SELF CHECKOUT CART

FINAL REPORT

Team 13 — Rohan Khanna, Pooja Kankani and Cherian K Cherian

ECE 445 Final Report —Spring 2020

TA: Johan Mufata

Abstract

Customers find it difficult to checkout items like fruits using the self checkout terminals as these items lack barcodes. The Fall 2014 Smart checkout team aims to solve this problem by using computer vision to automatically identify items placed on the self checkout panel. In the new solution, a programmable LCD screen is placed next to each group of items missing a tag. Customers will use the barcode scanner attached to each shopping cart to scan barcodes of items generated on the LCD screen.The items are then automatically added to their online cart from where they can checkout. The new solution is less expensive, requires lower computational power and has a higher accuracy.

Table of Contents

1. Second Project Motivation

Scanning items without barcodes is a challenging task at self checkout terminals. Optimizing this process has great benefits as it saves valuable customer time and optimizes the work time of the staff on duty. The Fall 2014 self-checkout project attempted to solve this problem by using a computer vision based approach. Their solution involved using a camera to identify fruits and other items without tags so that people do not have to do so manually. While their solution was unique and interesting, it seemed to trade off time for convenience. This was not the best way to approach this in our mind and the convenience of their approach did not make up for the time lost. Further, the accuracy of such an implementation was questionable and could lead to even more time lost.

Thus, the motivation for our project was to find a way to make scanning of items without any tags easier.Our solution should be easy and convenient to use for the end user. It should optimize their checkout experience by reducing their wait times.

1.1 Updated Problem Statement

Self checkout terminals have become an essential component of any major supermarket chain. They cut labor costs and make the checkout process efficient. However, these terminals are very inefficient when it comes to scanning items that do not have barcodes such as fruits and vegetables.Gizmodo reports that one of the main reasons people hated self checkout carts was that a cashier or supervisor had to be called over to enter some arcane code into the system for every other item[1]. Vox reports that a lot of people end up not buying certain items or just outright steal items because they get angry that an item won't scan and figure it's not their job to try that hard[2]. Moreover, 40% of customers complain about having to call staff when using the self checkout for entering data for food. Hence the customer suffers because he/she struggles to find the correct barcode and wastes time waiting for the attendant. In addition, the supermarket suffers because often people end up entering wrong barcodes which end up changing the cost of the item the customer is purchasing. Furthermore, the staff wastes time sorting out trivial barcode issues.

1.2 Updated Solution

We propose to solve this problem by designing a display box that will be placed next to each shelf with grocery items likely to be missing a tag. This box will contain a programmable LCD screen that shows the information for the items placed on the shelf. For example, if the shelf stores apples, the LCD screen will show the price per pound for the apples, the type of apples, and other useful information like when the apples were stocked or their specific kind. This can be changed by the store managers if the product on the shelf changes. The box will also have a weighing platform that the user can place the items on, and the LCD screen will show a barcode that stores the corresponding price for that weight of that particular item.There will be a detachable scanner attached to each cart at the entrance. The person shopping can use the scanner that they have attached to their carts to

scan the barcode. It will keep adding the price for all the items they have shopped. This way the scanner will keep the total bill of the person ready and they can pay using an app, without having to wait in any check out counter lines. This makes it extremely easy to self checkout. The handheld scanner can also be used to scan the barcode of packaged products like bread and will add the cost of those items to the customer bill as well.

This solution helps the customer do away with the time sink associated with searching for the barcode of fruits and vegetables at the checkout counter. Barcode scanner's scan a barcode within 10ms. In addition, the ability to pay on the app enables the customer to exit the store without standing in any line.

There are 4 main differences in our projects.

1. Accuracy
2. Processing power
3. Time sink
4. Data use

The original solution aims to solve the problem using computer vision. In this solution, the customer places the item on the self checkout panel. Thereafter, a camera takes images of the item while a motor rotates it on the scale. Based on these pictures, an edge and color detection algorithm tries to identify the item. There are many issues with this approach that the new solution addresses. Firstly, the accuracy of the image detection algorithms in the first approach is only 87.5% at best. Barcode scanning on the other hand has only 1 error in 1.7 million scans. A solution that is only 87.5% accurate in a best case scenario, will show errors for 2 out of every 20 fruits a customer buys. The time sink to fix these errors will waste customer time.

	Accuracy (in %)
Sobel	75%
Canny	87.5%

Figure 1. Original solution project accuracy

Barcode Type	"Worst case" Accuracy Rate
Data Matrix	1 error in 10.5 million
Code 128	1 error in 2.8 million
Code 39	1 error in 1.7 million
UPC	1 error in 394,000

Figure 2. New solution project error rate

In addition, the original solution is computationally expensive and requires a development board with onboard memory that is capable of processing 20mb images within the time limit of 4 seconds specified by the original team. This requires the use of a 1.37GHZ processor while our project needs to operate at only a 27MHZ frequency. As a result our microcontroller is 67$ cheaper than the board in the original solution. To train the dataset for the computer vision algorithm correctly, the original solution requires a dataset of 20622 images, each of which is 20mb large. This corresponds to a data requirement of 414 gb just for the images. The only data requirement of our solution is to store the items in the database. Since companies already have inventory databases for their products, our solution works beautifully in line with existing infrastructure.

The customer would love such a solution as it eliminates lines and the painful process of manually entering barcodes for food items. This solution would increase customer satisfaction and help better use the labour time of the staff. The supermarket would benefit from this solution as it could use the LED boxes for inventory management. The LED boxes alert the staff when a certain item needs restocking. The ability to checkout with the app enables the supermarket to reduce its labor costs and reduce its investment in self checkout machines. In addition, the supermarket could coordinate all of its promos on the app instead of sending out coupons over mail.

Walmart attempted to create a similar solution. However, instead of using a barcode scanner they used the phone's camera to scan items. Customers found this inconvenient, as a lot of customers did not have high quality cameras on their phones that could scan the barcode properly.Moreover, in undeveloped countries not everyone has a smartphone which makes it challenging to implement such a solution. According to CBC News, Customers of Walmart specifically complained that they still had to go to the checkout counters for scanning items like produce and this eliminated the whole point of the scan and go solution. By allowing barcode scanning to produce items through the led screen

mechanism, our solution enables the customer to scan produce as well. Furthermore, by providing a handheld scanner our solution takes the onus off the mobile phone of customers to do the scanning. The handheld scanner also has a greater accuracy in scanning and scans the barcode within 10ms. This solution also works in the event the phone switches off.

1.3 Updated High Level Requirements

- The display box should automatically sync with an online server every 5 hours and update contents of its LCD display such as the time the product arrived, quantity left and other relevant information.

- When the user places an item on the scale, the LCD screen should show the price barcode for the selected weight for 15 seconds before going back to the price/pound screen. The weighing error should be within a tolerance of 0.1%.

- Once the item is scanned, there must be add/delete options for the item available on the software application cart within 1s and an option to make a payment through the app.

1.4 Updated Visual Design

This picture represents the two different devices that this project consists of - the handheld scanner and the programmable shelf box. On each shelf, there will be a box pre-programmed with the prices and will have a screen displaying information specifically for that shelf item. Once the user places the desired quantity of that item on the weighing plate of that shelf, a barcode will be generated for that item and displayed on the screen. The customer can now scan this barcode using the handheld scanner, which in turn will communicate to the mobile application over WiFi. On the phone, the user will be able to see the cart of selected items and their prices. The user can remove items if he/she wants, using the delete button on the app. He/She can also make their final payment through this application securely.

