**Prototype Lab Flow**

Projects blog: [http://ssehra.github.io](http://ssehra.github.io/)

**Table of Contents**

1. [Introduction](https://github.com/ssehra/ssehra.github.io/blob/master/README.md#introduction)
   * [What is the Project?](https://github.com/ssehra/ssehra.github.io/blob/master/README.md#what-is-the-protoype-lab-flow-project)
   * [Why is is useful?](https://github.com/ssehra/ssehra.github.io/blob/master/README.md#why-is-is-useful)
2. [Build Budget](https://github.com/ssehra/ssehra.github.io/blob/master/README.md#build-budget)
   * [Core items needed for the this project](https://github.com/ssehra/ssehra.github.io/blob/master/README.md#heres-a-breakdown-of-the-item-needed-to-build-this-project)
   * [List components required for the PCB](https://github.com/ssehra/ssehra.github.io/blob/master/README.md#the-list-of-pcb-components-needed-to-build-the-pcb)
3. [Time Commitment](https://github.com/ssehra/ssehra.github.io/blob/master/README.md#time-commitment)
4. [PCB Soldering](https://github.com/ssehra/ssehra.github.io/blob/master/README.md#pcb-soldering)
   * [Schematic for the PCB](https://github.com/ssehra/ssehra.github.io/blob/master/README.md#schematic-for-the-pcb)
   * [My Completed PCB](https://github.com/ssehra/ssehra.github.io/blob/master/README.md#ariel-view-of-my-completed-pcb)
5. [Mechanical Assembly & Power Up](https://github.com/ssehra/ssehra.github.io/blob/master/README.md#mechanical-assembly--power-up)
6. [Unit Testing](https://github.com/ssehra/ssehra.github.io/blob/master/README.md#unit-testing)
   * [Webcam functionality](https://github.com/ssehra/ssehra.github.io/blob/master/README.md#webcam-functionality-testing)
   * [PCB functionality testing](https://github.com/ssehra/ssehra.github.io/blob/master/README.md#pcb-functionality-testing)
7. [Scanning QR Codes](https://github.com/ssehra/ssehra.github.io/blob/master/README.md#scanning-qr-codes)
   * [Installing zbar](https://github.com/ssehra/ssehra.github.io/blob/master/README.md#now-we-install-the-zbar-library-to-scan-qr-codes-follow-these-step-below)
   * [Core Programs](https://github.com/ssehra/ssehra.github.io/blob/master/README.md#i-used-the-follwing-code-to-scan-qr-codes-to-implement-the-qr-code-scanner)
   * [Program Execution](https://github.com/ssehra/ssehra.github.io/blob/master/README.md#to-run-the-final-code-follow-these-steps)
8. [Ending Notes](https://github.com/ssehra/ssehra.github.io/blob/master/README.md#ending-notes)

**Introduction**

This document will provide you with all the necessary information required to build your very own "Prototype Lab Flow" project.

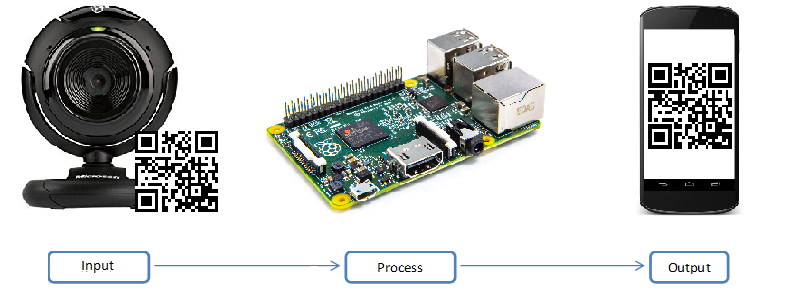
***What is the Prototype Lab Flow project?***

Well in simple terms, this project is a realtime database that allows the user to add/remove items into the database by simply scanning QR Codes.

***Why is is useful?***

This project replaces the current item sign-out system used at the Humber Prototype Lab as well as the Humber Parts Crib. The current system requires a student to fill out a form and lend their Humber ID Card in order to borrow items. This system is a slower process because students must wait in queues just a fill out a form. This project makes this system more efficient by allowing the administrator to simply scan QR Codes instead of waiting for the students to fill out their forms.

