WeighTrack 2.0

The Harvard community has made this article openly available. **Please share** how this access benefits you. Your story matters

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Citable link</td>
<td><a href="http://nrs.harvard.edu/urn-3:HUL.InstRepos:14398526">http://nrs.harvard.edu/urn-3:HUL.InstRepos:14398526</a></td>
</tr>
<tr>
<td>Terms of Use</td>
<td>This article was downloaded from Harvard University’s DASH repository, and is made available under the terms and conditions applicable to Other Posted Material, as set forth at <a href="http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#LAA">http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#LAA</a></td>
</tr>
</tbody>
</table>
WeighTrack 2.0

by

Tyler Reed Kugler

ES 100 Senior Capstone Engineering Project Report submitted to the School of Engineering and Applied Sciences in partial fulfillment of the requirements for the degree of Bachelor of Science (S.B.) in Electrical Engineering at Harvard University Spring 2015

Submitted by

Tyler Reed Kugler
School of Engineering and Applied Sciences
April 2, 2015

Supervised and Evaluated by

Marco Rivero
Anderson Research Group Electrical Engineer

Evaluated by

Marko Lončar
Tiantsai Lin Professor of Electrical Engineering
## Table of Contents

List of Figures ......................................................................................................................... VI

List of Tables ............................................................................................................................. VIII

Abstract ........................................................................................................................................ IX

1 Introduction .......................................................................................................................... 1

1.1 Motivation ........................................................................................................................ 1

1.2 Consumer and Market Need ......................................................................................... 1

1.3 User Story - Car Parts ................................................................................................. 2

1.4 Target Audience ........................................................................................................... 3

1.5 Mission Statement ......................................................................................................... 3

2 Background Research ......................................................................................................... 4

2.1 Patent US20110166699 A1 ......................................................................................... 4

2.2 Patent WO2007006309 A2 ......................................................................................... 5

2.3 Patent EP1860409 B1 ................................................................................................. 5

2.4 ChemTracker ............................................................................................................... 5

2.5 Beverage Metrics - TILT ............................................................................................. 6

2.6 Capton BeverageTracker ............................................................................................. 6

2.7 ProBar Wireless ............................................................................................................ 6

2.8 Partender ..................................................................................................................... 7

2.9 WeighTrack 1.0 ............................................................................................................ 8

3 Design Goals ..................................................................................................................... 9

3.1 Performance Specifications ......................................................................................... 10
3.1.1 Analog Data Collection ................................................................. 10
3.1.2 Signal Processing ........................................................................ 10
3.1.3 Bottle Identification ................................................................. 10
3.1.4 Information Processing ............................................................. 11
3.1.5 Structure and Form Factor ....................................................... 11
3.1.6 User Interface ........................................................................ 12
3.1.7 Scalability .............................................................................. 12

3.2 Consumer Requirements ............................................................. 12

4 Design Approach ............................................................................ 14

4.1 Analog Data Collection .................................................................. 14
4.1.1 Resistive Flex Sensor ............................................................... 14
4.1.2 Load Cells ............................................................................. 15

4.2 Signal Processing .......................................................................... 16
4.2.1 Single-Stage Amplification ..................................................... 16
4.2.2 Multi-Stage Amplification ....................................................... 17
4.2.3 Shielding .............................................................................. 17
4.2.4 Filtering ............................................................................... 18

4.3 Identification and Data Management .......................................... 18
4.3.1 Barcode Scanners ................................................................. 19
4.3.2 RFID Readers ..................................................................... 19

4.4 Information Processing ............................................................... 21

4.5 Structure and Form Factor .......................................................... 21

4.6 User Interface ............................................................................. 21
4.6.1 Indicator LEDs .................................................................... 22
4.6.2 RGB LED Strip

4.7 Design Approach Summary

5 Design Details

5.1 Analog Data Collection

5.2 Signal Processing

5.2.1 Amplification

5.2.2 Filtering

5.3 Identification and Data Management

5.4 Information Processing

5.5 Structure and Form Factor

5.6 User Interface

6 Design Evolution

6.1 Analog Data Collection

6.2 Signal Processing

6.3 Identification and Data Management

6.4 Information Processing

6.5 Structure and Form Factor

6.6 User Interface

7 Evaluation

7.1 Overall System Performance

7.2 Analog Data Collection

7.3 Signal Processing

7.4 Identification and Data Management
List of Figures

Figure 2.1 Partender Image Processing Application ......................................................... 7
Figure 2.2 WeighTrack 1.0 ................................................................................................. 8
Figure 5.1 Load Cell Price Comparison .............................................................................. 24
Figure 5.2 Load Cell Characterization .............................................................................. 25
Figure 5.3 Quarter Wheatstone Bridge Configuration ......................................................... 26
Figure 5.5 Quarter Bridge Testing ....................................................................................... 28
Figure 5.6 Stage One Amplification Testing ....................................................................... 30
Figure 5.7 Difference Amplifier Configuration ................................................................. 31
Figure 5.8 Stage Two Amplification Testing ....................................................................... 33
Figure 5.9 First Stage Lowpass Filter .............................................................................. 35
Figure 5.10 Third Order Lowpass Filter System ................................................................. 36
Figure 5.11 RFID Antenna Construction Setup ................................................................. 38
Figure 5.12 RFID Coil Induction Testing .......................................................................... 40
Figure 5.13 RLC Meter Inductance Test Confirmation ..................................................... 41
Figure 5.14 ID-3LA Pinout Mapping ................................................................................ 42
Figure 5.15 Collection of RFID Antenna Prototypes ......................................................... 42
Figure 5.16 Top Shelf Support Testing ............................................................................. 46
Figure 5.17 RGB LED Control Schematic ....................................................................... 49
Figure 6.1 Quarter, Half, and Full Wheatstone Configurations .......................................... 51
Figure 6.2 Full Wheatstone Bridge Testing ....................................................................... 52
Figure 6.3 Pass-over and Platform Design Structures .................................................. 54
Figure 6.4 RFID Tag Vertical Range Mapping ................................................................ 59
Figure 6.5 RFID Tag Surface Range Mapping ............................................................... 60
Figure 7.1 WeighTrack 2.0 Final Performance Testing .................................................. 62
Figure 7.2 Noise Measurement Interference .................................................................. 65
Figure 9.1 WeighTrack 2.0 Complete ............................................................................ 70
Figure 0.1 CAD Model and Final Build ....................................................................... 78
Figure 0.1 WeighTrack 2.0 Signal Processing Schematic ............................................... 79
List of Tables

Table 4.1 Subsystem Design Selections................................................................. 23

Table 5.1 RFID Selection Criteria........................................................................... 37

Table 5.2 Microcontroller Selection Criteria.......................................................... 43

Table 6.1 RFID Coil Performance Results.............................................................. 54
Abstract

In the growing field of the Internet of Things, the interactions with our surrounding environments are growing smarter and more reliant on data collection and analytics. WeighTrack 2.0 adds intelligence to liquid inventories, allowing users to precisely monitor the amount of liquid content in a bottle at any given time, without directly interacting with the fluid. Through the implementation of RFID technology, load-cell networks, and a Wi-Fi-enabled micro-controller, WeighTrack 2.0 provides a platform for dynamically tracking liquid consumption in labs, restaurants, and households with Internet capabilities open to web developers.
1 Introduction

1.1 Motivation

The world makes extensive use of bottles to package liquid contents ranging from chemicals, to medicines, to foods, and everything in-between. Although bottles are an effective means of sealing a liquid from the outside world, they tell users very little about the contents within. Labels provide information about the type of content in a bottle, the creation date, and any relevant warnings. However, some of the most valuable information about a liquid is dynamic, including the amount that remains, the rate that it is being consumed, the time until expiration, and any information regarding recalls by the manufacturer.

1.2 Consumer and Market Need

In chemical labs, inventories can be highly disordered, making it difficult to track how full a bottle is or when an expiration date is approaching. This can be especially costly when critical experiments cannot be completed due to an expired or empty reagent bottle. In order to sort a chemical inventory, a hired lab technician must manually check each bottle for its remaining content and approaching expiration date. Additionally, they must make a decision whether to buy a new supply solely based on how much remains and when it is set to go bad. This is both costly and highly inefficient.

