RFID Fridge

By

Jeffrey Lee

William Mercado

Yuanhao Wang

Final Report for ECE 445, Senior Design, Spring 2017

TA: Yuchen He

03 May 2017

Project No. 73

Abstract

While inventory tracking systems exist, they are mostly developed for large scale industrial uses and are too expensive for commercial usage. This project is an attempt to create a simple, cheap inventory tracking system more suitable for personal use. This paper will detail the entire design process for this project, from conception to implementation. Justifications for design choices are provided, as well as difficulties encountered in constructing the system and potential workarounds. Results show a general proof of concept for a simple, cheap, inventory tracking system.

Contents

1 Introduction

According to the NIAAA, Americans spent more than $163 billion on alcohol [1]. Indeed, for many college students, beer is an essential part of the college experience. However, living in an apartment with limited fridge space can cause confusion as to who bought what, and may result in some unscrupulous individuals stealing from others. We seek to solve this problem by creating an RFID tagging system so users can keep track of their inventory.

The goal of this project is to be able to alert users when their beer has been removed from the fridge by pushing an alert to their phone. We hope to use modern technology to provide users with peace of mind when using shared space to store their beer. The project will consist of a camera, an RFID reader and tags, and a microcontroller. The goal of our project is to provide a simple inventory system that can be easily fitted to most fridges and provide real time tracking of user inventory.

1.1 Objective

Beer is the most popular alcohol in America, commanding 47.7% of all store alcohol purchases in 2012 [2]. The average college student spends around $42 per month on alcohol, and male college students report averaging 9 drinks per week [3]. When living with roommates, situations may arise when beer is accidentally or purposely stole from one another. It may be awkward for roommates to confront one another without solid proof, so we seek to rectify that by providing a system in which users will exactly when their beer has been removed.

Currently, a company called Terso Solutions, Inc. makes RFID refrigerators, but they are not for commercial use [4]. Instead, it is more for use in hospitals, research labs, pharmacies, dental offices, and stockrooms where the RFID tag for specimens is extremely important. For commercial use, we do not need features such as remote temperature monitoring. Per their website, the main purpose of their refrigerators is to eliminate paper work and reduce costs through automation and smarter purchasing. We aim to have our product to not have these expensive features and make it affordable and consumer friendly.