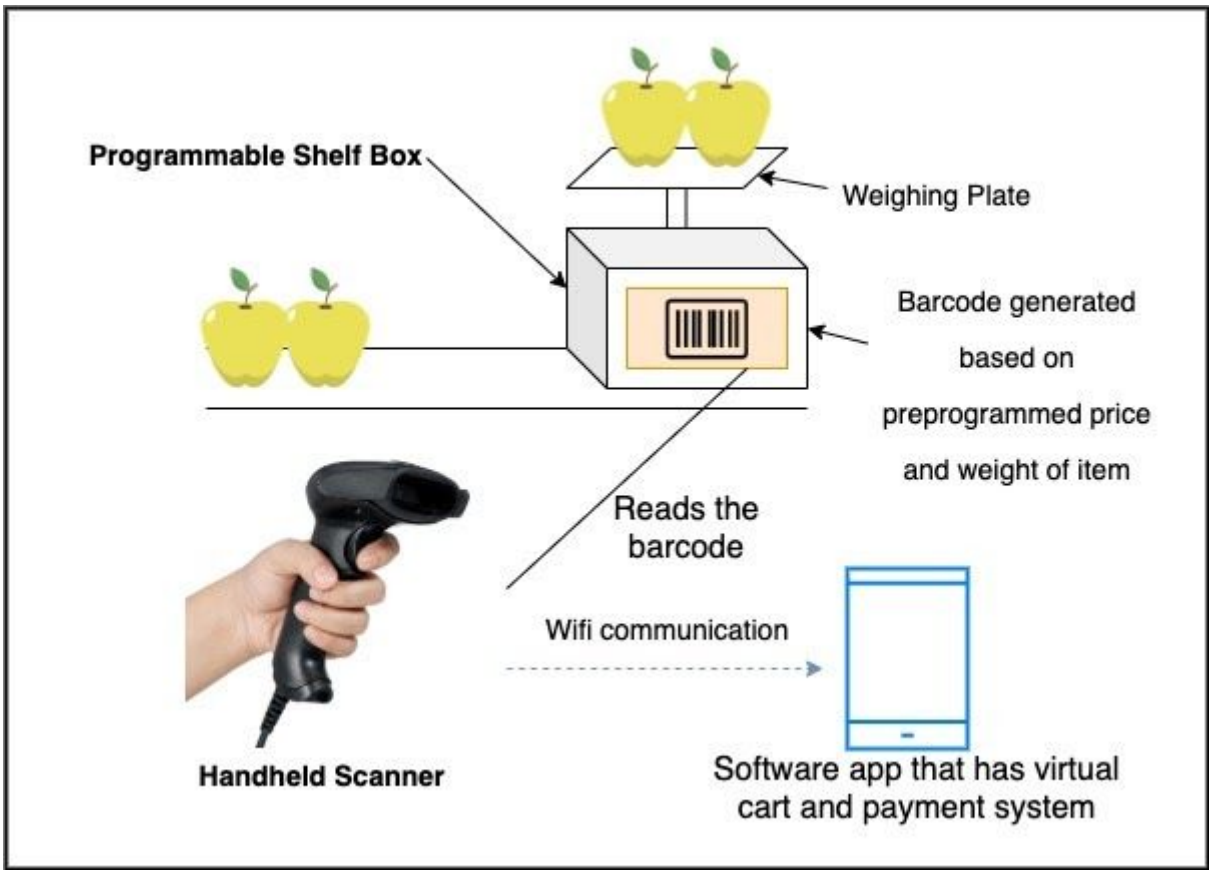

Figure 3. Visual Aid of Project Design

1.5 Updated Block Diagram

Our Block diagram is split up into 3 parts. The first is a programmable shelf box. This will be used for items such as fruits and vegetables for which it is difficult to weigh and scan items. Our solution involves using a load sensor to be able to weigh the item. Once we have weighed it, the Shelf box will calculate the cost and other information and display a barcode with the appropriate information. This can then be scanned by the handheld scanner to input the information for the quantity of the items you want to buy. The shelf is programmable so that we can change the item information depending on the item it is near.

The second part of our block diagram is the handheld scanner. This is a detachable scanner which can be used to scan the barcodes of any item. It contains a WiFi chip to connect to the server of the store to retrieve item information from it. Every time an object is scanned, it looks up the price, and updates the list of items and total cost stored in it. For items such as fruits whose price is dependent on weight, it scans the barcode from the programmable shelf box to receive information about the item. The handheld scanner also interacts with the Software app using the WiFi chip to send information about the items to the app.

The software app is used so that customers can view the items they have scanned and the total cost of all the items. It can also be used for payment so that checkout is done quicker.

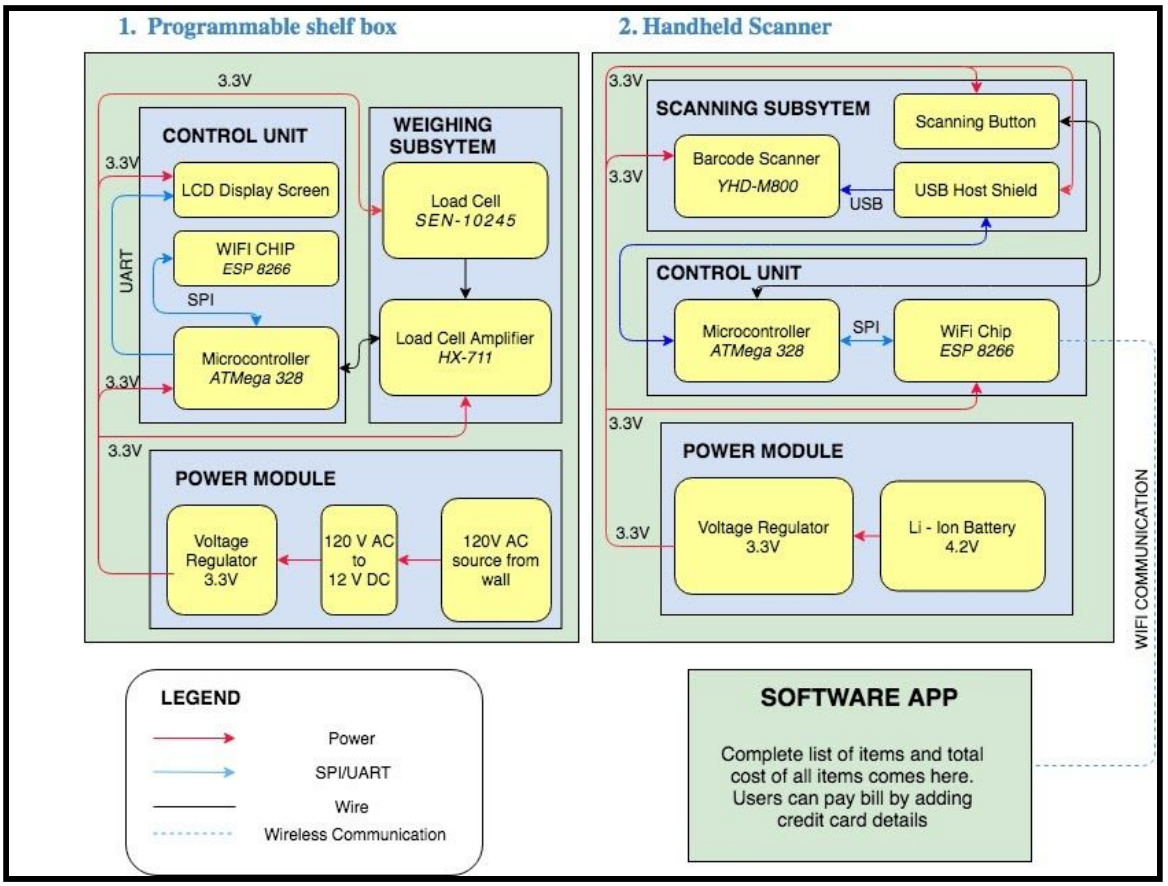

Figure 4. Block Diagram

2. Second Project Implementation

For our implementation, we focused on both hardware and software components which we thought were necessary for our project in relation to the high level requirements. The first was in regards to the syncing of the shelf box with the server. This required software/firmware implementation in which we tried to simulate the syncing of the server using the WiFi chip. We used Tinker Cad to complete the connection with an arduino and breadboard. This is similar to what we would have done with access to these components. Next, we implemented core components of our software app such as adding, deleting and viewing items that we have scanned as well as checkout options. The next part we implemented was the circuit schematics for the PCB. This is an integral component of our project and relates directly to the high level requirements. Lastly, we did an analysis on the feasibility of the project for large supermarkets and grocery stores in terms of cost since they are our primary customer targets. We wanted to ensure that this project was something that would be of interest to them as well. This was one of the feedback we got from the Design Review and we wanted to show the advantages of our approach and why it would work on a large scale.