At home, individuals face a similar challenge with foods and beverages. Refrigerators have a tendency to get cluttered with both new and old contents,
making it difficult to determine how long items have sat untouched on the shelves. This makes it easy for expired bottles to take up valuable shelf space, or even worse, make it into the dining room. Untracked foods pose an even greater risk of being recalled without a consumer knowing.

A third major location where the static nature of bottles poses a challenge to users is at a bar or restaurant, where consumption rates vary for different liquors and spirits. For a business owner closely monitoring profit margins, tracking the amount of alcohol poured over the course of a night is a challenge. In some instances, outside consultants are hired to weigh the amount of alcohol at both the start and end of a shift to compare with the bill of sales. This allows an owner to monitor their bartender’s behavior and performance while on the job. Again, this requires extensive manual labor, and comes at a steep cost.

1.3 User Story – Car Parts

A Tennessee-based BMW original equipment manufacturer (OEM) illustrates the impact that mismanaged inventory can have on a company. As a car parts assembly plant that builds certified OEM parts, the factory must use all BMW original components throughout their process, including oils and fluids. During several instances, workers discovered expired fluids on inventory shelves, requiring the factory to halt production until replacement fluids arrived from Germany. With replacement shipments taking 2-3 days and the hiatus costing the factory
approximately $1 million per day, the company paid a heavy cost for the untracked inventory.*

1.4 Target Audience

The WeighTrack 2.0 target audience falls under three categories: Consumer Market, Business Market, and Research Market. The Consumer Market targets users that would use WeighTrack 2.0 in their home or other private facility. This includes applications in smart appliances, such as refrigerators or cabinets, which could allow for personal inventory tracking of groceries and other consumable goods. The Business Market targets users that would apply WeighTrack 2.0 to their restaurant or bar for dynamic inventory tracking of perishable goods or alcohol. Finally, the Research Market targets users that would utilize WeighTrack 2.0 in their lab or research facility for monitoring chemical use.

1.5 Mission Statement

Design a platform accessible to open web development that is capable of tracking and accurately reflecting changes in a dynamic liquid inventory. The device must not come in contact with inventory contents, and should universally support a wide array of diverse package types.

* User story provided by Maxwell Dworkin Securitas guard, Steve, who was employed by BMW for over 20 years
2 Background Research

Beyond manually sorting through an inventory and checking bottle content levels one by one, there are a limit number of solutions to creating an automated dynamic liquid inventory. Many of these solutions have been restricted by the lack of accessibility to Wi-Fi enabled microcontrollers that have only recently become available. As a result, many current approaches make use of proprietary wireless communication methods. This results in less secure data transmission and greater obstacles for product scaling. Below is an overview of relevant patents and products related to liquid content sensing and tracking.

2.1 Patent US20110166699 A1

This patent discusses liquid level sensing methods for bartenders using an inversion-detected spout. As a bottle is inverted, a built-in gyroscope and accelerometer sense the motion, and begin taking measurements of the liquid being poured. By comparing the pre- and post-pour measurements, the device then calculates the occupied volume of the bottle. The device then wirelessly transmits this data to a local computing system. While making claims of tying into the point of sales system within the register, the patent does not explain how this process would be achieved. In order for this device to measure liquid content, however, it is required that it comes in contact with the liquid as it passes through the spout.¹
2.2 Patent WO2007006309 A2

This patent examines measurements taken by a scale to calculate the volumetric status of a bottle. The author lays out a method for this approach that utilizes a scale, a control unit attached to the scale, some means of identifying a bottle, and a database to handle the information. The patent specifies use of a bar code scanner to read-in and identify each bottle. The database for this system has three categories: bottle ID, density of contents, and empty mass of bottle. The patent also discusses a valve stopper that seals the bottle and is controlled by the weighing system.2

2.3 Patent EP1860409 B1

This patent discusses an apparatus for weighing liquid in a pharmaceutical bottle. The primary application purpose for this patent is automated filling in a conveyer belt configuration. The apparatus utilizes a series of weighing devices along a belt that is fed bottles. This device uses precision scales to ensure 100% filling on each bottle.3

2.4 ChemTracker

ChemTracker is a Stanford-developed inventory control software that allows for classification and static monitoring of chemicals in a lab environment. The software allows for initial database inputs to be made regarding a lab material’s container type, location, department, owner, and expiration date. This allows users to search the database for expired chemicals, and locate them based on where they
were originally stored. This system does not consider any dynamic changes in bottles, including intentional or unintentional location changes.⁴

2.5 Beverage Metrics - TILT

Beverage Metrics’ Total Inventory Liquor Tracking (TILT) utilizes wireless sensors that connect to the throat of a liquor bottle. These utilize gyroscopes, accelerometers, and temperature sensors. Throughout the measurement process none of these sensors come in contact with the liquid poured. This measurement approach yields 94% accuracy for large liquor bottles. To communicate with the corresponding database, the hardware uses proprietary 919 MHz RFID to send both volume data and identification information. To match RFID tags to bottles, the system uses a barcode scanner on each bottle.⁵

2.6 Capton BeverageTracker

Capton BeverageTracker is a specialized liquid control spout that measures the amount of liquid that passes with each pour. Each spout connects through a proprietary wireless protocol to a BeverageTracker wireless data capture unit. This then ties to a local computer and point-of-sales software server on-site. BeverageTracker has found success through implementation in hotel bars and restaurants.⁶

2.7 ProBar Wireless

ProBar Wireless also uses a flow control spout to monitor and regulate liquid content. This system, however, provides pre-programmed pour amounts for precise
and automated distribution. ProBar then documents all pours made with a corresponding timestamp. This device is battery powered, and also makes use of a proprietary wireless communication system with a custom receiver that connects to a computer.\textsuperscript{7}

2.8 Partender

Partender is a more recent approach to liquor inventory control in bars and restaurants. Rather than implementing auxiliary equipment, Partender utilizes image processing through a smart phone mobile application. Partender requires manual data collection of each inventory bottle, but makes use of a phone's built-in camera for taking measurements. Rather than weighing bottles, Partender captures images of each individual bottle, and allows users to set approximate content levels based on the image, as shown in Figure 2.1 (below).

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{partender_image.png}
\caption{Partender Image Processing Application\textsuperscript{*}}
\end{figure}
Although Partender requires increased manual labor over other solutions, by eliminating additional hardware components, costs are significantly reduced.\textsuperscript{9}

### 2.9 WeighTrack 1.0

Prior to developing WeighTrack 2.0, an initial proof of concept prototype was created in the form of WeighTrack 1.0, shown in Figure 2.2 (below). This system utilized a single load cell, two pre-manufactured RFID readers, and an Arduino Uno with Wi-Fly shield to collect bottle data and pass it to a webpage. This system had major limitations, including a bottle capacity of 2 and high susceptibility to noise due to the RFID readers. As a result, the measurements taken by this system provided very little accuracy and the platform offered only proof of concept functionality.

![Figure 2.2 WeighTrack 1.0](image)
3 Design Goals

In order for WeighTrack 2.0 to provide effective dynamic inventory data for applications in the Consumer, Business, and Research Markets, its design needed to meet numerous functionality requirements. For the overall system, these design requirements included:

- Accurately capture the volume of liquid in a bottle
- Identify each bottle as unique
- Handle a wide variety of bottle types
- Allow for use in multiple environments, with various temperatures and noise sources
- Allow for manipulation of multiple bottles on the platform
- Protect electronics from light spills and contamination
- Provide user feedback for warnings
- Store bottle data and allow for integration with web development

Within the overall system requirements, the subsystems of WeighTrack were given further design specifications to meet. These subsystems include analog data collection, signal processing, bottle identification, information processing, structure and form factor, user interface, and scalability.
3.1 Performance Specifications

3.1.1 Analog Data Collection

The responsibility of the analog data collection was turning real-world analog information about the bottle into a signal that could be used to calculate the volume of the liquid. The performance specification for this process required:

- 95% accuracy for volume measurement
- A lower detection limit of 500mg (a single pill)
- Affordability
- Upper detection limit of 20 kg

3.1.2 Signal Processing

The signal processing subsystem's functional requirement was to carry the initial analog signal generated to the information-processing unit to be converted to a digital signal. The performance specification for this process required:

- Protection from noise generated by either external sources or other internal components
- Amplifying the analog signal to a usable 0-5v range
- Allow for analog calibration