2 Design

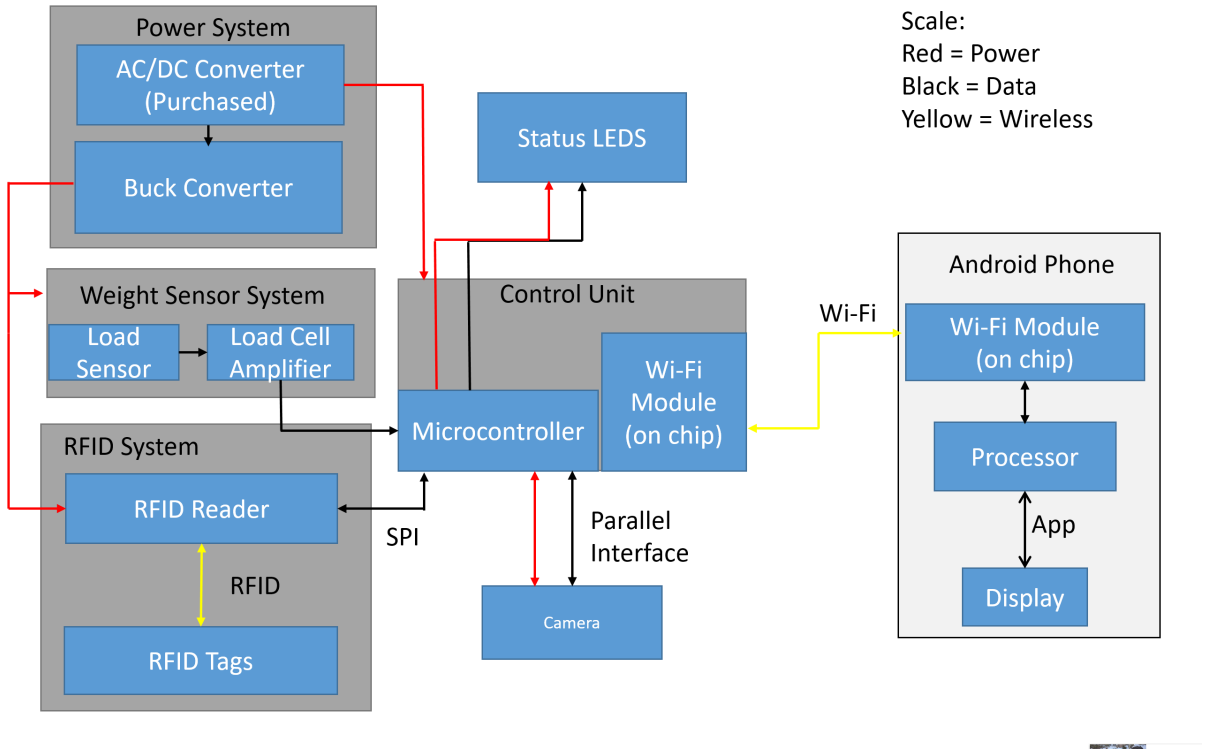

Figure 1: Block Diagram

Figure 1 shows the block diagram of our project. The RFID Beer Fridge can be broken up into 7 different components: a power system, RFID system, Weight Sensor System, Status LEDS, Microprocessor, Camera, and Software. The power system is going to be comprised of an AC/DC converter that is plugged in directly into the wall that takes in a 120V AC input and outputs a 5V DC output. It will then be attached to a buck converter that will take in the 5V DC input and output 3.3V DC. The power system will power up the RFID circuit, microcontroller and the camera. The RFID system will consist of an RFID reader, along with passive RFID tags that we will place on the beer. The camera will capture an image of a barcode from the beer labels and then use the built-in Wi-Fi Module in our microcontroller to send the image to our phone, where the processing will occur.The weight sensor system will be comprised of 2 load sensors and a load cell amplifier, which will convert the electric signals from the load sensors and convert it into weight. In addition, the sensors will be sandwiched between two shelves and the total weight will be recorded by the microcontroller. The system will detect whether beer is removed from the fridge by the microcontroller detecting a change in weight on the weight sensor. The status LEDS will be used as a confirmation that the RFID tag was scanned and the beer was placed on the shelf, in which the LEDS will turn green. If the beer were placed without scanning, the LEDs would turn red. In addition, if the beer were removed without scanning the RFID, the LEDS would turn red. If the RFID is scanned, however, the LED would turn green.The microprocessor we use must have an 8-bit parallel interface for the camera, an SPI interface for the RFID reader, GPIO pins for the weight sensor system, and a built in Wi-FI module able to be programmed in AP mode into a TCP Server.

Finally, on the software side, our device must be able to receive data from the microprocessor over the WiFi Module and store the total weight, RFID tag data, and barcode data after processing the barcode image. That is, the device must be able to take in an image from an outside source and then use a Barcode Scanner SDK with Android to process that image. After knowing what type of beer the barcode is, we will store the type of beer and quantity in an inventory. Finally, we will write a simple phone application that will allow the user to view his inventory and will send a notification whenever a beer is taken out incorrectly.

We made some changes for this block diagram as the semester progressed. Firstly, we struggled with the SPI communication between the RFID Reader and the MCU, so instead we used an Arduino for that integration. However, as that also failed, we attempted to mend this problem by communicating via Wi-Fi but unfortunately we did not have enough time to accomplish this. Therefore, we did not have any integration between the MCU and the RFID reader. Secondly, due to lack of microcontroller programming experience, we were unable to port the Arduino code for the RFID reader, the weight sensor, and Status LEDs over to TI's GNU compiler. So, we used an Arduino to control those three modules. Lastly, the barcode scanned by the camera was extremely unclear, so we used a phone camera instead to process the image.

2.1 Physical Design

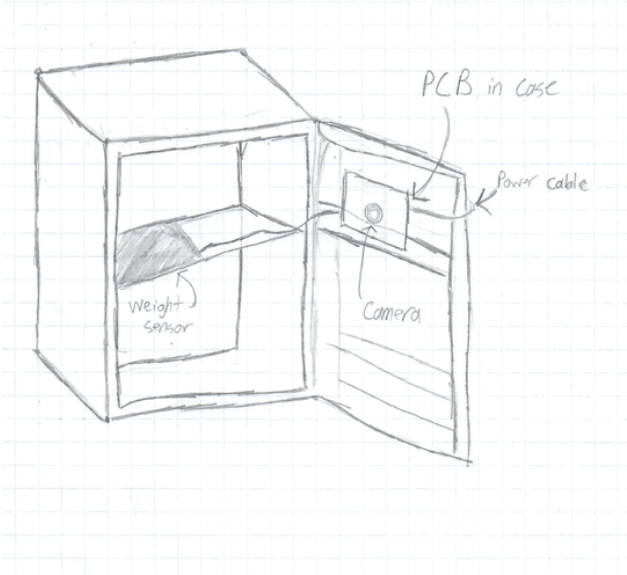

Figure 2: Physical Design for our RFID Fridge

A 3.3 cubic foot refrigerator will be used. It is large enough to hold the weight sensor on one of the shelves. In Figure 2 it is shown the location of the weight sensor and the location of the circuitry. The circuit is mounted in the interior of the fridge door.

2.2 Control Module

The control module handles the processing of the data from the RFID reader, camera, and weight scale through GPIOs and a built in Parallel Interface and can send this data over Wi-Fi to a user device. In addition, there will be a status bipolar LED that will display green if the beer is properly placed and red if it is improperly placed.

2.2.1 Microcontroller and Wi-Fi Module

The microcontroller, a Texas Instruments CC3200, processes all the data from the weight sensor module, RFID reader, and camera. The Launchpad we chose was the Arducam CC3200 UNO [5]. We chose this particular launchpad because it conveniently has a high speed 8 bit parallel interface for a camera with speeds up to 80 Mbps, along with 1 SPI port, and numerous GPIO ports. In addition, it has a build-in Wi-Fi module that can be programmed into Access Point mode which we can set up as a TCP server to send and receive data from an Android device. The microcontroller was chosen for its low cost and built-in Wi-Fi module with the "standard 802.11b/g/n radio, baseband, and MAC with a powerful crypto engine for fast, secure Internet connections with 256-bit encryption" [6]. In addition, we can program the Wi-Fi Module to have a static IP for easier socket communication. That is, we do not need to change the IP address of the user device every time one connects. In addition, this microcontroller is powered by a ARM Cortex-M4 core which has an 80MHz base clock so data transfers and processing can occur as quickly as possible.

However, due to lack of programming experience, we were unable to allow our MCU to communicate with the weight sensor module and the RFID reader, so instead we used an Arduino to communicate that system. Originally, we were going to have the Arduino communicate with the MCU via SPI so that the Arduino would be able to send data over to the MCU where it can be processed and sent via Wi-Fi to the Android Device. That is, we wanted to have the MCU act as a master device and the Arduino as a slave device. However, unfortunately we were unable to get this interface to work, as the SPI Chip Select Signal never got read by the Arduino.

2.2.2 Status LED

The purpose of the status LEDs is to alert the user when the process of placing a beer is complete or incomplete. We would use one 5mm Bi-Polar Red/Green LEDs that would change colors based on the polarity. The LED would be attached on the door and be powered by the GPIO pins of the microcontroller. The LED would turn green whenever a beer is tagged, the barcode scanned, and placed on the weight sensor. In addition, the LED would also turn green if a beer is removed correctly. That is, removed from the weight sensor and scanned with the RFID reader. In any other situation where the beer is placed or removed from the weight sensors improperly, the LED would turn red. In addition, the LED would be off if no action has been taken.