Thus, we have implemented many different parts of our project which cover all the high level requirements as well as software, hardware and design aspects of the project. This would put us in a good position to complete the project when we get the chance.

2.1 Implementation Details and Analysis

Here is our implementation of the project components - hardware and software, that we found most important for development of our project. We focused on designing the pcb as we felt that was a key component of our project and could be done without the lab resources. We also ran some circuit simulations on tinkercad that would help us verify some circuit designs.

Implementation of software/firmware components

The ability to display on an lcd screen is critical to the functioning of our project. At different times, we have different information on the lcd. For example, when a customer is buying something a barcode is generated on the screen whereas when there is no one buying anything at the LCD box it just displays the price of the item and the time of arrival of the stock. In addition to this, we have a wifi interface whereby our lcd box needs to sync with an inventory management system and update its contents. The Pin assignments for this circuit can be found in the code on appendix B of our project. [3]
To test this circuit out we used an arduino uno, esp8266 wifi chip and the Liquidcrystal LCD display. We simulated this circuit on a tinkercad.

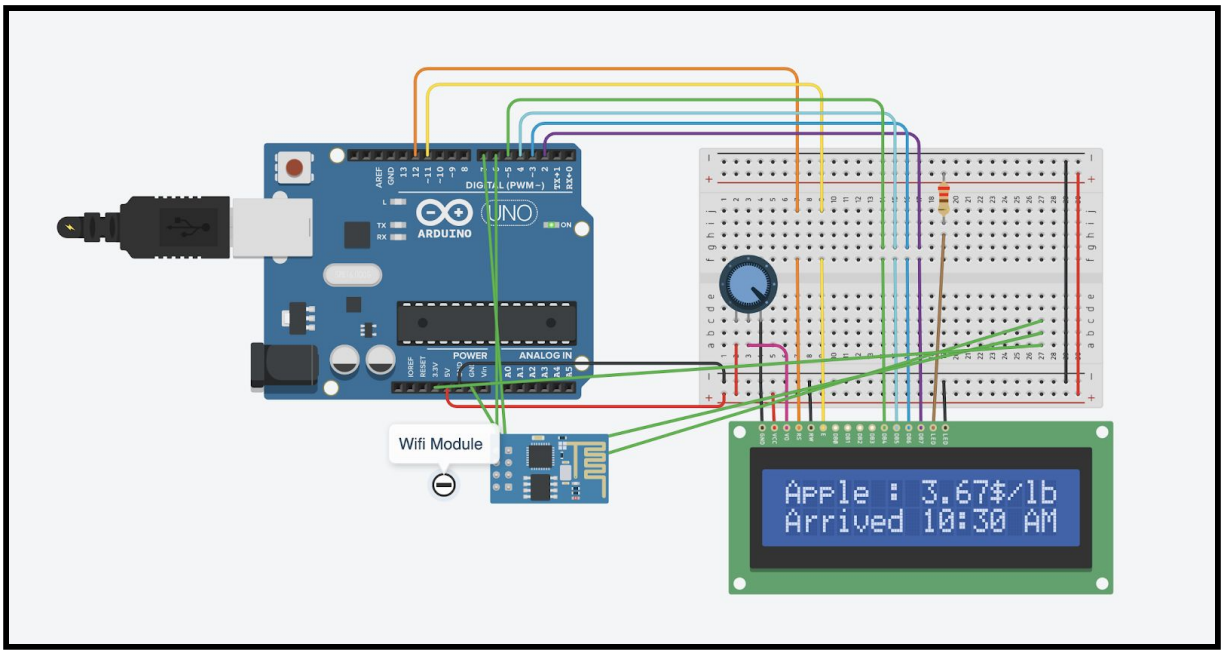

Figure 5. Circuit simulation

The code for this circuit is attached in the appendix. For testing purposes, every 5 minutes the contents of the lcd display will update whatever is in our server. To do this we connect the ESP8266 to the same access point as our localhost node js server. Our localhost node js server connects to a simple mock database. The Arduino uno has an internal counter and every time the counter hits a multiple of 5, the arduino through pins 7 tells the WIFI module to send a GET request to the localhost node js server. The WIFI module then sends a GET request to our server, retrieves the data and through pin 6 sends the data to the arduino. The arduino then interfaces with the LCD to update this data. The code for displaying barcodes/text on the LCD display and setting up the node js server is attached in the appendix B section of our project. [4]

In addition, a key component of our project is the ability to checkout via a phone app. Hence we implemented a mock page for our phone app. Through this page the customer can easily add or remove items. There are add and delete options via which the customer can easily alter his/her cart via the app.There is also a checkout option via which the customer can pay for the items.