3.1.3 Bottle Identification

The bottle identification subsystem had a functional requirement of easily and reliably identifying bottles being measured to allow for consistent database updating, as bottles were introduced to the system. The performance specification for this process required:
• Enable bottle placement over entire shelf surface
• Allow for reliable and unobtrusive identification
• Prevent measurement ambiguity between multiple bottles
• Allow for multiple bottle identification within system

3.1.4 Information Processing

The information processing subsystem had a functional requirement of processing the analog measurement and identification data into the database. This subsystem acted as the brain of the device and needed to provide feedback based on the information that the system received. The performance specification for this process required:

• Ability to handle sufficient analog inputs for measurement data collection
• Possess enough digital I/O pins to accept identification information and output system feedback to user
• Wi-Fi enablement for web integration and development
• Sufficient storage for managing inventory database

3.1.5 Structure and Form Factor

The structure and form factor of the system had the functional requirement of supporting the bottles, and encouraging proper usage of measurement and identification tools. The performance specification for this system required:

• Streamlined design to allow for wide range applications
• Dimensions capable of fitting within standard refrigerator shelf and on lab bench
• Preventing shelf deformation to ensure measurements remain in the linear regime
• Provide electronics with protection from spills

3.1.6 User Interface

The user interface of the system had the main functional requirement of providing user feedback to indicate errors or warnings relevant to the liquid inventory. The performance specifications for the user interface included:
• Providing reliable and effective communication of warnings and successful entries into inventory database

3.1.7 Scalability

An important consideration also applied to the overall system was the ability to scale. One of the largest drawbacks of current solution attempts outlined in section 2 is the use of proprietary wireless communication methods to transmit data. This opens access points for hackers, and requires both transmitters and receivers to be included with the systems for functionality.

3.2 Consumer Requirements

Consumer requirements were derived through direct observation and background research of the three primary target audiences. For dimensional constraints, measurements were taken of standard refrigerator shelves to guide form factor. Trips were also made to several of Harvard’s chemistry labs to understand current chemical storage and usage protocols. Online research was also conducted on the most common liquor inventory techniques in bars. Furthermore,
discussions with Sigma Aldrich’s supply chain management team aided in laying out specifications for compliance and bottle integration.

The engineering specifications for each subsystem of WeighTrack 2.0 were ultimately driven by the consumer requirements for usage. For the Analog Data Collection, the max weight specification was set by the overall weight of twenty 1-liter bottles of liquid on a shelf, corresponding to 20 kg. The lower detection threshold was set by the use case of detecting weight changes of one 500mg pill in a medicine bottle. This would allow for applications in medicine cabinets to serve as verification that a patient is correctly dosing their medicine consumption. The Signal Processing specification for noise resistance was set by the consumer requirements of shelf applications in various locations. Since each location presents a unique set of temperature and noise constraints, protecting against both these measures was important. The Bottle Identification specifications for handling multiple bottles and complex manipulations were derived from the observations of lab technicians and bar tenders removing and replacing bottles with relatively high frequency. Finally, the Structure and Form Factor specifications were driven by the dimensional constraints set forth by the shelf measurements taken for each target market.
4 Design Approach

The WeighTrack 2.0 design for capturing volumetric data was limited by the design requirement of restricting any measurements from coming in direct contact with the liquid. For this reason, the primary measurement technique options were limited to either image processing or weight-based. Due to the cluttered nature of shelves and the variability in opaqueness of bottles, image processing posed limited potential for this application. Since weight measurements provide a linear relationship to volume, and can be used independent of number of bottles on a shelf, this was isolated as the overall measurement technique selection. Additionally, WeighTrack 1.0 provided a proof of concept for this technique to be effective for implementation in WeighTrack 2.0. Below is an outline of the design approach for developing each subsystem of the WeighTrack 2.0.

4.1 Analog Data Collection

To interface the device with real-world analog measurements, a sensor needed to be selected to collect weight data. The primary sensor options included resistive flex sensors and load cells.

4.1.1 Resistive Flex Sensor

Pros: The resistive flex sensors offered a linear output with a dynamic range of 10kΩ to 20kΩ (greater than that of a load cell). This would allow for integration with a Wheatstone bridge to improve the dynamic range and provide a useable voltage output that could then be amplified.
Cons: The resistive flex components are standalone flexible strips that require mounting to some base cantilever. Attaching the flex sensors to custom machined cantilevers would be costly, time-consuming, and require extensive characterization and calibration between sensors due to the tolerance of 30% resistance on each part. Additionally, the dynamic range of flex sensors requires a large degree of deformation (flex) to achieve a wide output range. The deformation from placing a bottle atop a sensor would be relatively small compared to most applications.

4.1.2 Load Cells

Pros: Load Cells provide a linear output and are already integrated within a mountable cantilever. Some load cells include built-in Wheatstone bridges that allow for outputting a voltage due to minute resistance changes in the load cell. This integration would prevent having to work through the tedious process of balancing hand-made Wheatstone bridges. Load cells also provide various sensitivity options for selecting lower detection thresholds, or greater weight capacities.

Cons: Purchasing quality single load cells for prototyping can be extremely expensive. Additionally, since these cells have such a small dynamic resistive range, signals are prone to noise, and full balanced Wheatstone bridges are required to prevent temperature dependence in measurements.

Selection: Load Cells

Ultimately, load cells provided the most reliable and easy to implement approach for collecting the raw weight data from bottles. While the dynamic range of the resistive output is small compared to the flex sensors, the load cells are more
effective at measuring the smaller deformations seen in weight sensing applications. Additionally, the small output voltage range is capable of being amplified to the ultimate target range of 0-5v.

4.2 Signal Processing

The primary design goals for the signal processing of WeighTrack 2.0 included noise protection and voltage output of 0-5v. To achieve these requirements, the high-level design decisions involved choosing a means of amplification and filtering. The two amplification methods considered were single-stage amplification or multi-stage amplification. Furthermore, noise protection techniques were broken into shielding, filtering, or a combination of the two.

4.2.1 Single-Stage Amplification

Pros: Single-stage amplification offers the simplicity of only needing to use one op-amp or instrumentation amplifier. When considering scalability and cost efficiency, fewer components provide an advantage. Additionally, each stage of amplification can introduce its own noise to a system, which should be a consideration.

Cons: By attempting to amplify the mV-scale signal output from the load cell up to 5v, a gain on the order of 5000 is required. Even for some instrumentation amplifiers, this is a particularly high gain value. Additionally, if the load cell output has a dc offset, it would be extremely small and difficult to remove on the mV level. As a result, any DC offset would also be amplified by the single-stage amplification, making it difficult to achieve a true 0-5v range.
4.2.2 Multi-Stage Amplification

Pros: Multi-stage amplification allows for an initial gain-stage to transform the sensor signal from mV-level to V-level. This would allow for any DC offset in the signal to be identified and removed with a second stage of amplification. Additionally, the gain levels would be lower at each stage, allowing for more flexibility in choice of op-amps and instrumentation amplifiers. At each stage of amplification, signal filters could be integrated, allowing for multiple stages of filtering. Not only would this allow the signal to be protected along the full length of its pathway, it would also increase the amount of noise attenuation by 20dB/decade for each stage of lowpass filtering.

Cons: Multiple stages of amplification add complexity and additional components to the system. At each stage, the amplifiers introduce their own noise to the signal, although this should be far less significant than noise coming from outside sources.

Selection: Multi-Stage Amplification

The added complexity of a multi-stage amplifier is negligible in the context of this application. Additionally, the advantage of being able to calibrate the DC offset in a second stage of amplification is important for meeting the design requirement of including an analog calibration method.

4.2.3 Shielding

Pros: Provides a first line of defense for noise protection and addresses a wide range of noise sources.
**Cons:** If the noise source is in close proximity to the circuitry, shielding alone likely cannot provide an adequate means of fully protecting the circuit. This was the case in WeighTrack 1.0, which relied only on shielding for noise protection.

### 4.2.4 Filtering

**Pros:** Lowpass filters are easy to design and implement in a circuit, while also capable of effectively diminishing noise outside the cutoff frequency. Cascading filters at multiple stages of amplification can compound attenuations by 20dB/decade per stage.