This system ultimately managed to work, but instead of interfacing with the Microcontroller, we needed to interface it with the Arduino so that it would interface well with the RFID reader and the weight sensor.

2.3 Camera Module

The purpose of the camera module is to take a picture of a barcode with a high resolution. The camera that we are using is the MT9D111 because it interfaces well with our microcontroller, in addition to its cheap cost. That is, it conveniently connects to our Launchpad using the 8-bit parallel interface at 80MBps. In addition, the MT9D111 has an auto focus mode, which is important in getting a high-resolution image of our barcode [7]. To scan a barcode, however, the resolution itself does not matter. Instead, we must take a picture such that the barcode is wide enough. That is, the smallest bar or space in the barcode must take at least be two pixels wide [8].

After testing this, however, we found that the auto focus mode did not work as intended. This will be further

described in the Requirements and Verifications section. The workaround we used for this was to use the Phone Camera to take the picture for barcode processing. In the future, however, we would like to attach a lens to this camera or get a higher quality camera.

2.4 Microcontroller, Camera, RFID and Android Integration

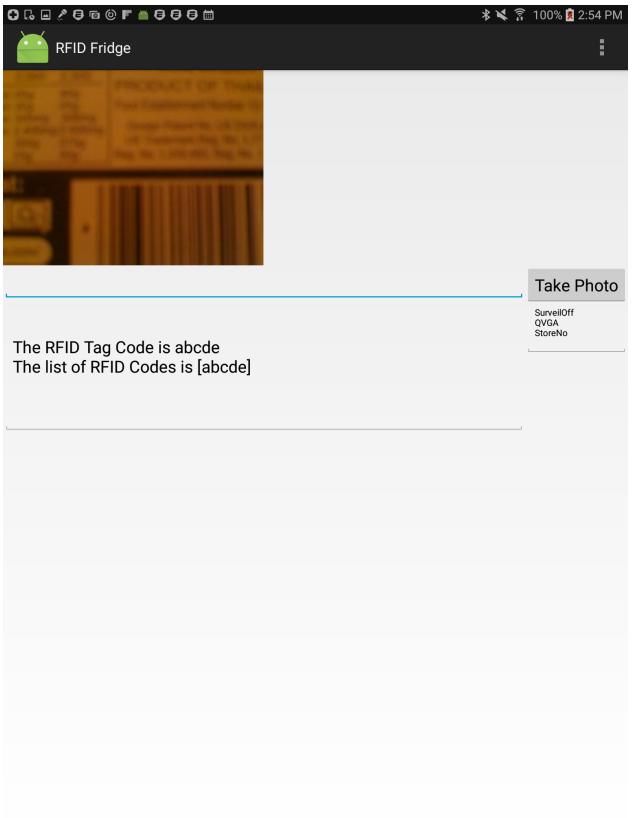

Figure 3: Screenshot of Microcontroller, Camera, RFID and Android Integration App

The general setup for our microcontroller, camera, RFID and android device integration is to have the microcontroller serve as the server and slave device that runs a TCP server and listens for incoming connections from the Android Device. The Android device then sends commands to the CC3200 and then the CC3200 responds to these commands by sending the requested data back to the Android Device [9].

Figure 3 is a screenshot of the app we created for the Microcontroller, Camera, RFID, and Android Device Integration. The user would first need to connect to the Access Point created by the CC3200, called "RFID Fridge. By doing this, the Android Device can connect to the TCP Server Socket created by the CC3200. Next, the CC3200 initiates the camera. The user then can hit the "Take Photo on the Android app, which will send a "get image size request to the MCU. The size depends on the resolution requested by the Android app. Next, the MCU will send the total image size to the Android app, which will then create an array based on that size. The Android will then automatically send a "Get Image Data request to the MCU, which then the MCU will send over the image. The image will then be displayed on the top left corner of the app and then processed by the barcode reader. Next, the app will send an RFID request to the MCU. The MCU will then turn on its chip select signal and wait for an RFID scan. After it is scanned, the data will be written into the RFID tag and it will be sent to the MCU and then over to the MCU. The RFID data will then be

displayed on the app.

Unfortunately, as we could not get the SPI interface working, we could not display actual RFID Data. So instead, we simulated the fact that we could send a string from the MCU to the Android App. In addition, the barcode picture could not be processed by the barcode processor, so we used a phone camera to take those pictures instead.

2.5 RFID Module

The RFID module consists of an RFID reader and passive RFID tags. The purpose of the RFID module is to tag the beer with one of the RFID tags and then scan the beer with the RFID reader, which then transmits the signal to the microprocessor that a bottle/can of beer was scanned and will be placed in the fridge.

2.5.1 RFID Tags

The RFID tags are placed on the items in the fridge and will store information. The type of tag we implemented is a passive tag. Passive tags do not require a power source because it will use the energy used from the RF waves from the RFID reader to power itself while also transferring information. This simplifies the power system. Tags must be able to hold at least 512 bytes so we can store enough information on the item. The RFID tags only directly interact with the reader through RF waves at the 13.553 13.567 MHz range. The frequency range is in the allowable range under FCC regulation and most passive, low range RFID systems operate in the range. RFID systems are generally cheaper than their UHF counterparts. In order to fulfill all of the requirements, we chose the Avery Dennison adhesive tags. It is a passive tag that opperates at the same frequency as the reader IC. The tag sticks to cans, which allows the user to easily apply and remove the tag from items.

2.5.2 RFID Reader

A 4 wire Serial Peripheral Interface (SPI) is used to communicate between reader and microcontroller. The frequency must be within range in according to FCC regulations. The RFID reader will run on 3.3V with 10% tolerance so it can use power directly from voltage regulator. TRF7960 from TI has an input range of 2.7V to 5.5V which can be powered from the buck converter or the AC/DC converter. The frequency used is 13.56 MHz. Unfortunately, the RFID Reader did not operate properly. The PCB was designed and printed. There was an issue with the soldering of the components with the board. In order to work around the issue, we used the Parallax RFID Reader/Writer. It has the an PCB antenna attached.