If you are still on how the project will work or what will it do the systems diagram below should be helpful.

[](https://github.com/ssehra/ssehra.github.io/blob/master/Build%20Log%20Files/Diagram.png?raw=true)

*The diagram above shows input, output and processing components of this project. The Webcam will scan QR codes as input while the Raspberry Pi will process that input to add into a database. The final output will the an android application showing that the item has been added to the database. However for now, a feedback from L.E.D. blinking will replace the android application in the systems diagram.*

**Build Budget**

Before we start building something, we need to plan a budget. A budget will be especially helpful in managing the financial costs related to our build.

**Here's a breakdown of the item needed to build this project.**

1. A Raspberry Pi
2. USB Webcam
3. A MicroUSB AC adapter
4. A Case for Raspberry Pi
5. HDMI Cable
6. A MicroSD Card (8GB or higher)
7. K
8. A

**The list of PCB components needed to build the PCB**

* 2x20 Header Female (Qty: 1) $3.73
* 100nF Capacitor (Qty: 4) $5.80
* 10uF Capacitor (Qty: 1) $1.45
* 1kΩ Resistor (Qty: 1) $0.15
* 3.9kΩ Resistor (Qty: 4) $2.40
* 4.7kΩ Resistor (Qty: 3) $1.35
* 3mm LED (Qty: 1) $0.61
* N-Channel Logic Level MOSFET (Qty: 1) $0.99
* R
* I
* 8
* 1
* 1

Let's start with the core of the project, a Raspberry Pi. The Raspberry Pi alone costs around $60-70 online if you're lucky enough find one. I highly recommend opting in for the Raspberry Pi starter kit instead as it comes with all the necessary items like a microSD card, an AC adapter and a case (as well as other stuff like a microSD reader). It's available for about $20-30 extra.

A case is highly recommended as the raspberry pi is relatively fragile microcomputer. A microSD card is required to run the Raspbian OS on the pi. And of course, the AC adapter can be substituted with any microUSB phone/tablet charger.

Moving on, a webcam will be required to scan QR codes. Any USB webcam will do fine. So no need a spend the extra bucks for a high definition webcam. An average webcam will cost between $20 and $35. Or even cheaper if you find a used/refurbished one.

Finally, there is cost associated with the fabricating of a PCB (Printed Circuit board) as well it's components. The Prototype Lab at Humber College charges about $40 to print a PCB. The components I have on my PCB cost about $42.52. However, I do a have a bunch of extra components like sensors and headers that you don't need. This is why is I have them crossed out on the list. Moreover, this brings the cost down from $42.52 to $16.49. The actual cost might still be higher since not all components are sold in the quantity of single units. As a Humber student, all of this is covered as part of my tuition fees.

Altogether I spent about $100 on the Raspberry Pi starter kit from Amazon. If you buy an older Raspberry Pi, you have to spend extra on a USB Wi-Fi receiver or use the Ethernet instead. I used my old webcam which I bought for about $35-$40 at the time. And I didn't have to pay extra for the PCB since I'm a Humber student as well as the PCB components. I'm not including a monitor, and a keyboard and a mouse since almost everybody owns at least one of each. If not then you can buy any monitor with an HDMI and any USB or Bluetooth keyboard/mouse combo.

Here's a table that shows all the items I bought and how much I paid for them. Note all prices are in CAD. I'm also including the PCB fabricating cost to create a reference for you to compare total costs.

| **Item** | **Cost** |
| --- | --- |
| Raspberry Pi 3 Starter Kit | $99.99 |
| Microsoft LifeCam VX-1000 | $34.99 |
| PCB Fabrication Cost | $40 |
| Required components for PCB | $42.52 |
| **Total** | **$245.78** |

My final build cost totals to about approximately $200 (That's including taxes and other expenses like shipping and handling).

**Time Commitment**

Time is a resource arguably as valuable as money. And this project demands plenty of it.

I had almost 15 weeks to work this project however with these detailed instructions I will be cutting that time significantly for anyone wanting to recreate this project. Majority of my time was spent in implementing the QR scanning functionality and blinking the L.E.D. for feedback. It was my first time soldering and I was able to solder the PCB in less than an hour.