**Cons:** Some lowpass filters do not provide a steep enough attenuation around the cutoff frequency, which is relevant if the noise is near the cutoff.

**Selection: Shielding and Filtering**

Since the primary noise source for WeighTrack 2.0 is from the RFID coils (greater than 100 kHz) and the signals received are on the human scale of ~10Hz, rapid attenuation is not a concern for this application. Additionally, by cascading multiple filters, an attenuation of 40dB/dec or even 60dB/dec can be achieved. Since shielding sensitive components doesn't provide any significant disadvantages, this method can be applied in addition to the filtering to fortify the noise protection in WeighTrack 2.0

### 4.3 Identification and Data Management

The primary design goal for the bottle identification system was to allow for uniquely identifying a bottle when it is introduced to the WeighTrack system. This allows for weight measurements and other warnings to be tied to each specific
bottle. The primary options for bottle identification include barcode scanners and radio frequency identification (RFID) readers.

4.3.1 Barcode Scanners

**Pros:** The biggest advantage of barcodes is their cost. The barcode tags are made of printed ink, which means the cost of production is very low. This cost also favors scaling to larger inventories. Due to the low cost, barcodes can be treated as disposable on each bottle. Barcode scanners are also highly accurate at reading tags when the item is properly aligned. Additionally, barcode scanners use light, which introduces little to no noise to the embedded electronics of WeighTrack 2.0.

**Cons:** The most significant disadvantages of barcodes are their durability and alignment requirements. Since barcodes are made of ink and paper labels, they can be easily damaged if they get wet or torn. Since barcodes use reflected light for reading, there must be a direct line of sight from the scanner to the tag. Additionally, the tag must be properly aligned for the scanner to read the code. This means that it must not only be in close proximity to the scanner, it must also have a flush orientation to the reader. An additional consideration for implementation of barcodes is security. Since the tags can be manufactured with any printer, barcodes are more easily forged than RFID tags.¹⁰

4.3.2 RFID Readers

**Pros:** RFID readers can read tags at a greater range than barcode scanners, allowing for design flexibility in user interfacing methods. RFID tags can also be read at any orientation, as long as they are within range of the reader. Since RFID tags are
sealed in plastic, they are both waterproof and highly durable. This durability allows RFID tags to be reused beyond the life of a chemical or liquor. RFID readers also read tags faster than barcode scanners, allowing for a more seamless interaction, as users replace and remove bottles from shelves. RFID tags also provide greater security, as ID codes can be encrypted, and physical tags are more difficult to forge.

**Cons:** The two biggest disadvantages of RFID technology are noise introduction and cost. In order for RFID readers to function, they produce a large RF field that excites nearby tags. While this can provide a large reading range, it also exposes any nearby electronics to RF noise at the reading frequency. RFID technology has not reached large-scale adoption by retailers largely due to the cost increase over barcodes. Although tag costs have dropped significantly over the past 10 years, they are still roughly $0.10 more expensive than barcodes.

**Selection: RFID Readers**

Identification tag durability is of particular concern for the applications in WeighTrack 2.0, since liquor bottles and chemical bottles generally have a long shelf life with fairly high removal and replacement traffic. Additionally, since all applications involve liquids, waterproof tags were a high priority. The ability to use RFID tags in any orientation, without the need for alignment allows for greater design flexibility in applications that may require pass-through scanning, or stationary platform scanning. Although RFID tag costs are higher, their ability to be reused when a bottle is empty allows for long-term sustainability. Additionally, prices for RFID tags have been falling yearly. Many experts believe that as tags
reach the $0.05 cost, adoption will significantly increase in many fields.\textsuperscript{11} Given the potential for future widespread adoption of RFID tags in packaging, RFID was also a more favorable technology selection in WeighTrack 2.0.

4.4 Information Processing

The primary design constraint for the information-processing unit of WeighTrack 2.0 was to handle the analog and digital signals of the device, while also allowing for Internet connectivity. The primary design decision behind the information-processing unit was the choice to use a microcontroller. An in-depth discussion of the actually microcontroller selection will be presented in section 5.4.

Selection: Microcontroller

4.5 Structure and Form Factor

After accumulating consumer observations and research data, the primary design consideration for the structure and form factor of WeighTrack 2.0 was to integrate the system within a streamlined shelf, allowing for universal placement of any bottle type. The final design details for this will be discussed in section 5.4.

Selection: Shelf Configuration

4.6 User Interface

The primary design constraint for the user interface of WeighTrack 2.0 was providing users with an intuitive shelf system that requires minimal deviation from normal behavior. Additionally, the user interface was designed to easily and noticeably warn users of any recalls or expirations on bottles they are using. The
two feedback design options for WeighTrack’s notification system were individual indicator LEDs or a full RGB LED strip.

4.6.1 Indicator LEDs

Pros: Individual LEDs are very power efficient and can run off of a microcontroller’s 5v power output.

Cons: Each LED would require its own digital output pin, or a shift register (3 pins) would be needed to control multiple LEDs. For a warning message, individual LEDs may not produce a noticeable enough visual signal.

4.6.2 RGB LED Strip

Pros: Sealed strip of LEDs are waterproof and capable of withstanding liquid spills. RGB strip allows for creating multicolor warnings across entire shelf.

Cons: Requires 9v power supply to light strip, which would be separate from microcontroller 5v power supply. Requires additional power transistors to control with microcontroller.

Selection: RGB LED Strip

Although managing a second power supply requires an additional stage of voltage regulation, this ensures that all components of WeighTrack 2.0 receive sufficient power, without seeing drops in voltage or current. The full RGB strip allows for much more noticeable notifications and warning, which is one of the primary design requirements.
### 4.7 Design Approach Summary

Table 4.1 Subsystem Design Selections

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Design Goal</th>
<th>Design Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume Detection Method</td>
<td>Capture volume measurement w/o direct liquid contact</td>
<td>Weight Sensing</td>
</tr>
<tr>
<td>Analog Data Collection</td>
<td>Collect weight data</td>
<td>Load Cells</td>
</tr>
<tr>
<td>Signal Processing</td>
<td>Output 0-5v</td>
<td>Multi-Stage Amplification</td>
</tr>
<tr>
<td>Signal Processing</td>
<td>Protect against noise</td>
<td>Shielding and Filtering</td>
</tr>
<tr>
<td>Identification and Data Management</td>
<td>Identify Bottles</td>
<td>RFID Readers</td>
</tr>
<tr>
<td>Information Processing</td>
<td>Handle analog and digital signals</td>
<td>Microcontroller</td>
</tr>
<tr>
<td>Structure and Form Factor</td>
<td>Support bottles and facilitate use</td>
<td>Shelf Configuration</td>
</tr>
<tr>
<td>User Interface</td>
<td>Notify and Alert Users</td>
<td>RGB LED Strip</td>
</tr>
</tbody>
</table>
5 Design Details

5.1 Analog Data Collection

One of the first major design decision for the analog data collection was selecting what type of load cells to use. Due to the nature of load cell production, however, purchasing individual load cells for prototyping is very expensive, with the base price of many load cells exceeding $100. Knowing that WeighTrack 2.0 would require multiple load cells, this eliminated most load cells of the desired sensitivity. Similar load cells, however, are used in several cheap kitchen scales that are mass-produced. Due to this mass production, purchasing the full digital kitchen scales were, in fact, far less expensive than purchasing individual load cells. Below is a price comparison of a sample load cell and kitchen scale that internally utilizes a similar load cell.

Figure 5.1 Load Cell Price Comparison
The selection of load cells to be pulled from the digital kitchen scales was determined by the accuracy and range of the scales. To achieve a sufficient maximum weight support, scales with a 0-5kg weight range and a 1-gram accuracy were selected.

Since the load cells being pulled from the scales did not have full documentation and datasheets, the first step prior to using them was characterizing the load cell to check for a linear resistance to weight relationship. To complete this characterization, known weights were incrementally added to the load cell, while the resistance across its output was measured. Below is a plot of this characterization.