2.6 Weight Sensor Module

The purpose of the weight sensor is to place it on a shelf and then the microcontroller would be alerted whenever there was a beer placed or removed. The system is composed of two components: load sensor and a load amplifier. In addition, there would be a plate on top of the load sensors. We are using 2 load sensors in the middle of the plate and the four corners would have some stubs on the bottom so that the plate does not tip on one side. This plate would be on top of the already existing shelf of the refrigerator. The beer would be placed on the shelf and then the analog signal would go through a load amplifier into a digital signal, which would go into the microcontroller.

2.6.1 Load Sensor

The load sensor that we will use is the SEN-10245 made by Sparkfun. This particular load sensor can measure up to 50kgs, or 110 pounds. This particular type of load sensor uses a strain gauge load cell, where the force is being sensed by the deformation of strain gauges on the element. A strain gauge is a device that measures electrical resistance changes in response to force that is applied to the device. Each different type of gauge has a different sensitivity to strain and the output is generally measured by resistance. However, the base resistance of each gauge isnt that high, and most microcontrollers would not be able to detect the weight, so we must also use an amplifier to do so. For our design, we will use 2 load sensor, which can detect up to 220 pounds. We would choose sensors with a lower range of weight for more accuracy, but this was the cheapest. Although the range of the load sensor is so large, we calibrated at a more reasonable weight of 30 pounds.

2.6.2 Load Amplifier

For our load cell amplifier, we will use the HX711 IC that will allow us to easily read load cells to measure weight. The purpose of a load cell amplifier is to be able to more easily read the changes of the resistance of the load cell, and by calibrating with the microcontroller, we will be able to get very accurate weight measurements. This particular Load Amplifier is a 24-bit analog-to-digital converter designed for weight scales. By using a 24-bit ADC, we can achieve higher resolution for weight. That is, we can more easily detect small changes in weight. It will take in a power supply of 2.7-5.5V, so our PSU should be able to provide enough power for our load amplifier. The HX711 uses clock and data pins, so we should be able to connect it with the GPIO pins of our microcontroller. The microcontroller and the load cell amplifier both use a clock of 80Hz. The HX711 Load Cell Amplifier accepts 5 wires from our load sensor. Four of the wires are hooked up in a Wheatstone Bridge Formation. How a wheatstone bridge works is that it is a configuration of four resistors with a known applied voltage. The output voltage can be calculated by using Equation 1:

$$V_{out} = (\frac{R3}{R3 + R4} - \frac{R2}{R1 + R2})V_{in} \tag{1}$$

2.7 Power Module

The purpose of our power system is to output a 3.3V DC voltage from a 120V AC Wall Power source. To do this, we must first use an AC/DC transformer that will convert our 120V AC voltage into a 5V DC output. That will directly power the weight sensor. The other components in the circuit require 3.3V so a buck converter will be used to efficiently lower the DC voltage from 5V to 3.3V. The input range is important because the buck converter will take its input directly from AC/DC converter.

2.7.1 AC/DC Converter

First step is to take the 120VAC and convert it to 5VDC. The AC/DC converter we plan on purchasing is RAC04-05SC/W from Recom. It takes an input voltage of 120VAC and outputs regulated 5VDC with a typical tolerance of 2%. The part is capable of outputing enough current to power the whole project.

2.7.2 Buck Converter

The AC/DC converter we originally wanted to use was the LXDC3EP micro DC-DC. It was able to take the full range of output of the AC/DC converter as in input. The max current is 1 A which is more than the AC/DC converter. During the semester we decided to switch to the PAM2305AAB330 from Diodes Incorporated. The first choice had a package that was challenging to solder. As a result, the component burned up. The second choice has a package of SOT-23.

Figure 4: 24MHz Graph

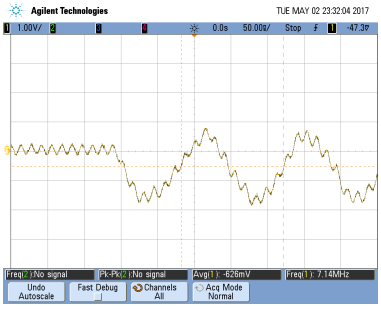

Figure 5: 7MHz Graph

3 Design Verification

3.1 Control Module

3.1.1 Microcontroller

Camera Speed Test

Our first requirement for our microcontroller was that the transfer speed between the 8bit parallel interface and the camera must be at least 80 MBps. This way, there is the least latency possible without increasing the cost of the overall system. To test this, we used an oscilloscope to probe one Data Out pin out of the 8 data out pins and then multiply that frequency by 8. We ran 10 trials as shown in Table 1 and found that the average was around 11.9 MHz. What's interesting is the wide range of values. That is, the standard deviation is around 5.28 MHz for one pin. One cause of this could be the the amount of frames that actually needed to be captured at that moment varied for each trial. How we actually tested this is that we put the Android app into Surveillance mode, in which it can capture a continuous stream of images. We realized, however, that depending on what is actually captured influences the frequency. Figure 4 shows the 24 MHz capture of the oscilloscope and Figure 5 shows the 7MHz capture of the oscilloscope.

