount of components and the huge pads to soldering.

Here is the schematic you can use to help build the audio amplifier with an addition LED light.

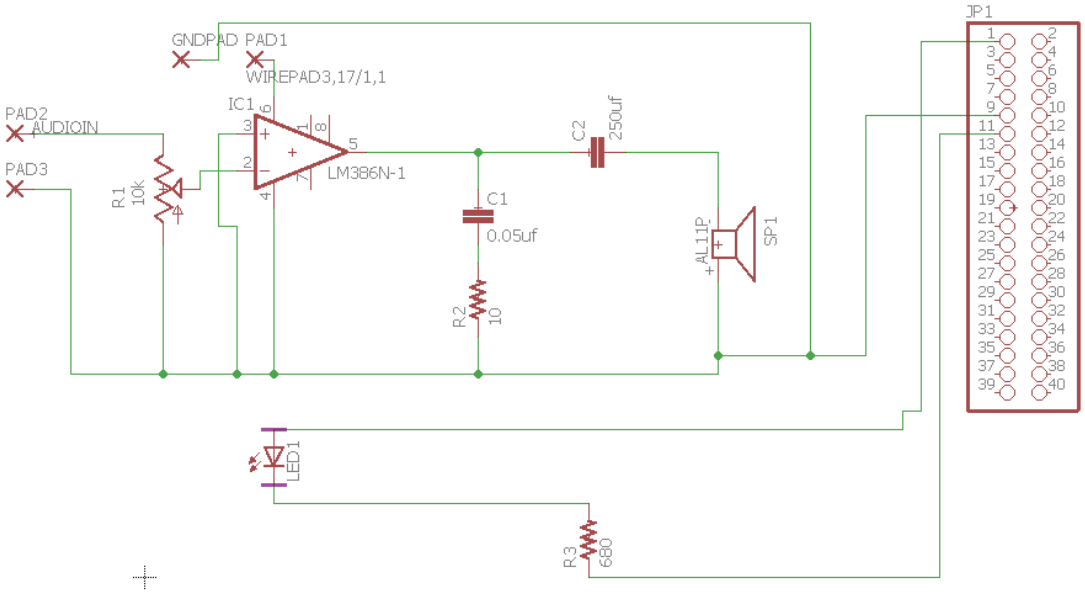


Figure 3: Audio Amplifier/LED

## 2.5.7 Web Interface Design

### Getting Started

First of all, when you're starting to build a completely new web interface, you'll need a few things planned. You'll need to make sure to answer each of the following:

- What server's will you be hosting your site on?
- What database fits best for your web interface?
- What language fits best with your database?
- Do you have a blueprint of each page you will be programming?

When you have all the answers to these questions, we can get started. In our case, we decided to use the munro servers to host our web application as it was free and provided by Humber college. For

our database, we went with Google's free online database called firebase, we decided to work with this database because the response from retrieving and adding data was instant and that's what we needed. Even though it was tough to get around things not supported by firebase, it was totally worth it. The language we choose to work with was HTML, JSP, CSS, and JavaScript. HTML was used for identifying users inputs and designing a friendly user interface while JSP and JavaScript was used to controlling the dataflow from the user. CSS was used to styling purposes as HTML doesn't provide much.

The first thing to do is to update your raspberry Pi to the latest settings, you can do so by the following commands:

```
sudo apt-get update
sudo apt-get upgrade
```

you'll want to do this as updating software gets rid of unwanted bugs that could lead to Malicious data being stoned inside your cache.

**Installing the firebase libraries**

To get JavaScript to recognize the firebase library you'll need to add a .jar file to the servers working directory. You can do so by downloading the .jar from here:

http://grepcode.com/snapshot/repo1.maven.org/maven2/com.firebase/firebase-client-jvm/2.0.2

And then adding it to working directory, don't forget to change the file's permission by executing the following command:

```
Sudo chmod 711
firebase-client-jvm-2.0.2-sources.jar
```

After doing so, your server should be firebase ready!

**Creating a firebase account**

In order to view your database and work with it from your web interface, you'll need to link your firebase account to the web application's servers.

You can do so by signing into your firebase account and clicking on “New Project” and following the instruction on the screen.

**Initializing firebase**

To add and retrieve data from firebase, you’ll need to initialize the correct configuration keys on each web interface page you’ll be using it in.