<table>
<thead>
<tr>
<th>Weight (grams)</th>
<th>Resistance (Ohms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>749.201</td>
</tr>
<tr>
<td>64</td>
<td>749.215</td>
</tr>
<tr>
<td>110</td>
<td>749.221</td>
</tr>
<tr>
<td>294</td>
<td>749.245</td>
</tr>
<tr>
<td>845</td>
<td>749.312</td>
</tr>
<tr>
<td>1141</td>
<td>749.348</td>
</tr>
<tr>
<td>1698</td>
<td>749.42</td>
</tr>
<tr>
<td>2556</td>
<td>749.484</td>
</tr>
<tr>
<td>2864</td>
<td>749.55</td>
</tr>
<tr>
<td>3159</td>
<td>749.587</td>
</tr>
<tr>
<td>3710</td>
<td>749.64</td>
</tr>
<tr>
<td>4024</td>
<td>749.67</td>
</tr>
<tr>
<td>5143</td>
<td>749.834</td>
</tr>
</tbody>
</table>

![Load Cell Characterization](image)

Figure 5.2 Load Cell Characterization

The characterization plot revealed a linear relationship between weight and resistance, which meant that the voltage output from the load cell would also
produce a linear relationship. This simplified the utilization of the load cells, since no software was needed to transform the voltage output to a linear scale.

With the characterization complete, the next step was optimizing the dynamic range output from the load cell. While applying a voltage directly across the load cell would produce a varying voltage as the applied weight changes, the range would be extremely small, since the resistance range calculated in the load cell characterization was only 0.733Ω. A common approach to optimizing the voltage output from this variable resistance is utilizing a Wheatstone bridge. A Wheatstone bridge utilizes four resistors, effectively creating two parallel voltage dividers. If each resistor is exactly matched in the bridge, then the output voltage read across each of the two parallel dividers is zero. Any slight change in resistance for a resistor, however, will cause one side of the bridge to produce a non-zero voltage. This subsequently causes the output voltage read across the bridge to vary. Wheatstone bridges exploit this by inserting a variable resistor in place of one of the fixed resistors. This configuration (shown below) is called a quarter Wheatstone bridge.

![Figure 5.3 Quarter Wheatstone Bridge Configuration](image-url)

Figure 5.3 Quarter Wheatstone Bridge Configuration
For initial load cell testing, this configuration was used. Using the load cell characterization measurements, resistors were selected to construct the rest of the bridge. To minimize the DC offset created by differences in resistance, particular care was taken to precisely match resistor values with the baseline impedance of the load cell. Due to the 10% tolerance on the resistors, finding precisely matching resistors was a challenge. This was ultimately achieved by attaching an ohmmeter to a bare wire test rig that allowed for the placement of a resistor across the raised wires. This facilitated rapid testing of a large number of resistors to find a set of three that matched the impedance of the load cell. Below is an image of the test rig.

![Resistor Test Rig](image.png)

The resistances selected for the initial bridge testing were $R_1 = 748.15 \, \Omega$, $R_3 = 748.70 \, \Omega$, $R_2 = 749.18 \, \Omega$, and $R_x = R_{cell} = 749.201 \, \Omega$. To balance the bridge, the resistances of $R_1$ and $R_3$ were closely matched, and the resistances of $R_2$ and $R_x$ were closely matched. This caused voltages C and D to be most closely matched with
no weight applied, resulting in a \( V_{\text{out}} \) near zero. After resistance values were set, the
quarter Wheatstone bridge configuration was tested by applying known values of
weight to the load cell, and reading the voltage across points C and D, as shown in
Figure 5.3. The results are shown below in Figure 5.5.

![Quarter Wheatstone Bridge Voltage Outputs](image)

This bridge configuration produced a dynamic range of 0.72mV (2.67mV \( \rightarrow \) 3.39mV). Although a slight DC offset (2.67 mV) existed, these measurements
provided a starting point for the signal processing design and implementation.

5.2 Signal Processing

The signal processing subsystem of WeighTrack 2.0 was designed to achieve
two primary goals: Amplifying the signal to a 0-5v range and filtering external noise
that would effect the sensitive load cell measurement.

5.2.1 Amplification

The first step in designing the multi-stage amplification system involves
laying out what each stage of amplification should provide. Since the input signal
from the Wheatstone bridge provides a range of $2.67\text{mV} \rightarrow 3.39\text{mV}$ the first stage was designed to provide a large amount of gain to increase this into the V range. This could be accomplished by either using a non-inverting op-amp configuration, or to use an instrumentation amplifier. The primary difference between the two options is that op-amps use external feedback to complete an operation, while instrumentation amplifiers use internal feedback and a single external gain resistor to set the amount a signal is amplified.\textsuperscript{13} In WeighTrack’s application, the instrumentation amplifier (INA) provides two significant advantages. The first is that setting the gain in an INA is very simple to adjust using a single gain resistor. The second, even greater advantage is that an INA provides a high common mode rejection ratio. This means that any signal component common to both input lines will be rejected as it is passed through the instrumentation amplifier. Since the noise produced by the RFID reader is exposed evenly to both INA input lines, having this high common mode rejection ratio is a valuable feature for the first stage of amplification.

The Analog Devices AD623AN instrumentation amplifier provided an ideal match for this application. The AD623AN runs off of a single-supply and provides a gain of up to 1000, set by a single gain resistor of 100 $\Omega$. Additionally, the common mode rejection ratio increases as the gain of the INA increases. Finally, the operating temperature of the AD623AN is from -40°C to +85°C, allowing for consistent performance in any WeighTrack 2.0 application.
The first stage of amplification was tested by measuring the two extremes of the load cell dynamic range (0g → 5000g). Below in Figure 5.6, these results are displayed. This took an input range of 0.91 mV to 1.7 mV and outputted a range of 1.078 V to 1.85 V. This means that a gain of slightly more than 1000 was achieved.

![Stage One Amplification](image)

**Figure 5.6 Stage One Amplification Testing**

The next step in designing the signal-processing pathway was implementing the second stage of amplification. This stage was passed a voltage input range of 1.078 V to 1.85 V, and was expected to output a voltage range of 0 to 5 V. In order to achieve this, the DC offset of 1.078 V needed to be removed, and an additional gain of ~5 was needed. To achieve this, a difference amplifier was applied using an LMC6482.
The LMC6482 was selected for its ability to run on a single supply, ability to output voltages from rail to rail of power supply (full output swing), and ability to handle inputs down to GND.

![Diagram](image)

**Figure 5.7 Difference Amplifier Configuration**

The difference amplifier configuration, shown above in Figure 5.7, allows for both a difference to be taken between V1 and V2, and then a gain to be set through feedback. This relationship is shown by: \( V_{out} = \frac{R_2}{R_1} (V_2 - V_1) \). Using this equation as a guide, the gain was set by the relationship of \( R_2/R_1 \). Initial values selected to achieve this gain were \( R_2 = 51k\Omega \) and \( R_1 = 10k\Omega \), providing a gain of \(~5.1\). To ensure proper operation, the values of \( R_3 \) and \( R_4 \) were set to have the same values as \( R_1 \) and \( R_2 \) respectively. To generate an appropriate correction voltage, \( V_1 \), a simple voltage divider was created off the 5v power supply rail. Since the input offset was 1.078 V, the designed value of \( V_1 \) was \(~1\) V. Given the voltage divider equation of \( V_{ref} = V_{in} \left( \frac{R_2}{R_2 + R_1} \right) \), the relationship of \( \frac{R_2}{R_2 + R_1} \) was set to equal \(~\frac{1}{5}\). The initial value selected for these resistors were \( R_a = 30k\Omega \) and \( R_b = 120k\Omega \). These produced a reference voltage of 0.989V. Although this was slightly below the 1.078 V offset, this was actually an almost ideal value. An important consideration for
removing the DC offset is that the reference voltage must not be greater than the offset. If this were the case, the signal would be erased until (V2-V1) became greater than 0v. This is because the LMC6482 cannot output voltages below the supplied power rails (GND $\rightarrow$ 5v). The very slight DC offset remaining after subtracting V2-V1 could then be removed in software. An additional railing concern to consider was limiting the gain so the maximum output voltage would remain below 5v. Again, since the LMC6482 can only output voltages between V+ and V-, this would reduce the maximum detection limit of the load cell.