Table 1: Frequency of Data-Out Pin

Trial	Measured Frequency (MHz)	Total Frequency(MHz)
1	12.85	102.8
2	24.04	192.32
3	8.15	65.2
4	11.9	95.2
5	16.54	132.32
6	7.14	57.12
7	13.67	109.36
8	8.19	65.52
9	7.56	60.48
10	8.96	71.68

GPIO Test

Because our load sensor uses a 24-bit Analog to Digital Converter (ADC), one of our main requirements was that the Microcontroller must be able to take in a 24 bit digital input via GPIO. How we tested this was that we wrote an Arduino code that could read the 24-bit digital input and display it in a 32-bit sign

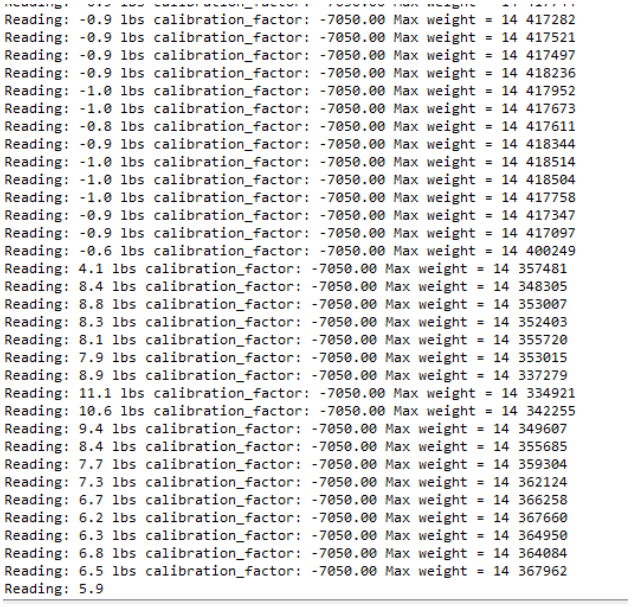

Figure 6: Screenshot of Arduino Serial Monitor

extended form. We decided to use the 32-bit sign extended form because that is the value that the Arduino is able to convert into weight. We also use the 80 MHz base clock on the Launchpad so that way it is able to take data at the rate of 333kHz. Figure 6 shows the Arduino Serial Monitor of this process, where the number at the very right of each row is the 32-bit sign extended version of the 24-bit GPIO input.

SPI Test

Since our RFID reader uses a 4 wire Serial Parallel Interface, one of our requirements for the MCU is that it must act as a master to to the RFID reader, which would act as a slave. That is, it must be the one to initiate communication with the RFID reader i.e. it must be able to send a chip select signal in order to send and receive data from the slave. However, this failed to work as the MCU failed to initiate a chip select signal to the RFID reader. The code we attempted to use was a modified version of the provided example code in the CC3200 SDK. However, the RFID reader was unable to retrieve this signal. In addition, because of limited time, we were unable to fully understand how SPI worked and merely based our code off the example code. We tried a workaround where we connected the RFID reader to the Arduino instead. However, the Arduino uses a different clock rate than the CC3200 and we were unable to adjust the clock rate on the MCU. In addition, the Arduino sent and received of the type “byte” whereas the MCU sent and received data of the type “unsigned long” so we had trouble convertering between the two. In the future, we would like to further address this issue and spend more time understanding SPI.

3.2 Camera Module

3.3 RFID Module

3.3.1 RFID Tags

Once the tag was read, it was possible to test the writing capabilities. The test required reading out the current value of the tag and instructing the RFID chip throught the MCU to write different values. After

attempting to write new data, the tag was read. The test was repeated 15 times. Each of the 15 times, the tag read the correct value.

3.4 Weight Sensor Module

The load cell was accurate in the weight range of 1 lb to 30 lb. In order to verify it, an item with a known weight was placed on the sensor. The calibration factor was changed until the output accuartly reflected the known weight. Once we got the calibration factor, we placed the weights on the sensor and measured the output voltage. Figure 7 shows the weight vs voltage plot of our calibration results.

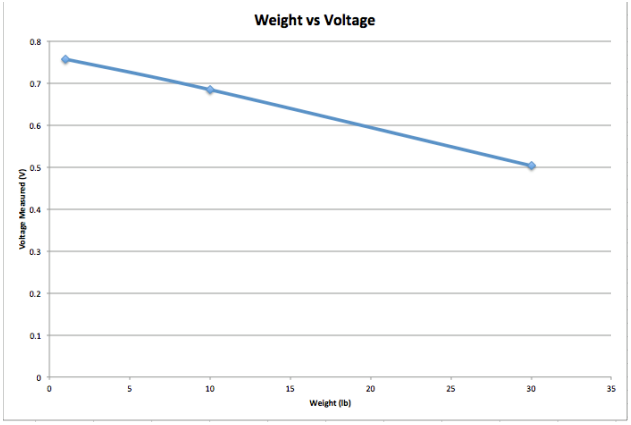

Figure 7: Weight Sensor Calibration Results

3.5 Power Module

3.5.1 AC/DC Converter

In order to test the AC/DC Converter, we soldered the converter to the PCB and powered it with a wall outlet. An oscilloscope was used to measure the output voltage. The output voltage was mesured at 5.05 V. The converter was left plugged for 5 hours to make sure the AC/DC converter still operated properly after in used for an extended amounted of time.

3.5.2 Buck Converter

A signal generator swept across the values of 3.3 V to 5.5 V as the input. An oscilloscope measured the output voltages. The output voltages were always within the 10% range.

4 Cost

4.1 Parts

Table 2: Parts Costs

Part	Manufacturer	Retail Cost ($)	Bulk Purchase Cost ($)	Actual Cost ($)
RFID Reader Module	TI	6.25	3.96	6.25
AC/DC Converter	Recom	15.30	12.23	15.30
Buck Converter	Diodes Incorporated	0.58	0.24	0.58
RFID Tags	Avery Dennison RFID	0.68	0.32	3.40
MCU	TI	11.63	11.63	11.63
2 MP Camera	Micron	18.99	18.99	18.99
Load Cell Amp	SparkFun Elctronics	9.95	9.95	9.95
Status LED	Lite-On Inc.	1.99	1.99	1.99
Load Sensor Strain Gauge	SparkFun Electronics	9.95	9.95	9.95
Shelf	N/A	2.00	2.00	2.00
PCB	ECEB Parts Shop	4.00	4.00	20.00
Total				100.04

4.2 Labor

Our labor costs are estimated to be $40/hour, 250 hours total, for three people. This brings our total estimate to $30,000.