An example might look like this:

```
var config = { apiKey: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
authDomain: "humberparts.firebaseapp.com",
databaseURL: "https://humberparts.firebaseio.com",
storageBucket: "humberparts.appspot.com",
messagingSenderId: "XXXXXXXXXXXXXX”
};
 firebase.initializeApp(config);
 var ref = firebase.database().ref("pathTo/folder/");
```

Where ref will be where you’re going to be referencing data from every time you want to add retrieve data.

**Retrieving data from firebase**

Once we have the database reference from firebase, we can now retrieve whatever we have up in the database. Receiving data is called data snapshots in firebase, the snapshot function goes through all the data in the current reference directory in a loop and saves everything under a byte array. You can then step into a child by using the .child() function and set out by using the .parent() function. In our case, we are retrieving student numbers and each of the student’s parts and input that in a corresponding table.

Here’s an example of the data snapshot function retrieving live student data and inserting it in a table.

```
/*
```

```
ref.once("value").then(function(snapshot) { var i=0; var index=0; var count=0;
var partCount=0; snapshot.forEach(function(childSnapshot) { //number of children
var numChild = snapshot.numChildren(); var key1 = [childSnapshot.key]; //
childData will be the actual contents of the child var childData =
childSnapshot.val(); var dateR =
snapshot.child(key1.toString()).child("date"+count).val();
 var partR =
snapshot.child(key1.toString()).child("part"+count).val();
cell1.innerHTML+=""+key1+"";


    for(var i=0;i<20;i++){
        if(snapshot.child(key1.toString()).child("date"+i).val()==null){
            //do nothing
        }else{
            dateR = snapshot.child(key1.toString()).child("date"+i).val();
            partR = snapshot.child(key1.toString()).child("part"+i).val();
            cell1.innerHTML+="";
            date_old = new Date(dateR.toString());
            var diffDays = Math.round(Math.abs((date_now.getTime() - date_old.getTime())
            if(diffDays > 3){
                //more than 3 days old
                cell2.innerHTML+=""+dateR+"";
            }else{
                //less than 3 days old
                cell2.innerHTML+=dateR+"
            }//else
            cell3.innerHTML+=partR+"";
            partCount++;
        }//if
```

```
    }//for
    cell2.innerHTML+="'''";
    cell3.innerHTML+="'''";
    //number of students
    itemN.innerHTML=numChild;
    //number of parts in total
    part_num.innerHTML=partCount;
    //add to array for search
    arr[index]=key1.toString();
    index++;
    count++;
});


}); //snapshot
*/
```

This function is also checking if a student has a part that is over 3 days old and if that returns true, it will add a yellow highlighter under its date for each student. The cell1,2, and 3 is how each student is getting inserted into each row of the table, cell1 is for the student number, cell2 is for the part's date, and cell3 is for the last column which is the part number. The partR and dateR variables are the part and date that is getting retrieved from each student.

As you can already tell, loops are the key for this function, here we have nested for loops each working together at corresponding rates in order to retrieve different data from the database.

**Adding data to firebase**

Adding data to firebase is way simpler from retrieving data. To add data, you'll again need your firebase configuration identifiers and your reference variable will have to set to where you'll want to add data. Next, you'll want to create element identifiers so that you can get input from the html input field and use them in JavaScript as strings, to do so use the following code:

```
Var id = document.getElementById("student\_id").value;
```

In my case, the 'id' variable is the student's id and that's what's going to be added to the database including the part number provided that same way. To this data to the database, we will have to use the .update() function because using the .set() function will delete all the older data and only display what's newly added.

Function source code:

```
if(document.getElementById("part"+i).value !=null){
firebase.database().ref("dirTest/test2/"+id).update({ date0: date\_p.toString(),
part0: document.getElementById("part"+i).value }); }//1
```

This if statement is first checking if the input field is null or empty, and if that returns a true, it will skip to the next field because we don't want to be adding null fields in our database. But however, if the input wasn't empty, it would take the input from the input id 'part+i' (I as in number) and we add that into the database. The date is added using the Date() function as it is build in within the JavaScript API.

**Getting the scanner to work with the web interface**

To get started with the virtual barcode scanner, you'll need to combine what we have build from last semester and use it to scan data and send it to firebase which we can then retrieve it and display it instantly on the screen using the .focus() Jquery function so that the barcode scanned will be instantly displayed to the web interface application when the user focuses on a certain field.