With the appropriate reference voltage created and the difference amplifier constructed, the circuit was then tested. Initial results, however, revealed that the reference voltage was being altered upon being connected to the difference amplifier. Rather than the expected 0.989 V, it drooped to 0.701 V. After and analysis, it was discovered that this was a result of poor input – source impedance matching. As a general rule of thumb, the input impedance should be $\sim$10x greater than the source impedance. The voltage divider, however, provided a source impedance of 24kΩ, while the input impedance of the op-amp was $\sim$8kΩ. To correct for this, the voltage divider resistors were each divided by 10, and the input resistors were each multiplied by 10. This resulted in a source impedance of 2.4kΩ and an input impedance of 83kΩ. Resistors R1 and R2 in the difference amplifier were also adjusted accordingly. Testing after the change showed a return to the expected Vref value of 0.989 V.
With the impedance matching issues corrected, the second stage of amplification was then tested to confirm gain and offset expected results. The resultant second-stage output voltage range was 0.49v → 4.01v and is compared against its input voltages on Figure 5.8 below.

![Stage Two Amplification](image)

Figure 5.8 Stage Two Amplification Testing

Although the output voltage range was not exactly 0-5v, the dynamic range generated was ~3.5 V and the maximum and minimum values were safely protected from railing, which would have erased data that could not be captured in software. By avoiding the top and bottom rails, the remaining DC offset can be removed digitally, and the majority of the desired dynamic range was preserved.

5.2.2 Filtering

The second design constraint for designing the signal-processing pathway was filtering external noise from the signal. The primary noise source for
WeighTrack 2.0 is the 125kHz RF field generated by the RFID reader antenna. As discussed in section 5.2.1, the instrumentation amplifier provided a first line of protection with its high common mode rejection ratio. To help ensure that each input line to the AD623AN received the same noise exposure, and to reduce the induced noise current, the transmission lines from the load cell to the instrumentation amplifier were twisted.

To protect the signal along the entire length of the pathway to the microcontroller, lowpass filters were implemented to remove higher frequencies. The first step in designing the lowpass filters was to determine the critical frequency. Since the signals being measured by the load cells were only dependent on the rate that a human would be removing or replacing a bottle, the only frequencies that should pass are on the order of several hertz. To be safe, a critical frequency of 20 Hz was selected. Resistor and capacitor values were then selected based on the relationship $F_C = \frac{1}{2\pi RC}$, where $F_C = 20$ Hz.

For the first stage lowpass filter, each line entering the instrumentation amplifier was filtered. Placing the filter in close proximity to the first stage of amplification was a key design consideration, since any noise introduced prior to amplification would be amplified with the signal. Since the source impedance of the load cell was \( \sim 375\Omega \), a comparatively large resistor value of 750kΩ was selected for the filter resistor value. The calculated capacitance value was then 10nF. A differential capacitor was also placed between the two lines, with a capacitance of .1uF (10x common mode capacitors).\textsuperscript{15} Figure 5.9 below shows the first stage filter
schematic into the instrumentation amplifier. This first stage lowpass filter provides 20dB attenuation per decade for noise affecting the transmission line.

Figure 5.9 First Stage Lowpass Filter

To further protect against noise, a second stage lowpass filter was included between the instrumentation amplifier and the difference amplifier. This filter was designed with the same cutoff frequency, but was designed with different R and C values. Since the input impedance of the difference amplifier was 125kΩ, the selected R-value for the second stage lowpass filter was 7.5kΩ. The capacitance was then calculated to be 1 uF.

To protect the final length of the transmission line, a third lowpass filter was integrated into the second stage of amplification. Since this was tied to the input impedance of the difference op-amp feedback, it required once again, a high resistance value. For this, the selected R-value was 750kΩ and the selected C-value was 10nF. Shown below in Figure 5.10 is the complete signal-processing pathway with each stage of filtering circled.
5.3 Identification and Data Management

The primary design goal of the identification and data management subsystem of WeighTrack 2.0 was to uniquely identify bottles being introduced to the shelf. Additionally, the design specifications added the constraint that the identification system allows for bottle placement over the entire surface of the shelf. In section 4.3, RFID technology was selected as WeighTrack 2.0’s identifying method.

The first step in developing the RFID system for WeighTrack 2.0 was selecting what driver/reader would be most appropriate for the system. WeighTrack 1.0 utilized two Mifare RC522 RFID readers. Although these readers performed sufficiently well for a proof of concept prototype, they had several major limitations. The first was that each reader required four digital pins from the microcontroller, putting a strain on the pin budget, and limiting flexibility for user interfacing peripherals. The next major drawback was the limited flexibility and
customization of the antenna. The RC522 utilized an on-board pre-manufactured antenna with a coil diameter of approximately 2 cm. This antenna functioned with limited reliability, only reading in tags carefully positioned over the reader within a 1 cm range. In WeighTrack 1.0, the shelf capacity was limited to two bottles due to the RFID reader constraints.

For WeighTrack 2.0's bottle identification system, several new RFID drivers were considered. The RFID selection criteria are shown below in Table 5.1.

Table 5.1 RFID Selection Criteria

<table>
<thead>
<tr>
<th>Cost</th>
<th>Under $30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication Method</td>
<td>Serial or I2C</td>
</tr>
<tr>
<td>Antenna</td>
<td>External for Customization</td>
</tr>
<tr>
<td>Required Supply Voltage</td>
<td>5v or less</td>
</tr>
<tr>
<td>Digital Pin Budget</td>
<td>&lt; 4 pins</td>
</tr>
</tbody>
</table>

The biggest constraint for selecting the RFID driver was the ability to install an external antenna. In order to provide complete coverage of the shelf (dimensions 12” x 24”), external antennas larger than 2 cm in diameter would be required. Most 13.54 MHz RFID drivers lacked the ability to install external antennas, so most the majority of candidates functioned at a frequency of 125 kHz. Based on the criteria in Table 5.1, one RFID driver stood out. This was the ID Innovations ID-3LA. The ID-3LA (125 kHz) cost $25.95 per unit, produced a serial readout, required an external antenna, functioned at a voltage of 2.8-5v, and required only two digital pins from a microcontroller.
In order to use the ID-3LA RFID reader, a customized coil needed to be built. The ID-3LA datasheet indicated that the driver had a built-in capacitance that required a matching coil inductance of **1.337 mH**. Additionally the datasheet (Available in Appendix D - Datasheets) recommended using 28-32 AWG Copper Mag wire.

To prototype and test RFID coils, rectangular wooden frames were constructed to avoid any interference issues with metals or other conductors. An important consideration while building coils is wrapping in a tight uniform fashion. This improves quality factor and helps to generate a more coherent field within the antenna. To construct the prototype coils for WeighTrack 2.0, the wooden frames were affixed to a free-rotating lab stool, while the spool of mag wire was allowed to freely spin on a wooden dowel handle. Figure 5.11 shows the construction setup.
The coils were then wound by rotating the stool and allowing the dowel resistance to tightly wrap the wire, as the angled wooden posts uniformly positioned the wire.

The next step in constructing the coils involved matching the inductance to the internal capacitance of the driver. As specified by the datasheet, the target inductance for the ID-3LA was 1.337 mH. The simplest method for testing a coil’s inductance is placing it in series with a known resistor, sending an AC signal through the two using a waveform generator, and adjusting the frequency, until the peak-to-peak voltage across the coil is equal to half the peak-to-peak voltage of the input signal. This represents the 3dB point, where the impedance is equal to that of the series resistor, and the voltage is evenly divided across the two. To then calculate the inductance of the coil, the following equation can be applied: 

\[ L = \frac{\sqrt{3} R}{2\pi f} \]

where \( R \) is the value of the series resistor, and \( f \) is the frequency of the 3dB point.

In order to begin testing, a razor blade was required to remove the varnish finish from the ends of the mag wire. This allowed for current to flow through the antenna, and voltage measurements to be taken. The next step was using an ohmmeter to measure the exact value of the resistor being placed in series with the antenna. Next, connecting the oscilloscope to the waveform generator input on one channel, and connecting across the antenna on a second channel allowed for monitoring the peak-to-peak voltages as the wave frequency was adjusted. With everything connected, the waveform generator could then be powered on, and the
frequency could be adjusted until the half power inductance point was reached. Figure 5.12 below shows the 3dB point on the oscilloscope and the testing configuration to the right.