If you already have acquired all the parts and got the PCB printed, I believe this project can be completed in a day. That's if you copy paste the all the code and it runs flawlessly without any errors.

**PCB Soldering**

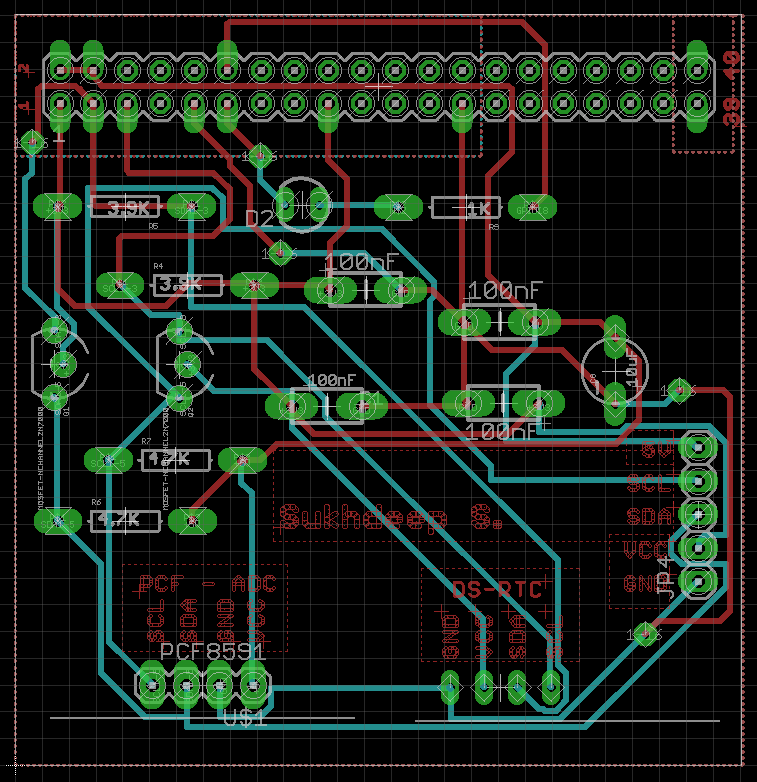
Time to start building the PCB! However you build something, you must design it first. That's where these files come in handy. Go ahead and download these files.

| **Board File** | **Schematic File** |
| --- | --- |
| [HSHV4-student version.brd](https://github.com/ssehra/ssehra.github.io/blob/master/Build%20Log%20Files/HSHV4-student%20version.brd) | [HSHV4-student version.sch](https://github.com/ssehra/ssehra.github.io/blob/master/Build%20Log%20Files/HSHV4-student%20version.sch) |

Now that you have these files, you'll need eagle to open them. Eagle can be downloaded **-->** [here](https://cadsoft.io/). After installing Eagle, edit the name in the files to your own. After you finish editing, go ahead and follow the instructions **-->** [here](https://learn.sparkfun.com/tutorials/using-eagle-board-layout/generating-gerbers) to generate the following files.

| **Top Copper (.GTL)** | **Bottom Copper (.GBL)** | **Top Soldermask (.GTS)** | **Bottom Soldermask (.GBS)** | **Top Silkscreen (.GTO)** | **Bottom Silkscreen (.GBO)** | **Drill File (.TXT)** |
| --- | --- | --- | --- | --- | --- | --- |
| [HSHV4-student version.GTL](https://github.com/ssehra/ssehra.github.io/blob/master/Build%20Log%20Files/HSHV4-student%20version.GTL) | [HSHV4-student version.GBL](https://github.com/ssehra/ssehra.github.io/blob/master/Build%20Log%20Files/HSHV4-student%20version.GBL) | [HSHV4-student version.GTS](https://github.com/ssehra/ssehra.github.io/blob/master/Build%20Log%20Files/HSHV4-student%20version.GTS) | [HSHV4-student version.GBS](https://github.com/ssehra/ssehra.github.io/blob/master/Build%20Log%20Files/HSHV4-student%20version.GBS) | [HSHV4-student version.GTO](https://github.com/ssehra/ssehra.github.io/blob/master/Build%20Log%20Files/HSHV4-student%20version.GTO) | [HSHV4-student version.GBO](https://github.com/ssehra/ssehra.github.io/blob/master/Build%20Log%20Files/HSHV4-student%20version.GBO) | [HSHV4-student version.TXT](https://github.com/ssehra/ssehra.github.io/blob/master/Build%20Log%20Files/HSHV4-student%20version.TXT) |