**Installing the python-firebase library on a raspberry Pi**

Installing the firebase-python library is fairly easy and can be done using the following commands:

```
Sudo pip install requests
Sudo pip install python-firebase
```

**Creating the python executable file to scan and send data to the database**

Programming in python is much easier than programming in any other language, to set up firebase, all that is required is to import the firebase library and then use the following code to connect it to your own database:

```
firebase = firebase.FirebaseApplication('https://yourfirebase.firebaseio.com',None)
```

When you need to send data use the following code:

```
firebase.put('dir/path','value', symbol.data)
```

where 'smybol.data' is a variable that holds the scanned barcode information.

**Setting up the web interface to retrieve scanned barcode**

When your raspberry Pi is correctly sending data, all we have to do now is to create a function that will take the data and do something with it on a trigger. This trigger is going to be using the focus() Jquery function.

```
if($("#part0").length){
                $("#part0").focus(function() {
                        //display the value
                        ref2.once("value").then(function(snapshot) {
                                snapshot.forEach(function(childSnapshot) {
                                        document.getElementById("part0").value=childSnap
                                }); //childSnapshot
                        }); //snapshot
                });
        }//if
```

This function first checks if the element id exists, then checks if the field with 'parto' id is set on focus and if everything returns a true, it will retrieve a value from the scanned value by the raspberry Pi and input that value instantly to the input field.

**Source files download**

**All the files to the web interface can be found under my github page which is linked bellow**

https://github.com/rafyo127/rafyo127.github.io/tree/master/web_interface

## 2.5.8 Mobile Application

### Introduction

Our project plan is to create a mobile application to help keep track of all the parts (inventory) in the parts crib. This mobile application will consist of a table list all the parts in the parts crib. All parts in the parts crib will be put in a database. Each part (item) will have a column with the number (#) of its parts that remain in the parts crib. This build log instruction will basically show the functionality of the mobile application, how we created the mobile application and how are we going to implement the inventory firebase. The admin should be able to use the mobile application in order to see which parts are signed out from the inventory. Once the parts are removed from the database through the web interface, the number of stocks in the parts will be decremented.

This mockup diagram shows a general functionality of the mobile application. It shows how th**e** user can use the mobile application in order to navigate throughout. In order to access the inventory, the admin does not need to login. He can have a free access to the inventory database. In the database where he can sign out, return or view the database, he needs to login using his email address and password. After login, the user will automatically get navigated to the database section where he can choose what he wants to do. If he wants to sign out a part, all he has to do is enter his student number and the part number and that record will be added inside the view database. He can check it by selecting the view database in order to view the full database with all the records including the specific ones he recently signed out. He can then select the return items/delete where he can remove the record (student number and parts) by using the student number. We also have a logout button which will allow the user to log out from the mobile application at the end of the work day.