![RFID Coil Induction Testing](image)

**Figure 5.12 RFID Coil Induction Testing**

Once the frequency was noted for the 3dB point, the inductance could be calculated. Since the inductance increases with the number of wraps, coils were designed with more wraps than necessary, and tuned down to the proper inductance by removing wraps. Adding wraps is a far more difficult process. Once an appropriate number of wraps were removed, the new end of the coil would have to be scraped clean with a razor blade, and the process could be repeated to find the new inductance value. Once the inductance approached 1.337 mH, wraps would be removed one at a time, to ensure precise induction matching. To confirm the results of the testing, an RLC meter was used to measure the inductance of each coil. The RLC meter automated many of the steps listed above, and returned the inductance and quality factor of an antenna by simply connecting the ends of a coil to the meter leads. Figure 5.13 below shows the RLC meter setup confirming the results of 3dB point tests.
One initial challenge of measuring the coil inductance was inadvertently grounding the voltage point between the resistor and coil with the oscilloscope probe. Since the testing setup quickly became cluttered with scope probes and other leads, it took some time to realize that the reading were inaccurate due to a ground being in the middle of the circuit. Once this was corrected, the coil testing (although tedious) was quite accurate compared to the RLC meter results.

With several coils calibrated to the proper inductance to pair with the ID-3LA driver, the next step was testing to see if an RFID tag could be identified. Connecting and configuring the ID-3LA was fairly straightforward, and followed the datasheet mapping shown in Figure 5.15. Pin 1 connected to GND. Pin 2 connected to any digital pin to control the driver reset. Pins 3 and 4 connected to the antenna leads. Pins 5, 6, and 8 remained disconnected. Pin 7 connected to GND to configure for ASCII output. Pin 9 connected to a serial communication pin. Pin 10 connected to a resistor and LED to indicate when a tag was read. Pin 12 connected to a 5v power supply to drive the antenna.
Since the custom coils built were considerably larger than most built-in RFID antenna, a major concern was the ability of the reader to drive this large of an antenna. However, with the 5v power delivered by the microcontroller, the reader was able to drive the coil and trigger its indicator light when a tag was placed in range. To determine the optimal dimensions of an antenna, numerous prototypes were constructed and measured, allowing for both flexibility in user interface design, as well as optimization of RFID range. Below, Figure 5.15 shows the complete set of calibrated RFID antenna prototypes.
5.4 Information Processing

As discussed in section 4.4, the information processing responsibilities were designed to be handled by a microcontroller. The selection of the microcontroller was based on several design constraints that are outlined below in Table 5.2. To handle sufficient digital logic for the RFID reader(s) and user feedback interface, a digital pin budget of 8 pins was set. To handle the analog signals from load cells, an analog pin budget of 6 pins was set. The other primary requirement for the controller was to be Wi-Fi enabled to allow for application specific web development.

Table 5.2 Microcontroller Selection Criteria

<table>
<thead>
<tr>
<th>Microcontroller</th>
<th>IMG</th>
<th>Cost</th>
<th>WiFi</th>
<th>Analog Pins</th>
<th>Digital Pins</th>
<th>Ethernet</th>
<th>Wiring</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arduino Leonardo</td>
<td><img src="image" alt="Arduino Leonardo" /></td>
<td>$24.95</td>
<td>12</td>
<td>20</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Arduino Yun</td>
<td><img src="image" alt="Arduino Yun" /></td>
<td>$71.95</td>
<td>X</td>
<td>12</td>
<td>20</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Raspberry Pi</td>
<td><img src="image" alt="Raspberry Pi" /></td>
<td>$35</td>
<td>external</td>
<td>17</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Intel Galileo</td>
<td><img src="image" alt="Intel Galileo" /></td>
<td>$63.95</td>
<td>6</td>
<td>14</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Spark.io Photon (un-released)</td>
<td><img src="image" alt="Spark.io Photon" /></td>
<td>$19.95</td>
<td>X</td>
<td>6</td>
<td>8</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Ultimately, the microcontroller selection reduced to two primary options, the Arduino Yun and the Spark.io Photon. With the recent rise of the Internet of Things marketplace, Internet-connected microcontrollers are in increasing demand. For WeighTrack 2.0, the Spark.io Photon presents a very attractive (and low-cost) option, however, as part of an emerging market, this microcontroller is still in
development phase and will not be available until June 2015. Although considerably more expensive, the Arduino Yun satisfied all specification, and even allowed for additional external SD card storage for handling large inventory databases or even hosting a web page. The Yun also utilizes the same form factor as the Arduino Leonardo, which allowed for easy prototyping prior to adding Wi-Fi connectivity.

One challenge that arose with programing the Yun and Leonardo was that their protocols differed from most Arduinos’ for communicating over the serial port. These microcontrollers utilize Serial1, rather than Serial, which introduced additional confusion while debugging the RFID reader. Although the reader appeared to be working according to the indicator light, the tag data was not being transmitted to the serial monitor. Ultimately, this was resolved by the Serial1 correction, but this should be noted for future uses of these microcontrollers.

For the purposes of developing functionality for WeighTrack 2.0, the database was created as a 2-dimensional array, which stored information about bottle ID, expiration data, full weight, current weight, and recall status of a bottle. While a database of this form may cause latency issues in memory for a microcontroller with limited storage, for the sample inventory of ~10 bottles, this was not an issue.

For determining the overall weight on the shelf, the individual weight measurements for each load cell were summed together. As long as the shelf top was rigid enough to avoid significant deformation, the total weight was simply equal to the sum of the weight on each load cell.
5.5 Structure and Form Factor

The primary design constraints for the structure and form factor of WeightTrack 2.0 required matching dimensions to market-specific applications, preventing shelf deformation to maintain linear regime for all measurements, and to be spill resistant to protect internal electronics.

The first step in developing the structure of WeighTrack 2.0 was properly dimensioning the shelf for market applications and for material constraints. The most stringent dimension specification set forth by market research was for the standard shelf size used in a refrigerator, 26.375” x 18”. In order for WeighTrack 2.0 to fit this application, its dimensions needed to be less than these to account for slight variability, and clearances for installing. Since one of the primary fabrication techniques that was to be used for the construction of WeighTrack 2.0 was laser cutting, the limitations of the laser cutter also put constraints on the dimensions. The Epilog laser cutter, however, was capable of handling acrylic sheets up to 24” wide. This falls just under the standard refrigerator shelf width, which provided a natural width selection for WeighTrack 2.0. Additionally, since standard acrylic stock comes in 24” x 12” sheets, this provided a simple baseplate dimension to build around. Since the Business Market and Research Market applications utilize an open, rather than contained space, their dimensioning requirements were less stringent, making the 24” x 12” base a suitable selection as well.

Since a major requirement of the structure was to prevent deformation, an aluminum frame made of .5” x 1” aluminum was constructed to support the system.
The aluminum allowed for rigid lightweight torsion resistance, and provided an appealing outer frame for WeighTrack 2.0. For the completed frame, the outer aluminum was given a brushed finished, with fileted edges using a ¼” corner rounding end mill.

One of the most impactful design decisions made for the structure and form factor of WeighTrack 2.0 was the number of supports selected for the top plate of the shelf. To determine this, a simple experiment was conducted. A 12” x 24” x ¼” piece of acrylic was placed on four corner supports, and a pressure of ~10 kg was applied to the center. The deformation was then visually observed. The same experiment was repeated with six supports. The results are displayed in Figure 5.16 (below). Since any deformation of the top shelf results in measurements deviating from the linear regime, it was important to minimize shelf deformation. For this reason, six supports were selected for the top shelf.

![4 supports](image1)

![6 supports](image2)

Figure 5.16 Top Shelf Support Testing
An important consideration for adding additional supports to the shelf was that each support represented a load cell, which affects the overall system performance. The additional supports each increase the capacity of the shelf by 5 kg, which is beneficial for most outlined applications. Six supports provides 30 kg (66 lbs), while four supports provides 20 kg (44 lbs). A drawback, however, is that each additional support reduces the lower detection limit of the shelf. Assuming each load cell had a lower detection limit of one gram, by distributing the weight of a mass over six load cells increases the lower detection limit to six grams. Ultimately, this was an important tradeoff to make, however, since deformation in the shelf would throw off all measurements, and the lower detection threshold could be adjusted by increasing gain. Additionally, since the load cells received a proportional amount of weight depending on the placement location of a bottle, the weight wouldn’t actually be distributed evenly over all six load cells. For example if a bottle was placed directly over one of the supports, it would receive the large majority of the force, and the detection limit for the load cell would remain near the individual detection limit of a single load cell.