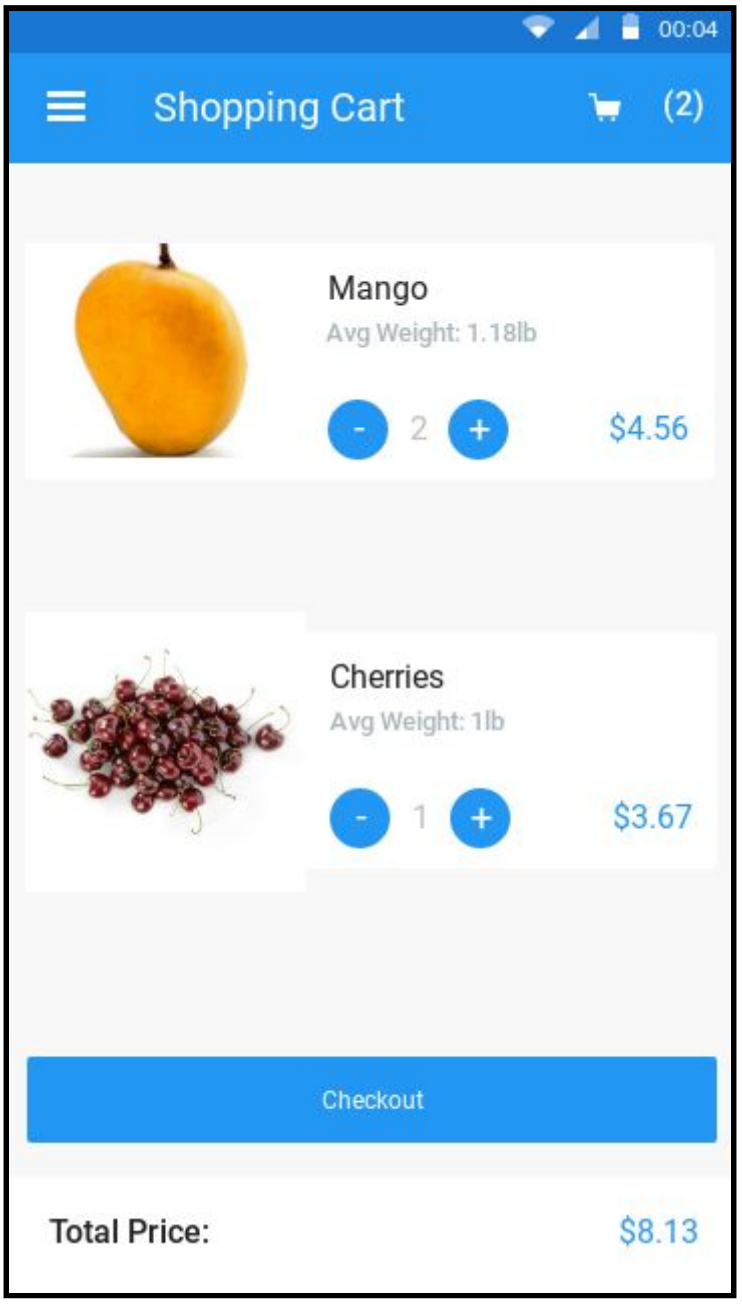

Figure 6. App mock page

Schematic for Programmable Shelf Box

Having done the PCB design for the first project, we decided to try and implement the design for the programmable shelf box. This consisted of multiple new parts such as the load cells, the load cell amplifier and the LCD screen, along with the microcontroller and the WIFI chip which were similar to last time. The challenge here was understanding how the pin assignments for each of these work, getting them all to connect together correctly. The microcontroller is the brain of the circuit and helps communication between each part. It connects to the load cell amplifier using the Data In port. The load cells connect to the load

cell amplifier using a wheatstone bridge configuration. These are colored as RED, BLK, WHT, GRN and YLW and correspond to the following color coding of load cells:

- Red (Supply+ or VCC)
- Black (Supply or GND)
- White (Output)
- Green (Output-)
- Yellow (Shield)

The yellow connection is optional(Shield against outside EMI) and we decided not to complete this connection since we were not sure if it was needed or not. The LCD screen was connected using a parallel 4bit interface to the microcontroller and is used to display information about the product being weighed. This information is sent from the amplifier to the microcontroller and finally to the LCD screen. We also have a WIFI chip connected to the microcontroller using the SPI protocol. This is needed to sync the box with the online server.

This schematic is important since it shows us how the parts for the programmable shelf box are connected and how they communicate with each other. Finalizing these connections helped us understand each of these parts and would allow us to easily implement our project in practice. The schematic would help directly with the first two points of our high level requirement. First, it shows us how the WIFI chip is used to sync the online server and the microcontroller. Second, it shows us how the Load cells and amplifier communicate the weight information to the microcontroller which in turn communicates it to the LCD screen. Thus, this schematic is a fundamental part of the implementation of our project since it relates directly to the high level requirements.

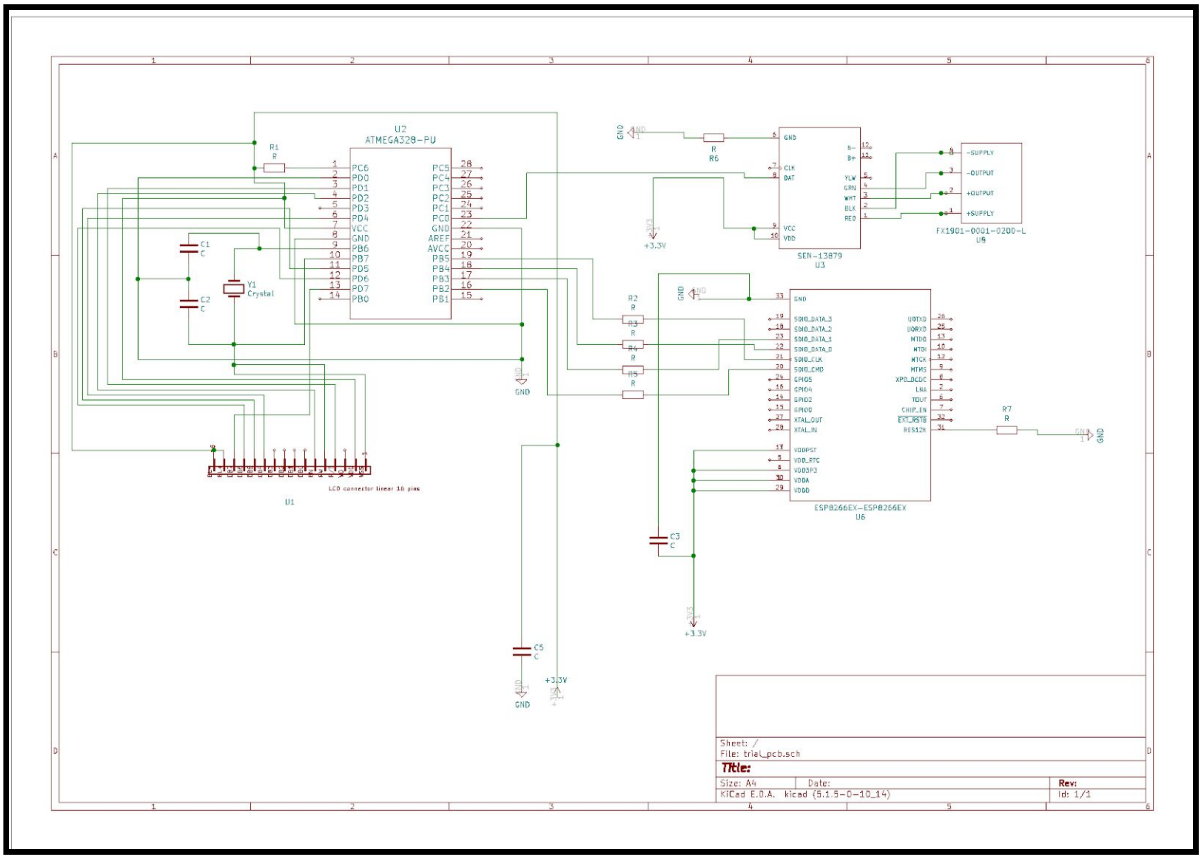

Figure 7. Circuit Schematic for Programmable Shelf Box

Schematic for Handheld Scanner

The second schematic we worked on was for the handheld scanner. It consisted of the microcontroller, the WiFi chip and the USB host shield (which will connect to the scanner module). The reason we used a USB host shield is because there is no direct way to connect the microcontroller and the scanner module we were working with. The microcontroller is connected to both the scanner module through the USB host shield (for signalling to start scanning) and the WiFi chip (to send scanned information to the mobile app) and is the main link of communication between the two components. We looked at the data sheets for all the 3 components and example projects to understand how to interface them together. For the microcontroller and the wifi connection, we used the SPI protocol whereas for the microcontroller and USB host shield, we used UART protocol.

This schematic implementation is necessary for making our entire handheld scanner to work as expected - start scanning items, read the scanned item and send information to the mobile application. Understanding the connections with the help of datasheets and making them on Kicad helped us finalize our circuits, and would make it easier if we ever get to implement our project in practice. This is an integral step for our project, and would help support our overall project goal and high level requirement 3 which is to allow the handheld scanner to communicate with the software application on the mobile. It is one of the first and most fundamental tasks we required to support our hardware implementation of the handheld scanner.