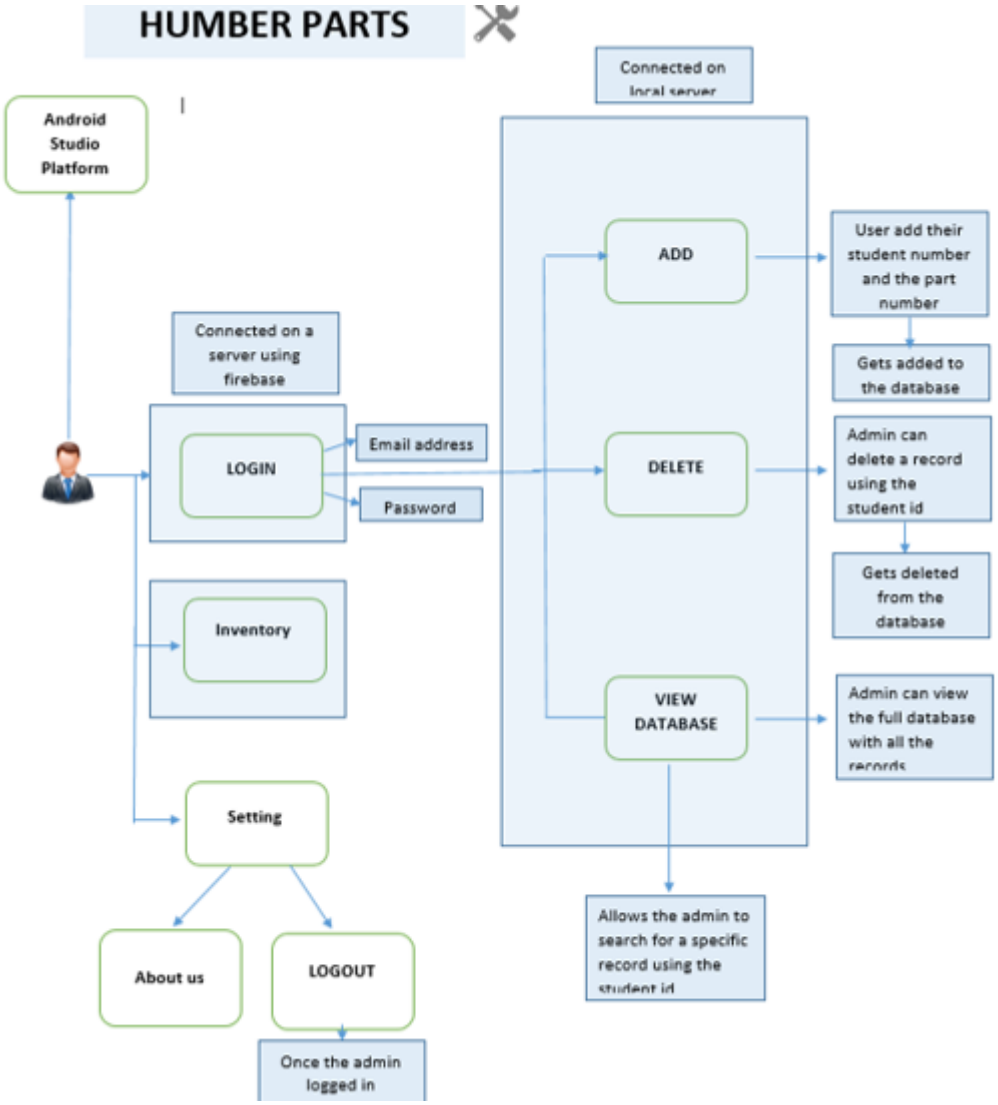


Figure 4: Mobile Application Diagram Mock up

**Gradle Build and Firebase Setup with Libraries**

The gradle should be built properly in order to build the project successfully. Make sure you have those prerequisites in order to proceed and create and build a successful project:

1. Prerequisites

- A device running Android 4.0 (Ice Cream Sandwich) or newer, and Google Play services 10.2.1 or higher
- The Google Play services SDK from the **Google Repository**, available in the Android SDK Manager

The latest version of Android Studio, version 1.5 or higher

You can also check below as an example how your gradle build should look like:

```
/\*

apply **plugin**: **'com.android.application'**

android {
    compileSdkVersion 24
    buildToolsVersion **'25.0.0'**

**   ** defaultConfig {
        applicationId **"humberparts.walkingprogrammers"**
**       ** minSdkVersion 19
        targetSdkVersion 24
        versionCode 1
        versionName **"1.0"**

**       ** testInstrumentationRunner
```

```
**"android.support.test.runner.AndroidJUnitRunner"**
**       ** testInstrumentationRunner
**"android.support.test.runner.AndroidJUnitRunner"**

**   ** }
    buildTypes {
        release {
            minifyEnabled **false**
**           ** proguardFiles
getDefaultProguardFile(**'proguard-android.txt'**), **'proguard-rules.pro'**
**       ** }
    }
}

dependencies {
    compile fileTree(**dir**: **'libs'**, **include**: [**'\*.jar'**])

androidTestCompile(**'com.android.support.test.espresso:espresso-core:2.2.2'**,
{
        exclude **group**: **'com.android.support'**, **module**:
**'support-annotations'**
**   ** })



    compile **'com.google.gms:google-services:3.0.0'**
**   ** compile **'com.android.support:appcompat-v7:24.2.1'**
**   ** compile **'com.android.support:design:24.2.1'**
**   ** compile **'com.google.firebase:firebase-auth:9.6.0'**
**   ** compile **'com.google.firebase:firebase-database:9.6.0'**
```

```
**   ** compile **'com.firebaseui:firebase-ui-database:0.4.0'**
**   ** testCompile **'junit:junit:4.12'**
}

buildscript {
    dependencies {
        classpath **'com.android.tools.build:gradle:2.3.0'**
**       ** classpath **'com.google.gms:google-services:3.0.0'**
**   ** }
    repositories {
        jcenter()
    }
}

apply **plugin**: **'com.google.gms.google-services'**

allprojects {
    repositories {
        jcenter()
    }
}

\*/
```

It is also very important to set up the firebase with proper firebase libraries. Connect it in the right way to a firebase server I in order to have the functionality of fetching live data from the database.