Table 3: Labor Cost

Name	Hourly Rate($)	Hours	Total($)	Total x 2.5($)
Jeffrey Lee	40	250	10000	25000
Yuanhao Wang	40	250	10000	25000
William Mercado	40	250	10000	25000
Total			30000	75000

5 Conclusion

5.1 Accomplishments

Overall, for our project, we were able to get most if all of the individual components to work, with some integration between modules. That is, for our PCB, we successfully converted a 120 VAC to a 5V using AC/DC Converter and then back down to 3.3V using a Buck Converter. In addition, we were able to program our Wi-Fi Module into Access Point mode so that we could create a TCP Server to send and receive data via Wi-Fi. Subsequently, we are able to send a notification to the Android Device whenever an item is removed correctly or incorrectly. Our Android App is also able to have multiple users access its inventory and have a working inventory system that can store 50 items.

5.2 Uncertainties

The overall integration was a big flaw in our project. That is, even though the individual components worked, we were unable to accomplish integration between most of the parts. For example, we could not get the SPI interface to work because the chip select signal failed to function, so we could not get the RFID and MCU interface to work. We even used an Arduino to see if that would work, but ultimately that failed as well. Our final plan was to use a second Launchpad board to see if we could send RFID data through Wi-Fi but we did not have enough time to accomplish this. Another flaw was that because of our inexperience in microcontroller programming, we were unable to port Arduino code into TI CCS (Code Composer Studio) code as they used different compiliers. Thus, we ended up with errors that were ultimately undebuggable from copying the libraries over from the Arduino to CCS. Thus, our workaround for this was to use an Arduino for the RFID/LED/Weight Sensor System which we had hoped could communicate with the MCU via SPI. Also, the RFID portion of our PCB failed to function as well, as we used through-hole components, so there was too much inductance for the RFID chip to work. Thus, our workaround for this was to use a premade RFID system that could automatically read and write RFID tags using the Arduino. Our final uncertainty is that our barcodes taken by the camera were mostly unreadable because of the failure to auto-focus, so we used a phone camera instead to amend this.

5.3 Ethical considerations

The main safety we might encounter with our project is the camera and regulator circuity. These cameras will be placed in the fridge, where the moisture could lead to short-circuits and damage components. While the microcontroller itself doesnt need to be in the fridge, it will share connections with cameras, which are inside the fridge. We will need to ensure that our camera and microcontrollers and their connections are cased so that the moisture will not cause any safety hazards.

As engineers, we must be sure to credit properly the contribution of others as stated in #7 of the IEEE Code of Ethics [10]. We acknowledge that we will be building off of or incorporating many already existing components into our project, such as the ZXing library. We will ensure to always cite the authors wherever necessary to avoid accidentally representing any of those components as proprietary.

As engineers, we are called to improve the understanding of technology; its appropriate application, and potential consequences, as stated in #5 of the IEEE Code of Ethics [10]. We believe that our project will have a positive impact on the application of RFID technology, and will serve to provide a way for an owner

to protect his items. However, we are responsible for all information sent through our technology, and it is known that RFID tags can be compromised, and have their information extracted, deleted, or rewritten. In event that the tags are compromised, a user could lose track on items in the fridge, which could result in potential false alerts. However, we believe that the likelihood of a dedicated attack on our system is small, simply because of the low value of goods that our system is intended to deal with. Some users may attempt to use our project to facilitate in their storing of items that may be deemed illegal, or could be used to harm others. While we do not have a method to ensure that our project is used solely for legal purposes, we do not believe that limiting the functionality of our system is the right course of action, as they can easily use any number of alternatives. Overall, we believe that our project will have successfully used technology to improve the life of others.

5.4 Future work

A goal for future work would be to fully integrate the project. After gaining more experence programming microproccessors, integrating the RFID reader and weight sensor to the MCU should be feasible. Another area we would work on is the RFID system. A new PCB should be designed with surface mount components and with the matching netwrok on the board as well. Once all of the modules of the project work and are intergrated, the next step is to expand the scope of the project. The orignal idea was to have the system work with food as well. In order to do that more sensors would need to be included in order to detect items without barcodes. Adding more cameras inside the fridge can help with the detection of what items are currently present.

References

[1] "Alcohol Facts and Statistics," National Institutes of Health. [Online]. Available: https://www.niaaa.nih.gov/alcohol-health/overview-alcohol-consumption/alcohol-facts-and-statistics

[2] B. T. @bradrtuttle, "The Recent Evolution of How We Get Tipsy," Jun 2012. [Online]. Available: http://business.time.com/2012/06/25/the-recent-evolution-of-how-we-get-tipsy/

[3] "How Much Do College Students REALLY Drink?" Feb 2014. [Online]. Available: http://collegiatecook.com/2014/02/17/much-college-students-drink/

[4] "TS074 Large REFRIGERATOR." [Online]. Available: https://www.tersosolutions.com/solutions/rfid-refrigerators/ts074-rfid-refrigerator/

[5] "ArduCAM CC3200 UNO board: User Guide," Jul 2016. [Online]. Available: http://www.arducam.com/downloads/CC3200_UNO/ArduCAM_CC3200_UNO_DS.pdf

[6] "CC3200 SimpleLink Wi-Fi and Internet-of-Things Solution, a Single-Chip Wireless MCU," Datasheet, Texas Instruments, July 2013, revised Feb. 2015. [Online]. Available: http://www.ti.com/lit/ds/symlink/cc3200.pdf

[7] "1/3.2-Inch System-On-A-Chip (SOC) CMOS Digital Image Sensor," Datasheet, Micron, July 2013, revised Feb. 2015. [Online]. Available: http://www.ti.com/lit/ds/symlink/cc3200.pdf

[8] "Image Resolution and Barcode Reader," Mar 2013. [Online]. Available: http://www.bardecode.com/en1/image-resolution-and-barcode-reading/

[9] R. Chin, *Home Security System DIY PRO using Android and TI CC3200 SimpleLink*. CreateSpace Independent Publishing Platform, 2016.