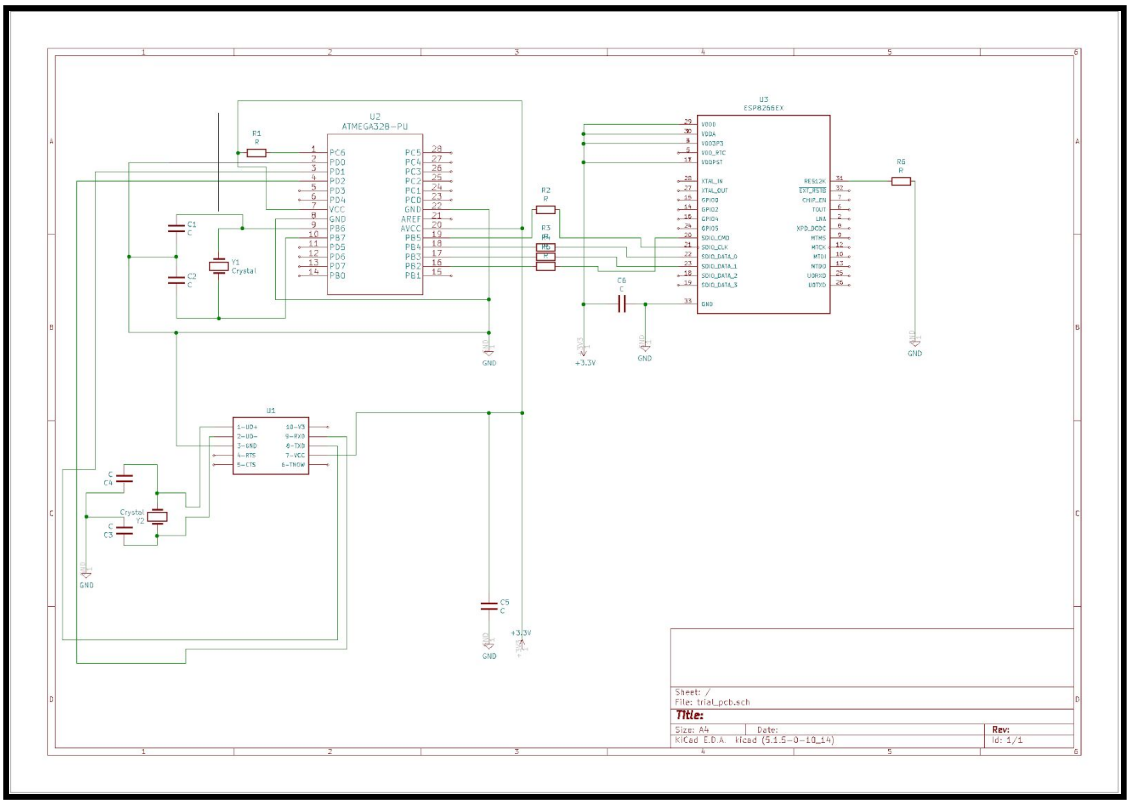

Figure 8. Circuit Schematic for Handheld Scanner

Feasibility for large scale implementation

Many projects fail because they are not feasible on a large scale, and may not always be beneficial for all parties using it. With our project, we attempt to see the tradeoffs financially between a more automated approach and having employees present. We know that our product makes shopping significantly easier for the average customer, through shorter lines, wait times and by giving them control of their shopping experience. However, this may not always be in the interest of the businesses who might lose money by using our product and never gain it back. We want to ensure that our products also make sense for the businesses as well, especially in a fiscal sense since this is a driving factor for them to purchase it.

Instead of comparing costs for employees versus our product, we are doing an analysis on the cost of self checkout lanes versus the cost of our product. This is because our idea will make it easier for the employees to do their daily work rather than replace them. It will be replacing the self-service checkout counters which are usually very expensive.

The cost to install a new self checkout kiosk is $30,000 on average, and a four-lane setup is $125,000 on average. [5]

The cost of individual parts for our product is $67.38 for each handheld scanner and 59.26$ for the programmable shelf box. The following table shows an analysis of the average one time cost for a typical grocery store.

Product	Cost/Unit	Units required	Total Cost
Handheld Scanner	67.38	360	24480

Programmable Shelf Box	59.26	300	17778
Total cost	129.64	-	42258
Self Service Kiosk	30000	8-15	240000-450000

Table 1. Cost analysis of self service kiosk vs our project

Explanation of calculations in the table:

Number of Handheld Scanners Required on Average

A supermarket averages around 20000 customers per week.[6] Assuming this is divided equally for each day of the week, and on each day the customers are split equally among the 8 hours of most traffic in the store, we can estimate that the store will have around 360 customers on average per hour. This brings the total cost for the handheld scanner to $24480.

Number of Programmable shelf boxes required on average:

The number of programmable shelf boxes will vary for each store and would depend a lot on the number of vegetables, fruits and other such items that are sold without tags on them. On average there are around 200 varieties of fruits and vegetables available in an average grocery store. Assuming there are another 100 items that are untagged, such as fresh baked goods, meat, etc., we can assume an average of 300 units of the shelf boxes will be required.[7]

Number of Self Service Kiosks on Average

We couldn't find an exact data for the number of kiosks in an average grocery store, so we decided to estimate the number based on our knowledge and other helpful data available online. A typical grocery store can have anywhere from 8-15 self checkout kiosks, and given the $30000 cost per kiosk, we get the total cost to be in the range $240000-$450000.

The total cost for maintaining, manufacturing and deploying our product will definitely be more than our estimated cost, but the above calculations show that we have enough leeway to still be more cost efficient than the expensive self service kiosks.

Based on the calculations done above, we can conclude that grocery stores can greatly benefit from the use of our product. Not only is it significantly cheaper, it greatly improves customer experience due to the shortening of wait times and lines. This adds additional financial sense for the stores since it provides more room for additional customers. The feasibility of such a product on large scale was one of the points brought up during our design review and after discussion with our TA, we felt this was an important factor for the implementation of our project.

3. Second Project Conclusions

In conclusion, our team worked on implementing major software and hardware components necessary for our project implementation. We analyze our progress in the implementation of the project, things we couldn't test because of no access to the lab, uncertainties in our project (due to the nature of our project and due to the current scenario), safety and ethics and improvements for the future. We believe that our project was right on track and could've been implemented and expanded if we had the resources to work as a team in the lab.

3.1 Implementation Summary

For our project implementation, we tried to implement all the necessary project components that could be finished without hardware and lab access. For the overall project, we decided to look at the datasheets and make our board schematics, for both the programmable shelf box and the handheld scanner. It was hard to find Kicad libraries and footprints for some parts, and so it required a lot of research and time to figure out the correct schematics. This implementation helped us finalize our circuit design and connection, and would have made it easier to implement it in practice.
In terms of the software components of our project, we worked at implementing the code for barcode generation using binary search, as that is an integral part of our project and will be done in the programmable shelf box to be displayed on the screen. We also worked on making the layout of the software mobile application - to store a cart of items and provide a clickable button that leads to the payment page. This screen would be the main page that the user views for keeping track of items and hence we decided to focus on this aspect than the other possible screens. We also decided to simulate a circuit that prints data to an lcd screen based on communication with the ESP8266 chip.
Successes:

1. Board schematics for both components
2. Feasibility in large scale implementation analysis
3. Software App Cart screen layout
4. Circuit simulation for Wifi and LCD interfacing

3.2 Unknowns, Uncertainties, Testing Needed

We needed the lab environment to test our weighing component. This could potentially have errors that we have not taken into account like errors due to the mechanical design or due to the actual material we are using while creating our sub component. Huge errors in this system will cause errors in cost and that will lead to an unsuccessful project. Our tolerance analysis on a theoretical level explains how our project should be tolerant to manufacturer defects in load cells but to ensure this we need to test out our wheatstone bridge design in the lab.
Our weighing system should be tolerant to tolerate weights upto 10kg as bags of oranges or certain big fruits can weigh that much.
Current load cells have 5 major types of errors :

1. Non Linearity : the maximum difference between load cell output readings for repeated loadings under identical loading conditions (that is, either increasing the

load from zero or decreasing the load from the load cell's maximum rated capacity) and environmental conditions[8]

2. Hysteresis:The difference between load cell output readings for the same applied load, one reading obtained by increasing the load from minimum load and the other by decreasing the load from maximum load.[8]
3. Non-Repeatability:non-repeatability refers to the ability of a load cell to maintain a consistent output when an identical load is repeatedly applied.[8]
4. Creep: is the change in load cell output over time when a load remains on the cell for a long time[8]
5. Error due to fluctuating temperature.