**Steps on how to add firebase to your project:**

It is advisable that you use the latest version of Android Studio (version 2.2 or later). In this way, you can use the Firebase Assistant in order to connect your app to Firebase instead of doing it manually.

1. Use the Firebase Assistant

- Click **Tools > Firebase** to open the **Assistant** window.
- Click to expand one of the listed features (for example, Analytics), then click the provided tutorial link (for example, Log an Analytics event).
- Click the **Connect to Firebase** button to connect to Firebase and add the necessary code to your app.

1. Manually adding Firebase using Firebase Console

**Prerequisites**

- You will need a Firebase project and a Firebase configuration file for the mobile application.

Steps:

1. Create a Firebase project in the Firebase Console in case you are missing it. Or else if you already have a google project which is linked to your mobile application, all you have to do is click Import Google Project or create new project.
2. Click Add Firebase to your android mobile application and download the config file.

Steps to download the config file:

- Open the project by signing in to Firebase
- Select setting and click on project settings
- Select the package name of the mobile application in the Your apps card
- Click download in order to download the google-services.json

1. Add the sdk by adding the rules to your root-level build.gradle file in order to include the google-services plugin then apply it at the bottom of the file in order to enable the gradle plugin. You can check the example attached above.

2. Lastly, add these libraries under the dependencies for the Firebase SDK you want to use:

```
/\*

'com.google.firebase:firebase-auth:9.6.0'
'com.google.firebase:firebase-database:9.6.0'
‘com.firebaseui:firebase-ui-database:0.4.0'

\*/
```

### 2.5.9 Unit Testing

The uniting testing begins the PCB as mentioned in the power up. It is recommended to use the code that is given by Humber College or at the GitHub page to test if the light indicator works. The light indictor can work with the command “sudo./traffic2B” if the code is there. The next step is to check if the webcam work. This can be done by installing FSWEBCAM by putting “sudo apt-get install fswebcam” in the command line of the terminal which will install an easy way to check if the camera works. Once installed type “fswebcam image.jpg” in the command line and if it takes a picture with good quality the webcam works. After this, the code created for this project can be used which is in the GitHub page. The program can run by typing “python bar_code.py” in the command line.

#### JUnit Setup and Libraries

JUnit testing can also be used in order to check the mobile application. But these requires some steps:

1. Set up your testing environment

You must have a directory module-name`/src/test/java/` where you must store the source files for the local unit tests under the Android Studio project. It is normally already exists when you create a new project. Then you will have to configure the testing dependencies for your project so that it will be able to use the standard APIs provided by the JUnit 4 framework.

1. Add libraries under dependencies

```
dependencies {
    // Required -- JUnit 4 framework
    testCompile 'junit:junit:4.12'
    // Optional -- Mockito framework
    testCompile 'org.mockito:mockito-core:1.10.19'
}
```

1. Create a local unit test class

For instance, you can follow this example for sampling purposes:

```
package humberparts.walkingprogrammers;

import android.content.Context; import android.test.InstrumentationTestCase;

import org.junit.Assert; import org.junit.Before; import org.junit.Test;

import static org.junit.Assert.assertTrue;

/\*\* \* Created by Divesh on 2016-12-12. \*/ public class DatabaseActivityTest
extends InstrumentationTestCase {


private DatabaseActivity testDB= new DatabaseActivity(getInstrumentation().getTargetCont

@Test
public void insertData() throws Exception {

    assertTrue(testDB.insertData("n123","kk", "p123"));
}
```

```
@Test
public void search() throws Exception {

    Assert.assertNotNull(testDB.search("n123"));
}

@Test
public void databaseViewer() throws Exception {
    assertNotNull(testDB.databaseViewer());
}

@Test
public void deleteData() throws Exception {
    assertNotNull(testDB.deleteData("n123"));

}
}
```

In this example, we tested the insertion of a new record in the database (Sign out items), search for a specific record, and show all the records in the database (View database) and deleting a record from the database (Return item)

**Test Cases**

We created a testplan with multiple test cases in order to see the functionality of the mobile application and check if everything is working as expected. The test cases are as follows:

**Mobile Application**

Test Case #1: Inventory

Purpose:

Have access to the inventory in order to be able to see the items name and number of items available in the stock

Precondition to run the case:

Have access to the inventory in order to be able to see the items name and number of items available in the stock

Steps:

The admin should be able to navigate to the inventory successfully.