[10] "IEEE IEEE Code of Ethics." [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html

Appendix A Requirement and Verification Tables

An appendix is a good place for the Requirement and Verification Table from your design review. Below is a starter table. Including these details here will help to avoid lengthy and tedious narrative descriptions in the main text, which may not be of immediate interest to your imagined audience of company managers and professionals. Any requirement that is not verified should be explained either in the main text or the appendix. Note that both the pagination and the numbering of figures, tables, and equations continues from main text to appendices.

Table 4: Requirements and Verification for the Buck Converter (5 points)

Requirements	Verification
1. Output must be 3.3 with 10% tolerance with 5±5% V input. (2.5)	1. Signal generator to create the min and max voltage input, and use oscilloscope to verify output. 2. Sweep the input from 3.3V to 5.5V.
2. Output current must be able to reach rated current max (1 A) as according to data sheet of buck converter (2.5)	Connect a resistive load with appropriate load that would lead to max current, verify with oscilloscope.

Table 5: Requirements and Verification for the RFID Tags (5 points)

Requirements	Verification
1. Must be able to write new RFID data after each scan.	1. Through attempt to scan new data to tag (place tag next to RFID reader). 2. Verify that data is either removed or written on microcontroller

Table 6: Requirements and Verification for the RFID Antenna (5 points)

Requirements	Verification
1. The antenna must transmit waves with a peak frequency in the 13.553- 13.567 MHz range (2)	1. Attach the matching network circuit to the spectrum analyzer. . 2. Use spectrum analyzer with the antenna and verify the peak frequency is in the 13.553-13.567 MHz range.
2. Must be able to transfer data at 848kbps(2)	1. Attach the matching network and antenna to a spectrum analyzer. 2. Verify that the bandwidth is 848 kbps.
3. Must be able to read data from reasonable distance, 10- 30 cm (1)	1. Program the reader to read tags. 2. Wave a tag from 10 cm and 30 cm. 3. Verify on microcontroller that data was read.

Table 7: Requirements and Verification for the Microcontroller (5 points)

Requirements	Verification
1. Must be able to receive data from the 8-bit parallel interface (camera)at 80 Mbps.(2)	1. Take a picture with the MT9D111 Camera. 2.Use an oscilloscope to probe the D0 pin on the camera. 3. Verify that the frequency is greater than 10Mhz. (8 bit parallel interface so multiply by 8 for final bitrate)
2. Must be able to take a 24 bit digital input via GPIO. (2)	1. Use the HX711 (ADC for our weight sensor module) and probe the voltage between the two weight sensors (around 0.768V). 2.Put this value in our AC power supply and verify that the MCU is reading a 24 bit binary input. 3. Repeat with decreasing voltages and verify that the MCU is reading a lower 24 bit binary number for a lower voltage. 4. The number will be stored as a 32 bit sign extended integer
3. The MCU must be able to perform as a master device for the RFID reader (slave) via SPI. That is, the MCU must be able to initiate all communications with the reader.(1)	1. Program the MCU to send a start condition to the RFID reader. 2. Verify that the RFID reader is able to receive data by scanning an RFID tag next to it. 3. Check on MCU that the RFID tag data is stored.

Table 8: Requirements and Verification for Wi-Fi Module (5 points)

Requirements	Verification
1. The built-in Wi-Fi subsystem must be able to communicate over IEEE 802.11b/g/n at 4.5Mbps.(2.5)	1. Setup the Wifi and register the IP Address on the MCU. 2. Use an app to determine the bitrate of the Wi-Fi bitrate.
2. Wi-Fi module must be able to transmit and receive at 40ft with at least 18.2 dBm transmit power and -95.7dBm receiver sensitivity.(2.5)	1. Place MCU 40 feet away from user device in an open-field environment. 2. Use an app to measure the RSSI.

Table 9: Requirements and Verification for Status LEDs (5 points)

Requirements	Verification
1. The LEDs must turn on with a drive current of 2,4, or 6 mA (the drive strength of the GPIO for MCU)(2.5)	Attach LED to breadboard with the pins programmed with a drive strength of 2, 4, or 6 mA and then see which option provides the brightest LED without burnout
2. Microcontroller must be able to change the polarity of its two ports in order to change LEDs from Red to Green(2.5)	Program the microcontroller to change polarities every few seconds and use a multimeter to test if the voltage is reversed

Table 10: Requirements and Verification for Weight Sensor System (5 points)

Requirements	Verification
1. Each plate must have a maximum weight of 30 pounds overall. Plot the weight vs voltage curve. (1)	1. Acquire three different weights of 1lb, 10lbs, and 30 lbs. 2. Place three weights on plate separately 3. Use a multimeter to measure the output voltage of the weight sensor system. 4. The 1lb weight should output the highest voltage, whereas the 30lbs should output the lowest voltage.
2. Each shelf must have a sensitivity of ± 0.5 pounds.(2)	1. Acquire three different weights of 0, 10, and 30 pounds. 2. Place three weights on the shelf separately. 3. Place a 0.6 pound weight and verify on the MCU that the weight changed.
3. The load amplifier must be able to take in an input of around 0.783 volts (The highest voltage with no weight) to the 0.616 volts (the voltage at 15lbs per load sensor) with a tolerance of $\pm 5\%$ and convert it to a digital signal, where it will be processed by the microcontroller and interpreted as weight(2)	1. Use an AC power source and output voltages of 0.783 to 0.616 with steps in between. 2. Connect the output of the AC power source into the ADC and connect the digital output of the ADC into the MCU via GPIO. 3. Record the value that is outputted by the ADC. 4. Plot the curve