We also need to ensure that we measure a weight of 10kg. We use the SEN-10245 load cell that has a much greater rated capacity of 50kgs. The load cell uses resistance wires whose value goes up when a weight is applied as the wire is stretched. It's hard to measure the small changes in resistance that are created by the strain gauge.Thus we have performed an analysis on how our design choice ensures that these load cells fulfill the requirement of the project.

Now we need to measure weights upto 10kgs.

A 10kg weight on a strain gauge of 1.7 m by 2mm by 2mm causes the length to change by 0.04m
Strain = ($\Delta L/L$) = 0.04/1.7 = 0.0235 ε

The strain gauge has a gauge factor of 2 meaning the change in electrical resistance will only be 0.047 ohm. To solve this we would need to set up the load cells in a wheatstone bridge network to measure even minor fluctuations in resistance. In addition, we would need to add in an amplifier circuit to measure the voltage change produced by the resistance. We would need a lab environment to test out our weighing subsystem and verify that the wheatstone bridge network along with the amplifier produces a change in voltage that eventually shows the correct weight of the object. We need the appropriate wires, testing equipment and mechanical tools to actually verify this part.

The lab equipment would also be needed to check for the power supply being provided by the voltage regulator. This is important since we are using a low drop voltage regulator which ensures the 3.7V from our battery does not go below 3.3V. We also would need to check the communication between various components which would require the parts as well as equipment from the lab. One of these would be that between the microcontroller and barcode scanner through the USB host shield. Since we are adding this additional component(USB host shield) to connect the two, we want to ensure the communication does not get interfered with.

3.3 Ethics and Safety

Our project has quite a few safety or ethical issues based on our current project design and vision. We will completely abide by the IEEE Code of Ethics [9], throughout the development of our project and its usage. We will follow guideline #1 in the IEEE Code of Ethics, and our main goal is the wellbeing of our users and of the environment, while making the shopping experience easier and more efficient for all. We will be transparent about our design and limitations, so that the project cannot harm people. We will make ethical design choices and hold the user's safety and well-being as the main priority while designing and implementing the project.

Through our scanner system, the grocery stores would get a lot of data about customer buying behavior that could be used by the store for different purposes. We will ensure that we do not store or allow misuse of any personal data of the users. We will make our system reliable to prevent data manipulation and information stealing through strong internal security measures and robust data transfer mechanisms. On the software application, we will allow users to pay online and we have to make sure that the payment data is securely stored and properly encrypted to prevent stealing. The user will also be given preference on whether they want to store that information or enter it everytime. The financial details of customers will not be shared or used in any way.

According to point #3 in the IEEE Code of Ethics, we will be honest and transparent with the user about any data we collect and how it will be used. Our project has a hand held device that allows users to scan barcodes. For user safety, we will make sure that the device is safe and does not, in any situation, cause any harm to the user. The current within the device will be cut off through a fuse if any circuit errors are detected. This will ensure that the user cannot get an electric shock due to exposed parts or other issues. Following guideline #7, we will always be willing to improve and correct any errors and criticism we receive, to improve the product and make it more useful.

The Li-ion batteries in the handheld scanner can pose risks that can be very harmful for human life, due to overheating and overcharging. This can cause the scanner to catch fire, while in the hand of customers. To ensure that this does not happen, the employees will be trained on how to charge the scanners and will be given instructions for proper storage. While working with the batteries, we will be careful and closely follow the instruction manual to ensure our safety and those of the people around us.

3.4 Project Improvements

Using this project for inventory management at a large scale could be a great future development for this project. Whenever items are added the LCD box can store the updated count and whenever a customer weighs something the count can be reduced. When the amount of an item such as apples is low, the LCD box can automatically trigger an order for more apples from the supermarket chain's apple distributors.
Even though customers can quickly checkout items using this system, bagging the items remains an issue. As a project extension, we could have bags placed inside the machine under the weighing subsystem. A motor attached to the weighing panel subsystem will

remove the plates of the panel after the customer has finished weighing the items. The items will automatically fall into the bag and the customer can put the bag in their shopping cart.
A potential design alternative for our project would be to remove the hand held scanner as this is the costliest part of the project. Instead, The machine could put the items in a bag as suggested and then just attach a barcode sticker on the bag that can be scanned while checking out. This could potentially save on the costs of adding a handheld scanner to each cart.

4. Progress Made on First Project

Post the design review, we took all the feedback from the professor and our TA to improve the project details. We met with the machine shop to finalize our design, and looked at the different box options (waterproof and without) and picked one that would be best as a case for our wearable. We then worked on doing our board schematics and PCB design, as it was important for our project development. We got our PCB design approved and were working on finalizing our list of items to be ordered before the Covid-19 situation.