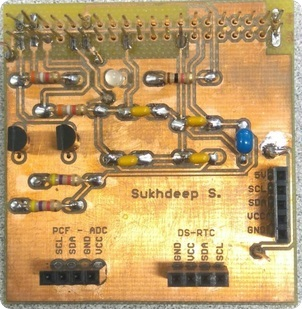
**Schematic for the PCB**

[](https://github.com/ssehra/ssehra.github.io/blob/master/Build%20Log%20Files/Schematic.PNG?raw=true)

All of these files are used to instruct the machine with the coordinates and to engrave the PCB. It takes about 4-5 hours for the machine to print it. After it's done, wash with water *(preferably deionized water)* and wipe it a piece or cloth to avoid leaving water marks (I recommend using a micro-fiber cloth) and wait for it to dry completely. After that's done you can sand the PCB a bit just to make it shinier. If you do sand it you have the repeat the washing process again.

Now the PCB is ready to soldered. However, safety comes first. Make sure to wear safety glasses and dress appropriately for a lab/workplace environment. Now Use the schematic as the guide and solder each of the components to their respective places and making sure that the excess solder is not causing a short. Now inspect each soldering carefully again use a voltmeter to test for continuity. If all goes well you are ready to move on to the next chapter.

***Ariel view of my completed PCB***

[](https://github.com/ssehra/ssehra.github.io/blob/master/Build%20Log%20Files/PCB.png?raw=true)

**Mechanical Assembly & Power Up**

By now, you have acquired all the parts you are ready to start doing mechanical assembly. It's extremely easy from here onwards since this guide a shortcut to the finish line.

You should now have the following: a Raspberry Pi, a USB Webcam, a MicroUSB power source and A MicroSD Card (8GB or higher). The starter kit's MicroSD card comes with pre-loaded Raspbian.iso so all you have to do is, install it. If it doesn’t, then it can downloaded from the Raspberry Pi website and copied on the MicroSD via a card reader (also included in the starter kit). Insert the flash card into Raspberry Pi.

Begin by connecting power, HDMI and a keyboard and mouse and the USB Webcam to the Raspberry Pi. Also connect your PCB to the Raspberry Pi. After it boots to the desktop, use **sudo apt-get update** on the command line terminal to update the Raspbian to the latest version. Then use **sudo apt-get dist-upgrade** to upgrade all the installed packages. These commands require internet (Ethernet or Wi-Fi) to download the updates.

That's it with the assembly you're ready to scroll down.

**Unit Testing**

**Webcam functionality testing**

For testing, start by installing the fswebcam library by typing **sudo apt-get install fswebcam**. Then type fswebcam followed by a filename with .jpg extension. For example, **fswebcam image.jpg**. This will snap a picture and save it the current directory with the filename image.jpg. If this works, that means the USB Webcam functions properly.

**PCB functionality testing**

To test the PCB, run the the traffic2B.c (source code available **-->** [here](https://github.com/ssehra/ssehra.github.io/blob/master/Build%20Log%20Files/traffic2B.c)) with the following steps.

1. Type **sudo apt-get install wiringPi** on the terminal to the the Pi Wiring Library
2. Compile the file using **gcc -Wall -o traffic2B traffic2B.c -lwiringPi** (this creates an executable object called traffic2B)
3. Run the file using **sudo ./traffic2B**

This should start blinking the L.E.D's in a traffic light pattern for an endless loop. Use **Ctrl+C** to exit the program. If this works, this means the PCB is ready to be used.

You are now ready to add the code to implement the QR Code scanning.

**Scanning QR Codes**

You can generate your own QR Codes ---> [here](http://www.qr-code-generator.com/), or any other website of your choosing. If you are using a service thats allows you to set the pixel count of the QR code, I recommend setting it to the minimum as it's faster and can be handled by webcams with lower resolution.