Table 11: Requirements and Verification for Camera Module (3 points)

Requirements	Verification
The camera must be able to take a picture of a barcode from 1 to 5 inches away and the smallest space or bar must be at least 2 pixels wide.(1)	1. Program the camera to take pictures with the MCU. 2. After programming, toggle the camera to take a picture of a barcode that ideally has perfect resolution (take the barcode from a commercial product). 3. Transfer the image to the computer. 4. Verify that each the minimum size of the smallest bar or space is 2 pixels wide. 5.Repeat for 1, 3, and 5 inches away from camera
The camera must be able to capture an uncompressed QVGA image and convert it into a JPEG format and then send it to a MCU. (2)	1. Capture image with camera with MCU. 2. Use the MCU to send the picture over Wi-Fi to a user device (computer or phone). 3. Check whether the sent image is in a compressed JPEG format.

Table 12: Requirements and Verification for Data Processing (7 points)

Requirements	Verification
The software must be able to send received barcode images to the SDK and retrieve information. (4)	1. Take a clear, unobstructed picture of a barcode and ensure that existing applications can recognize it. 2. Use the same photo and send it to the application. 3. Verify that the application can recognize the image as a barcode. 4. Verify the accuracy of the information retrieved by comparing it with manual lookup.
User must be able to able to modify all inventory entries associated with each item (i.e. warning date and amount), including creating and deleting items in the inventory.(3)	1. Upon ensuring the software is able to recognize the image as barcode, attempt to have it search up the barcode information. 2. Verify the accuracy of the information retrieved by comparing it with manual lookup.

Table 13: Requirements and Verification for the Inventory(5 points)

Requirements	Verification
All users on the same Wi-Fi network must be able to access the inventory (1)	Have all group members attempt to access the inventory while on same network.
User must be able to manually modify the inventory (2)	1. Have user input a custom entry and verify that the inventory has been updated. 2. Have user change amount of item and verify that the inventory has been updated. 3. Have user remove item and verify it is deleted from inventory
Inventory must be able to store at least 50 items (2)	Manually enter 50 test points and verify that the inventory is able to successfully hold all points

Appendix B Tolerance Analysis

One important risk we must consider is the readability of the barcode pictures. One important aspect for unreadable barcodes is that the resolution is too low. However, for a barcode reader SDK on Android to work, resolution itself does not matter: what matters is how wide the barcode is, measured in pixels [12]. The number of pixels for the barcode depends on the number of characters encoded and the type of barcode. In addition, a barcode can only be decoded if every bar and space in the original barcode is preserved in the scanned image. That is, if the bars and spaces somehow blur into each other, then the barcode would be extremely difficult to decode. A barcode module is defined as the smallest width of a bar or a space in a barcode, and all other bars and spaces are merely multiples of a module. A perfect scan of a barcode would only need 1 pixel. However, this will most likely result in a barcode with mostly shades of grey. Thus, we must allow at least 2 pixels for each module. Additionally, different barcodes have different pixel widths, depending on the type of barcode being used and the number of characters being encoded in the barcode. For a UPC-A/EAN-13 barcode, which are generally found on beer, the pixel width is 166 Pixels. Next, to calculate the resolution that we need to use, we use Equation 2:

$$Resolution(dpi) = \frac{Pixel \quad width}{Width \quad of \quad a \quad barcode \quad in \quad inches} \tag{2}$$

So if we had a barcode width of 2 inches, the minimum resolution that we would need is shown in equation 3.

$$Resolution = \frac{166}{2} = 83ppi \tag{3}$$

Note that the scanner SDK normally has resolutions of 100, 200, etc ppi so if our Resolution were 83 ppi in this case, then we would use a 100 ppi resolution.

Another issue we may encounter is the low contrast of the barcode. That is, during when a user wants to scan in an item, the lighting in the room may be too low for the phone to process the image. This is because the barcode scanner must be able to differentiate between the light and dark elements of the symbol. If there is not enough contrast between these two barcode elements, then the barcode SDK might be unable to distinguish the barcode from the substrate (the black from the white) and the result will most likely end in a read failure. Figure 8 shows a couple of examples of unreadable barcodes.

Figure 8: Examples of Unreadable Barcodes [13]

Ensuring distinct and uniform barcode elements is the first step to preventing unreadable codes due to low contrast [13]. To do this, we can use lighting equipment that is tailored to produce the most uniform and

highest-contrast images of barcodes. If our beer has a glossy label, we can use diffused lighting. Figure 9 shows the difference between proper lighting on a glossy label. We can also use a dark field lighting to enhance the white-parts.

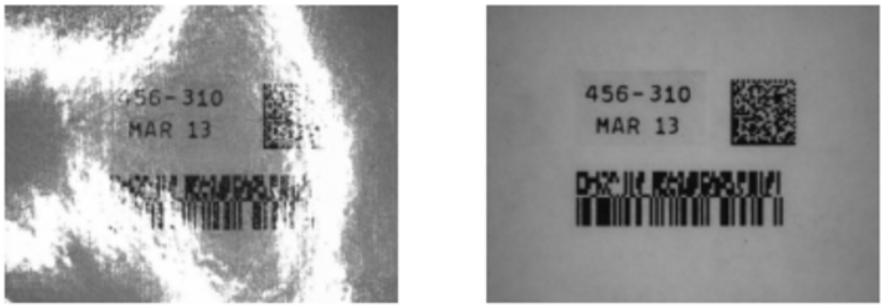

Figure 9: Effects of Proper Lighting on Glossy Label [13]

Another big issue we may encounter is with the read angle of a barcode. That is, a barcode sometimes may fail to be read due to the physical position of the camera relative to the barcode. Thus, it is important for our Android SDK that we use that it is able to auto-adjust the barcode to a right angle. That is, if it is slightly angled, the SDK must adjust the code so that it is level.