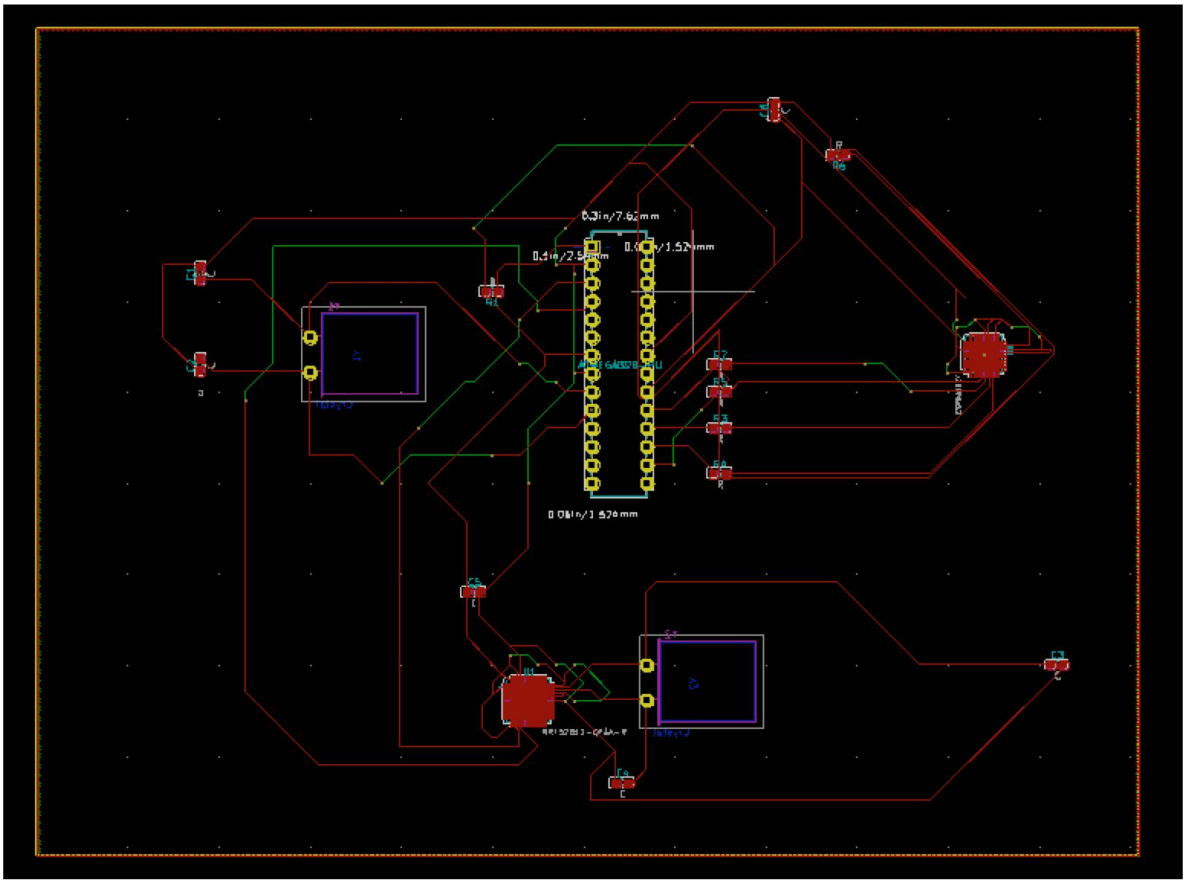

Figure 9. Approved PCB design for child wearable

In addition we also wrote a basic pseudocode to measure wifi rssi and calculate distance.

```
#include <SPI.h>
#include <WiFi.h>

//SSID of your network
char ssid[] = "yourNetwork";
//password of your WPA Network
char pass[] = "secretPassword";

void setup()
{
 WiFi.begin(ssid, pass);

  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("Couldn't get a wifi connection");
    while(true);
  }
  // if you are connected, print out info about the connection:
  else {
   // print the received signal strength:
  long rssi = WiFi.RSSI();
  int distance = double Dist = pow(10.0, (double)(Rssi) / 20.0);
  }
}

void loop () {}
```

Figure 10. Code for distance estimation[10]

References

[1]Why Self-Checkout Is and Has Always Been the Worst. (n.d.). Retrieved May 8, 2020, from https://gizmodo.com/why-self-checkout-is-and-has-always-been-the-worst-1833106695

[2]Tiffany, K. (2018, October 2). Wouldn't it be better if self-checkout just died? Retrieved from https://www.vox.com/the-goods/2018/10/2/17923050/self-checkout-amazon-walmart-automation-jobs-surveillance

[3]Antepher. (2019, March 18). ESP8266: HTTP GET Requests. Retrieved from https://techtutorialsx.com/2016/07/17/esp8266-http-get-requests/

[4]HelloWorld. (n.d.). Retrieved May 8, 2020, from https://www.arduino.cc/en/Tutorial/HelloWorld?from=Tutorial.LiquidCrystal
The next I see is in Feasibility for large scale implementation (if there's any before that, numbering will have to change here and in that section)

[5] Costs and Benefits of Installing Grocery Store Self-Checkout Kiosks. (n.d.). Retrieved May 8, 2020, from https://www.kompareit.com/business/kiosks-grocery-store-self-check-out.html

[6] - Supermarket Patronage: An Analysis of Customer Counts Among Outlets within a Geographic Area, Retrieved May 8, 2020 from https://core.ac.uk/download/pdf/6988363.pdf

[7] - How many different fruits and vegetables are there available in your typical major grocery store during the year? (n.d.). Retrieved May 6, 2020, from

https://fruitsandveggies.org/expert-advice/how-many-different-fruits-and-vegetables-are-there-available-in-your-typical-major-grocery-store-during-the-year/

[8] Strogonovs, R. (n.d.). Strain Gauge based weight sensor (load cell) - MORF - Coding And Engineering. Retrieved April 17, 2020, from https://morf.lv/strain-gauge-based-weight-sensor-load-cell

[9] IEEE Code of Ethics. (n.d.). Retrieved April 23, 2020, from https://www.ieee.org/about/corporate/governance/p7-8.html

[10] (n.d.). Retrieved May 8, 2020, from https://forum.arduino.cc/index.php?topic=550546.0

APPENDIX A - Requirements and Verification

Programmable Shelf Box

Microcontroller

Requirement	Verification
1. Should be able to receive load values from load cell amplifier	a) Connect the HX711 Load Cell Amplifier to the load cell using the color labeled pins b) Connect the load cell lifier to the microcontroller. The DAT pin is hooked up to an input pin in the microcontroller. c) Apply a small force on the load cell. Verify that the input pin on the microcontroller is receiving this value by printing to a terminal.
2. Should be able to send information to LED screen to be displayed over SPI protocol	a) Connect the microcontroller USB - SPI bridge. b) Use a terminal to send information from the microcontroller via SPI bridge. Verify that this is the same data getting echoed back.
3. Should be able to calculate price of item based on programmed information	a) Set preprogrammed price per kg value as 0.5$/kg. b) Send value of 10 to input pin on microcontroller. c) Calculation of price is Weight*price per kg. So, verify that the calculated price printed to the terminal is 5$. d) Repeat for different values of price per kg. The price per kg can be programmed depending on the item being measured.

Table 2.

LCD Screen

Requirement	Verification

1. LCD screen should be able to display information sent to it from microcontroller	a) Connect LED display to microcontroller using an USB-SPI bridge. b) Send data to the LED display and verify that this shows up on the display. Eg. Send “Hello” and verify that this is being displayed.

Table 3.

Load Cell and Amplifier

Requirement	Verification
1. Voltage change required for weight measurement should be detected in the wheatstone bridge network	a) Assemble the strain gauge sensors in the wheatstone bridge circuit shown in figure 3. b) Apply no weight. Measure the v output. It should be 0 c) Apply a 10kg weight on the strain gauge. d) There should be a voltage drop of a few mv across the wheatstone bridge network
2. The amplify the voltage difference generated in the wheatstone bridge.	a) Connect the wheatstone bridge from the first test to the op-amp circuit and set up the circuit as shown in figure 3 b) R1=R3 = 1Mohm R2=R4= 100kohm The gain should be Av = R1/R2 =10
3. Measure a 10kg weight on the scale with an error rate of < 5%.	a) Apply an excitation voltage of 10V b) Multiply the load cell amplifier’s reading * 50 which is the full scale weight capacity of the load cell. c) Divide the value in part b by 1mV and multiply it into 10 which is the excitation voltage. d) The output should be 10Kg +/- 0.5Kg

Table 4.

Handheld Scanner

Microcontroller

Requirement	Verification

1. Should be able to send information regarding price to the WiFi chip.	a) Connect the microcontroller to the WiFi chip using the SPI protocol. b) Send dummy data to the WiFi chip through the MOSI port. Eg. “Apple 5$” c) Print input from MOSI port in WiFi chip to terminal and verify that it matches the data sent.
2. Should be able to receive data from the USB host shield which is connected to the barcode scanner module.	a) Connect the microcontroller to the USB host shield using the SPI protocol. b) Connect USB host shield to an input source, Eg. Computer. c) Send dummy data value from USB host shield to microcontroller through input source Eg. “12345”. Print data received on microcontroller input to the terminal and verify that the printed value matches data sent.
3. Should be able to indicate to the barcode scanner to start scanning while the scanning button is pressed.	a) Connect button to microcontroller which is connected to the USB host shield. The USB host is connected to an output source such as a computer. b) Press the scanning button and send the signal received by the microcontroller(high signal) to the USB host shield. c) Print value received by host shield and verify that it is a high signal, indicating the scanner to start scanning.

Table 5.