Expected Results:

As expected

Test Case #2: Inventory Search

Purpose:

The admin should be able to search for a specific item alongside with the number of item in stock in the inventory

Precondition to run the case:

Get the id number in order to do the search

Steps:

Enter the id number in the search on the toolbar and view that specific item

Expected Results:

As expected

Test Case #3: Decrement # of stocks

Purpose:

The admin after he signs out an item, the number of stocks for that part should be decremented

Precondition to run the case:

Have the student number to be entered and the part number of the part

Steps:

Enter the student number and part number and click submit

Expected Results:

As expected

Test Case #4: View Database

Purpose:

Show the entire student currently on the database with their information (Student #, Date, Part #)

Precondition to run the case:

Enter student in the database successfully

Steps:

Enter the student in the database using the student number and part number which are being rented out.

Expected Results:

As expected

Test Case #5: Settings

Purpose:

Has 2 buttons (About us, Version #). Button 1 (About us) shows some information about our project and Button 2 (Version #), displays a toast message the version

Precondition to run the case:

Be able to navigate when setting is clicked

Steps:

Click on setting and the click on either “About us” or “Version #”

Expected Results:

As expected

Test Case #6: Back Button

Purpose:

Allow the admin to navigate back to different activities

Precondition to run the case:

Be able to a set up all the activities and link them using appropriate buttons in order to allow the user to navigate forward when setting is clicked

Steps:

Click on the “Back” button on the toolbar and go to the previous activity

Expected Results:

As expected

**Web Interface**

Test Case #1: Admin Login

Purpose:

Able to have access to the database.

Precondition to run the case:

The admin should have a username and a password

Steps:

Insert a username.

For example; admin and password: root

Expected Results:

As expected

Test Case #2: Add Student

Purpose:

The admin should be able to add a student number and the part number

Precondition to run the case:

Get the student number from the student and enter the part number of the items

Steps:

Scan/Enter the student number in the input field, scan/enter the part numbers and click the submit button

Expected Results:

As expected

Test Case #3: Remove all Student parts

Purpose:

Delete a student from the database

Precondition to run the case:

Enter the student Id

Steps:

Enter the student id in the input field and press Search ID. You will then be prompted with a table of the part numbers that the student has signed out, Press the remove all button and all the parts will be returned.

Expected Results:

As expected

Test Case #4: Remove Individual Student parts

Purpose:

Delete a student from the database

Precondition to run the case:

Enter the student Id

Steps:

Enter the student id in the input field and press Search ID. You will then be prompted with a table of the part numbers that the student has signed out, Click on the 'Remove Individual Part(s)' button and you will be prompted with input equal to the same number of parts that the student has. Enter the key for each part and click 'Submit'

Expected Results:

As expected

Test Case #5: Search student in view database

Purpose:

Search for a specific student Id to check for usage

Precondition to run the case:

Navigate to the view Database page with a student Id ready for searching

Steps:

Scan/enter the student Id and click 'Search'. You will be prompted if the student was found with 'Record Found!' or a 'Student not found!' if the student wasn't found in the database

Expected Results:

As expected

### 2.5.10 Production Testing

The production testing is fairly simple. Run the barcode program bar_code.py and place a barcode in front of the webcam. In addition, focus the webcam (manually if needed) so that it can quickly scan the barcode. Once scanned the barcode will be place in a txt file and in the terminal it can display the txt file by typing for example "cat nameoftxt.txt".

Downloads:

All the downloads will be posted under my github page:

Source Files: Here

## 2.6 Schedule

Stage 1

- Proposal Tue. 9/6/16 - Tue. 9/13/16
- Individual Project Schedule Tue. 9/16/16 - Tue. 9/20/16
- Status Meeting Tue. 9/20/20
- Individual Budget Tue. 9/14/16 - Tue. 9/27/16
- Individual Progress Report Wed. 9/28/16 - Tue. 10/11/16
- Status Meeting II Tue. 10/4/16
- Mechanical Assembly Milestone Wed. 10/5/16 - Tue. 10/11/16
- Power Up Milestone (Individual PCB) Wed. 10/12/16 - Tue. 10/18/16
- Status Meeting III Tue. 10/18/16
- Group Placard Thu. 10/20/16 - Tue. 10/25/16
- Hardware Demonstration (November 12th, 2016)
- Peer grading of demonstration Thu. 10/27/16 - Tue. 11/8/16
- Individual Build Video Wed. 10/12/16 - Tue. 11/15/16
- Status Meeting IV Tue. 11/15/16
- Individual Progress Report IV Wed. 11/16/16 - Tue. 11/22/16
- Individual Presentations Thu. 11/10/16 - Tue. 12/06/16
- Individual Filming and Demonstration Mon. 11/21/16 - Tue. 12/13/16