**Now we install the zbar library to scan QR codes. Follow these step below:**

1. Type **sudo apt-get install zbar-tools** on the terminal
2. Use **ls /dev/video\*/** to find your video source
3. Test by scanning a QR code by typing **zbarcam /dev/video#** on the terminal. (Note: the pound sign "#" is not literal value, it's supposed to represent the number that step 2 returns returns).

Your webcam should now be working, so go ahead and scan a QR code. You should now see the value of QR Code on the terminal saying preceded by the string "QR Code:". You can scan as many QR codes as you'd like. Use **Ctrl+C** to exit the program.

Congratulations! You've just implemented a QR Code scanner. Hurray!.

Nope. You're still not done yet. You need to add the capability to blink the L.E.D as well as save the results into a file for external use (ex: storing information into a database). Keep reading to find out how you add these.

**I used the following code to Scan QR Codes to implement the QR code scanner**

| **Shell Script** | **Python Blink** | **greenLight.c** |
| --- | --- | --- |
| [scanqr.sh](https://github.com/ssehra/ssehra.github.io/blob/master/Build%20Log%20Files/scanqr.sh) | [alertLight.py](https://github.com/ssehra/ssehra.github.io/blob/master/Build%20Log%20Files/alertLight.py) | [greenLight.c](https://github.com/ssehra/ssehra.github.io/blob/master/Build%20Log%20Files/greenLight.c) |

The python code is pretty simple it turns the the L.E.D on for 1/8th of a second and then turns it off for the same duration. This cycle is repeated 11 times to show the L.E.D feedback.

To change the 1/8th second on/off duration simply change the **time.sleep(0.125)** value to any number you like in the blink function. Lower values results in short durations and conversely higher values result in longer durations.

def blink(pin):

GPIO.output(pin,GPIO.HIGH)

time.sleep(0.125)

GPIO.output(pin,GPIO.LOW)

time.sleep(0.125)

return

Moreover you can also change the number of cycles for how many times this led on/off will be repeated. Simply change the loop count by changing 11 to any value you desire.

for i in range(0,11):

When it's all done you test run this program by typing **python alert.py** on the terminal. The L.E.D. should display according to your modifications.

The C file *greenLight.c* is just there to initiate the green light for use during the scanning process. It just blinks the led in green for 1/10th of the second. I recommend leaving it as is.

The Shell Script file *scanqr.sh* is where the bulk of the program lies. It starts off by compiling and executing *greenLight.c*. Then it defines a few variables for storing information. Every session of script execution produces a file called "scan" followed by a timestamp and the .txt extension. You can change the output filename by editing the variable $ScanResult.

# Name of scan results file

ScanResult="$cwd/scan\_$DATE.txt"

The function *scan()* initiates zbarcam to scan QR codes and then kills the task after saving the output. It even asks the user to scan multiple entries. Since it's using zbarcam you apply modifications by using flags like *--prescale* to set the resolution of the scanning resolution.

function scan() {

zbarcam --raw --prescale=320x240 /dev/video0 > $tmp &

...

# Kill tasks, free up space and call test.py to blink L.E.D.

# Append scan results to file

# Show scan results

# Prompt to continue scanning

}

More information about the flags and their usages can be found --> [here](http://manpages.ubuntu.com/manpages/xenial/man1/zbarcam.1.html).

**To run the final code follow these steps:**

1. Make sure all three core files: *greenLight.c*, *alertLight.py* and *scanqr.sh* are in the same directory.
2. On the terminal, use **chmod 777** followed by the filenames and grant all Read-Write-Execute permissions to the core files.
3. Type **./scanqr.sh** to execute the shell script.

You should now have the program working. The program scans QR code and then blinks the L.E.D. as well showing the scanned item on the screen. After scanning an item it prompts to scan another. If you choose yes, it scans another item and prompts to scan more items. If you choose no, then the program exits and shows the list of items scanned during the session. According to my code, every session generates a new file with a timestamp. You can use one file and reuse if you want. Furthermore, you can press **Ctrl+C** anytime to exit the program.

Congratulations! You've just implemented the full QR Code scanner program. Hurray!.

**Ending Notes**

I hope you found this guide helpful and were able to reproduce your very own Prototype Lab Flow project. With everything functioning as it should and costing you a total of hopefully less than $300. Here's a sample output of the program.