Li-Ion battery

Requirement	Verification
1. Should output at least 3.7V	a) Connect battery terminals to positive and negative terminals of multimeter b) Ensure multimeter output is at least 3.7 volts.
2. Should store 500mAh amount of charge	a) Connect positive terminal to a voltage source of 3.7V and negative

	terminal to ground b) Discharge battery at a rate of 100mAh for 4.5hours c) Use a multimeter to ensure battery voltage is still 3.7V. This means that it has at least 50mAh amount of charge since the battery still shows the correct amount of voltage.

Table 6.

Barcode Scanner and USB Host Shield

Requirement	Verification
1. Should be able to scan a barcode and decipher the price of the item with no more than 4% error.	a) Generate a barcode for a given price (decimal number). b) Use the scanning module to scan the barcode. c) Connect the YHD-M800 to a computer using the USB port. d) Display the decoded price on the computer screen and calculate: Error% = (Original Price - Decoded price)/100 Ensure that the error percentage is less than 4%.
2. Should be able to communicate with the USB host shield and start scanning when input is sent.	a) Connect the barcode scanner to a computer using USB. b) Send a high signal(ie, 1) to the barcode scanner and verify that the barcode scanner attempts to scan.
3. Should be able to communicate with the microcontroller about scanned information.	a) Connect the barcode scanner to a computer using USB. b) Scan a barcode for items whose data we know beforehand, Eg. “Apples” and send to the computer through USB. c) Print input onto the terminal and verify that it matches what we scanned.

Table 7.

Button

Requirement	Verification

1. Should be able to send signals to the microcontroller when pressed.	a) Connect button to the microcontroller. b) Press button, which sends a high signal(ie, 1) to the microcontroller. c) Print input received by button from microcontroller onto the terminal and verify that it is a high signal.

Table 8.

WiFi chip

Requirement	Verification
1. Should be able to communicate with the microcontroller and receive price information of the scanned product from it.	a) Connect microcontroller and WiFi chip via the SPI protocol. b) Send dummy data from microcontroller to WiFi chip, Eg. ”Apples 5$” c) Print input received by WiFi chip onto terminal. Verify that it matches data sent.
2. Should be able to communicate with the software app and send item information to it.	a) Connect WiFi chip to dummy phone app. b) Send dummy data from WiFi chip to phone. c) Verify that the data received on the app is what is sent from the WiFi chip.

Table 9.

Common functionality

Voltage Regulator

Requirement	Verification
1. The 3.3V regulator should be able to output 3.3V +/- 2% when connected to the 3.7V battery.	a) Connect voltage regulator to a power supply of 3.7V. b) Connect output of voltage regulator to an oscilloscope to verify 3.3V output.
2. Maintains thermal stability below 60°C.	a) Connect battery to resistor. Voltage regulator is connected to GND and VDD across the battery. b) The IR Thermometer is used to

	ensure temperature is below 60°C.

Table 9.

Software Application

Requirement	Verification
1. Allow the user to delete an item from the cart.	a) Make a cart on the app and add 3 hard coded items in it. b) Press the delete option for a particular item on the cart. c) Ensure that the deleted item does not show up in the user's cart anymore and the remaining items stay intact.
2. It should display the correct bill for the user based on all the items in the cart.	a) Add 10 items to the cart with an associated price for each. b) Press "Checkout" to see the total bill for the user. c) Check if the bill calculated for the accounted items is correct and only uses the items presently in the cart.
3. Should let the user make a payment using their credit card for the correct amount securely.	a) Create a hard coded cart for the user. b) Go to "Checkout" and then "Make a Payment". c) Make a payment using a credit card to a known bank account. d) Check in the bank account to see if the money has been deposited.

Table 10.

APPENDIX B - Code

Printing barcode to screen code

```
#include <serialGLCD.h>
#include <ctype.h>
serialGLCD lcd;
char alphabet[]="0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"; // alphabet we need for our code.
// in barcode generation code W=Wide line and N= Narrow line.
// Each alphabet has unique code. W=1 N=0; using this the numbers can be generated

int  intcode[] = {52,289,97,352,49,304,112,37,292,100,265,73,328,25,280,88,13,268,76,28,259,67,322,19,274,82,7,262,70,22,385,193,448,145,400,208,133,388};
int xc=1;

void intr(int c){
  char s=1;
  for(int i=0;i<9;i++){
    if(c&256){
      if(s){
      lcd.drawLine(xc, 0, xc, 64, 0);
      delay(7);
      lcd.drawLine(xc+1, 0, xc+1, 64, 0);
      delay(6); // drawline code taken online
      }
      xc=xc+2;
    }
    else{
      if(state){
        lcd.drawLine(xc, 0, xc, 64, 0);
        delay(7);
      }
      xc++;
    }
  }

}
int binarysearch(char a,int s,int end){
  if(s>end){
    return -1;
  }
  mid = (s+end)/2;
  if(alphabet[mid]==toupper(a)){
    return intcode[mid];
  }
  else if(alphabet[mid]>toupper(a)){
    binarysearch(a,s,mid-1);
  }
  else{
    binarysearch(a,mid+1,end);
  }
}
```

Figure 11: Printing barcode to screen part 1

```
void str(char *s){
  int s =0;
  int end = strlen(s);

  for(int i=0;i<end;i++){
    int code = binarysearch(a,0,36);
    intr(code);
  }
}
// Setup script taken from https://forum.arduino.cc/index.php?topic=104464.0
void setup(){
  Serial.begin(115200);
  delay(1500);
  lcd.resetLCD();
  delay(250);
  unsigned char x,y;
  for(x = 0;x < 128;x++){
    for(y = 0;y < 64;y++){
      lcd.togglePixel(x, y, 1);
    }
  }
  str("*TEST*");
}
void loop(){

}
```

Figure 12: Printing barcode to screen part 2

Circuit simulation code

```
/*
  The circuit:
 * LCD RS pin to digital pin 12
 * LCD Enable pin to digital pin 11
 * LCD D4 pin to digital pin 5
 * LCD D5 pin to digital pin 4
 * LCD D6 pin to digital pin 3
 * LCD D7 pin to digital pin 2
 * LCD R/W pin to ground
 * LCD VSS pin to ground
 * LCD VCC pin to 5V
 * 10K resistor:
 * ends to +5V and ground
 * wiper to LCD VO pin (pin 3)

 Library originally added 18 Apr 2008
 by David A. Mellis
 library modified 5 Jul 2009
 by Limor Fried (http://www.ladyada.net)
 example added 9 Jul 2009
 by Tom Igoe
 modified 22 Nov 2010
 by Tom Igoe

 CODE MODIFIED AND USED IN SMART SELF CHECKOUT CARD PROJECT.

 ESP8266 CODE REFERENCED FROM : https://techtutorialsx.com/2016/07/17/esp8266-http-get-requests/
 http://www.arduino.cc/en/Tutorial/LiquidCrystal
 */
// include the library code:
#include <LiquidCrystal.h>
#include <esp8266wifi.h>
#include <esp8266httpclient.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
const char* ssid = "yourNetworkName";
const char* password = "yourNetworkPassword";
void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("Apple : 3.67$/lb");
  lcd.setCursor(0, 1);
  lcd.print("Arrived 10:30 AM");

Serial.begin(115200);
```

Figure 13: WIFI and LCD display part 1

```
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
    delay(1000);

}

}
}
void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  //lcd.print(millis() / 1000);

if (WiFi.status() == WL_CONNECTED) { //Check WiFi connection status
HTTPClient http;  //Declare an object of class HTTPClient
http.begin("localhostdestination");  //Specify request destination
int httpCode = http.GET();

if (httpCode > 0) { //Check the returning code

String payload = http.getString();   //Get the request response payload
Serial.println(payload);                     //Print the response payload

}
http.end();   //Close connection

}
delay(500000);    //Send a request every 5 minutes
}
```

Figure 14: WIFI and LCD display code part 2