Stage 2

- Scheduling & Group Meetings Mon. 1/09/17
- Group Project Status Update Fri. 1/13/17 - Mon. 1/16/17

- App, Web, and Database SRS Mon. 1/09/17 - Mon. 1/23/17
- Group Project Status Update II Fri. 1/27/17 - Mon. 1/30/17
- Group Project Status Update III Thu. 2/2/17 - Mon. 2/6/17
- App, Web, and Database Independent Demonstration Tue. 2/14/17
- Group Poject Status Update IV Sat. 2/25/17 - Mon 2/27/17
- Group Integration Tue. 2/14/17 - Fri. 2/17/17
- Group Project Status Update V Mon. 3/6/17 - Mon. 3/13/17
- Group Troubleshooting Wed. 3/1/17 - Mon. 3/20/17
- Group Project Status Update VI Fri. 3/24/17 - Mon. 3/27/17
- Project Demonstration at Open House (April 8th, 2017)
- Group Presentations Mon. 4/10/17
- Group Final Report Sat. 3/18/17 - Mon. 4/17/17
- Group Video Script, Final Filming and Demonstration Sat. 3/18/17 - Mon. 4/24/17

## 2.7 Progress Reports

January 30, 2017

Progress Report

At this moment on week 4, we are supposed to handle in our progress report stating the progress of the whole project. On Week 2, we submitted our proposal and port it into Markdown and the two relevant pages printed out, the one with the inline citations and one with the references list. On week 3, we submitted our SRS (Requirement specification) which included the following sections:

1. Hardware present
2. Skeleton with SRS completed

3. Database and work breakdown
4. Application and work breakdown
5. Web and web breakdown

As of week 4 we are all on track according to our project schedule, we have started working on the web application for our project, the application is connected to a firebase server and currently has the functionality of fetching live data from the database. The application has 3 main functionalities, 'sign out item' which lets the administrator scan student ids and the part numbers. A 'Return Item' which lets the administrator scan or manually input the student id and then ability to return parts all at once. And finally a 'View database' button which lets the admin access the full database with all the current holdings, it also tells you the number of people that have items signed out in total and a search option which is a work of progress as of right now. The application was build using HTML, JavaScript and JSP for the login credentials.

As for the problems we encountered, the search option still does not work properly, it only displays the last item in the database list and doesn't find a result for the entries above. I'm hoping to fix sometime tomorrow or at the end of this day. The next step is to insert data from the raspberry pi into the database and them after, fetch that data and use it for the following functionalities.

We have not exceeded our financial status as our project consists of only programming for this stage.

Links to the web application: http://munro.humber.ca/~n01040349/

March 07, 2017

Dear Kristian,

As of week 7, we are currently on track with our schedule for our project. A lot was accomplished for this week as everyone of us had a busy project schedule. Our technical report was updated to suite the correct Technology Report Guidelines.

As of the past weeks, I have been working on the web application from the software side. I was able to fix a couple problems during these past weeks. One of the main problems I fixed was adding multiple parts at once for each student. This was a problem because of the way firebase was designed, it does not allow auto increment so I had to come up with code that does that on its own. The search function was also

fixed as it didn't work properly before, the way to fix this problem was to stack every student number in a string array and search from the array when needed. Another thing that was fixed was also the view database page where it did not print if the student had more than 1 part. Now it is able to print multiple parts and dates corresponding to each other.

As for Masoud, he has been busy developing an acrylic case for our hardware components as well as a designing a circuit schematic for our audio and LED functionalities. As of last week, Masoud printed a case which did not fit our requirements perfectly so he decided to redesign the case and add more features to fit what we have in mind for our project.

Divesh has been working on changing small aspects of the mobile app by making it more user friendly, and also updating the inventory functionally for the app. The inventory functionally will consist of how many parts there are in the parts crib and so whenever a student decides to check out a certain item, it will decrement in the inventory list. A problem that he is currently working on is how will he determine when a part has been checked out by a student. On solving this problem, he will have to collaborate with me to have a better understanding of how parts are being sign-out out from the web application aspect.

In the coming weeks, we plan to continue working on our individual parts of this project as well as keeping the technical report up to date. There are no changes to the financial status as it stays the same. We hope to have the web application fully functional and connect to the hardware by open house ready for demonstration.

Sincerely,

Rafil Yashooa

March 21, 2017

Dear Kristian,

As of this week, a lot of progress has been done towards the hardware and the software of our project. We are currently on track as of this week's progress for our project. The project budget has been updated as we purchased a new 720p HD camera and also updated the technical report as Masoud added the project conclusion.

As we tested our scanner last week, we figured that the time that was taking the camera to scan the

barcode was way off of what we expected so we decided to purchase a new camera to fix this issue. The first camera we bought was a 1080p Logitech with auto focus. As the package arrived on Thursday, we tested and timed the camera and found out that it took 4 seconds to focus and scan the barcode. Knowing how fast paced the Humber's part crib is, we decided to purchase a higher quality camera that was known to preform better. We talked to our previous group member Gurpreet and he advised us to purchase a Microsoft LifeCam Studio. We are looking forward to testing this new camera as soon as it arrives.

Our audio amplifier's circuit is also fully designed ready to be printed as our previous method was just throwing off a frequency of 350KHz without an RC filter to the speaker which was creating a static sound when it was powered. The acrylic case was also redesigned to fit our new camera. The Pi is now mounted on the case using 4 M2.5 screws, we also hope to mount the camera on the acrylic case by purchasing a 3/8"-16 male screw to hold the camera from the bottom.

As for the web application, I talked to Kelly last week about improving the web interface of the application and I was given a couple of suggestion that I needed to improve. One of the things we talked about was having an auto drop input field so that whenever the admin clicks on scanning a part number, it would automatically drop a box for the next part. Another thing we talked about was altering the admin on whenever a student is late on returning a part. This can be done by highlighting the overdue parts yellow on the view database page.

I started integrating these 2 features early last week but have not had the time to complete them fully, I hope to have them complete by next week. Our financial status has been increased by $76.01 (Camera purchased from Canada Computer) and makes a total of $264.97 for our project budget.

Sincerely,

Rafil Yashooa

# 3. Conclusion

The project, Humber Parts Crib has been created to keep track of all items in the parts crib when items are lent out and returned back. The Humber Parts Crib consists of a camera, raspberry pi, PCB (speaker & lights) and a case. This project intern will be able to scan barcodes representing the students and items. The data will be placed in a database called firebase using the website application which will be fetched by the mobile application. The mobile application will be able to display the inventory status of the parts crib, helping students determine if there is an item they want in the parts crib. The website application will contain the administrative tools/control to be able to lend out and return items which will be used by the employee at the parts crib.

# 4. Recommendations

Throughout the creation process of our project there are recommendations that could be made if reproduced again. The first being the web camera, having a better web camera will help accelerate the speed at which barcodes are scanned, thus increasing the performance. In addition, having small web camera can allow for a smaller and compact case which can be portable if need. Secondly, have a better designed circuit board for the audio amplifier in which power can be drawn out of from raspberry pi instead of a power outlet can help lessen the need of power cables and any other type of cable. Lastly, making the code much more efficient allowing for better results when scanning barcodes.

# 5. References

Bulan, O., & Sharma, G. (2011). High capacity color barcodes: Per channel data encoding via orientation modulation in elliptical dot arrays. *IEEE Transactions on Image Processing*, *20*(5), 1337–1350. https://doi.org/10.1109/TIP.2010.2092437

Deng, X., Zijlstra, P., Zhang, J., Wu, Y., Zhou, G., & Linnartz, J. P. M. G. (2015). Performance of barcode scanner using peak detection with interference from lED lamps. In *2015 iEEE symposium on communications and vehicular technology in the benelux (sCVT)* (pp. 1–6). https://doi.org/10.1109/SCVT.2015.7374231

Muniz, R., Junco, L., & Otero, A. (1999). A robust software barcode reader using the hough transform. In *Proceedings 1999 international conference on information intelligence and systems (cat. no.PR00446)* (pp. 313–319). https://doi.org/10.1109/ICIIS.1999.810282