A final major consideration for the structural framework design was integrating the RFID reader into the shelf. To prevent any shorting of the mag wire, this was achieved through 4-40 machine screw posts covered by Teflon spacers on the top shelf. Since the 4-40 machine screws had a slotted head, they could be countersunk into the top shelf, so there were no interference issues with bottle placement and scanning on the shelf.
5.6 User Interface

The User Interface design goals were to provide valuable feedback to users, indicating proper check-in and checkout attempts of bottles, and clearly signifying warnings. In section 4.6, RGB LED strips were selected as the means of communicating user feedback.

Since the RGB LED strip required a 9 V power supply, rather than the 5 V output from an Arduino, an additional power supply was needed. Although some Arduinos can accept a Vin of 9 V and regulate the voltage down to 5 V for its digital output pins, the Yun does not have a built-in regulator. For this reason, the 9 V power supply needed to be externally regulated to 5 V to power the Arduino. This was achieved through an LM7805 fixed voltage regulator. Although the LM7805 essentially burns excess voltage as it regulates, power efficiency wasn’t a key constraint for WeighTrack 2.0 (since it was powered by a wall outlet), so this form of voltage regulation was sufficient for this application.

Although the additional power supply and voltage regulator added slightly to the budget, the overall system saw an important benefit from the external power supply. Due to the draw of the load cells and RFID reader, the 5 V output of the USB powered Arduino measured only 4.195 V. This posed a significant issue, since the many of the amplification components relied on using a true 5 V supply. Additionally, the dynamic range of the load cells was reduced by 1/5, since they could no longer access the 4-5 V range. Connecting the externally regulated 5V power supply to the Arduino, however, solved this problem. By allowing the
Arduino to draw current from the wall outlet, the measured Arduino voltage output increased to 4.975 V.

To power and control the RGB LED strip, NPN power transistors were required to control the flow of current to each color LED. Figure 5.17 below shows the configuration of the control system.

![Figure 5.17 RGB LED Control Schematic](image)

Using TIP47 NPN transistors, the base (pin 1) through a 150 ohm resistor to the Arduino digital I/O pins, the collector (pin 2) connected to the respective RGB LED channel inputs, and the emitter (pin 3) connected to ground. Finally, the 9 V input power line on the RGB strip connected to the 9 V wall outlet power rail. Since continuously powering the LED Strip would draw a relatively large current, PWM-designated I/O pins were used to control each NPN transistor. The Arduino has several digital pins that can be used for pulse width modulation, which allows for brightness control, and increased efficiency in running the RGB strip. The evolved warning indication system will be discussed in section 6.6.
6 Design Evolution

Throughout the design and fabrication process, several new obstacles and discoveries were made that altered and ultimately helped shape the final implementation of WeighTrack 2.0. The significant design evolution steps are outlined here.

6.1 Analog Data Collection

After completing initial testing and achieving functionality through the quarter Wheatstone bridge configurations, several design improvements were made. Inherently, the quarter Wheatstone bridge has several drawbacks. One of the most relevant drawbacks is that by only using a variable resistance on one leg of the bridge, the system is susceptible to performance fluctuations with temperature changes. To overcome this temperature dependence, balanced bridges are implemented, which use variable resistors on both legs of the bridge. A half bridge utilizes two variable resistors, while a full bridge utilized four variable resistors. In this configuration, any temperature related effects are applied to both legs of the bridge, negating their impact on the final output voltage. In order to implement this configuration, the load cell cantilever requires multiple strain gauges to be used as the variable resistors. In the case of a full Wheatstone bridge, the dynamic range can be increased up to four times. This is achieved by placing two strain gauges on the top and bottom of the cantilever. As the cantilever deflects, two strain gauges increase in resistance, while two decrease equal and oppositely in resistance. By
pairing a positive gauge with a negative gauge on each side of the bridge, any
deformation in the load cell will result in a dynamic range four times greater than
the original measurement.\textsuperscript{18} Quarter, half, and full Wheatstone bridge configurations
are depicted below in Figure 6.1.

![Figure 6.1 Quarter, Half, and Full Wheatstone Configurations\textsuperscript{18}]

After performing a component teardown of the digital scale load cells, four
strain gauges were discovered along the aluminum cantilever, with an integrated
full Wheatstone bridge. After closely examining the connections, the positive signal,
the negative signal, and the power and ground leads were sorted out. Similar to the
testing performed in section 5.1, the load cells were retested utilizing the full
Wheatstone bridge configuration. Figure 6.2 below shows the results. The full
bridge configuration not only provided temperature compensation for the circuit, it
also increased the dynamic range from 0.72 mV to 4.814 mV. It also reduced the DC
offset from 2.67 mV to 0.186 mV. Overall, this significantly improved load cell
sensitivity and reduced the amount of calibration required for each individual cell.
Additionally, by utilizing the embedded Wheatstone bridge, the connection lengths
were shortened, which decreased the region for noise to be introduced to the transmission line.

6.2 Signal Processing

With the improvements made in the Analog Data Collection, the signal processing needed several adjustments to maintain the 0-5v output goal. The first major adjustment was reducing the reference voltages that would be subtracted from the signal during the difference amplification stage. Since the DC offset produced by the full Wheatstone bridge was considerably less than the DC offset produced by the quarter Wheatstone bridge, only some load cells required a reference voltage. To determine this, the signals were read into the analog inputs of the Arduino Yun and monitored for offset levels. For load cells that lacked a DC offset, or only required a minor adjustment, the reference voltage was connected to ground, and the remaining offset was subtracted in software. For load cells that still had a significant DC offset, an appropriate reference voltage was created using a
voltage divider with source impedance of less than 20kΩ. As the initial design specification required, this provided a mechanism for analog load cell calibration.

The second major adjustment made to the signal processing pathway was increasing the gain of the second stage amplification. Since the top plate required six load cell supports to prevent deformation, the maximum weight of the shelf increased to 30 kg, but the lower detection threshold increased slightly as well. Since design specifications only require 20 kg maximum weight, additional amplification could be applied to increase the lower detection threshold, while also decreasing the maximum weight allowance. Since the instrumentation amplifier was already at its maximum gain of 1000, the gain was adjusted using the op-amp difference amplifier. Altering the ratio of input resistors accordingly increased the gain of five at this stage to a gain of six.

6.3 Identification and Data Management

One of the most evolving systems of WeighTrack 2.0 was the RFID identification system. One major drawback of RFID readers is their ability to read multiple tags at once. In fact, if multiple tags are within range of an RFID reader, then neither tag will be identified. For this reason, several RFID interfaces were designed to work around this limitation. There were two primary configurations that were considered. The first was utilizing several RFID drivers and antennas to create fixed platforms across the shelf for bottles to rest upon. The second was creating an RFID antenna positioned at the front of the shelf that bottles were
passed over to check in and check out. Below are CAD models of the two system designs utilizing either pass-over scanning or stationary platforms.

![CAD models of two system designs](image)

**Figure 6.3 Pass-over and Platform Design Structures**

To determine coil performance for each configuration, the methods described in section 5.3 were used to build and test coils of several different dimensions. These results are shown in Table 6.1 below.

**Table 6.1 RFID Coil Performance Results**

<table>
<thead>
<tr>
<th>Dimensions</th>
<th>Number of Wraps</th>
<th>Vertical Read Range</th>
<th>Inductance</th>
<th>Design Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.66” x 10”</td>
<td>45</td>
<td>6.75”</td>
<td>1.337 mH</td>
<td>6 stationary platform readers</td>
</tr>
<tr>
<td>4.4” x 10”</td>
<td>45</td>
<td>7”</td>
<td>1.405 mH</td>
<td>5 stationary platform readers</td>
</tr>
<tr>
<td>2” x 10”</td>
<td>54</td>
<td>4”</td>
<td>1.333 mH</td>
<td>2 front pass-over readers</td>
</tr>
<tr>
<td>1” x 10”</td>
<td>52</td>
<td>1.5”</td>
<td>1.340 mH</td>
<td>2 front pass-over readers</td>
</tr>
<tr>
<td>2” x 22”</td>
<td>38</td>
<td>2”</td>
<td>1.339 mH</td>
<td>1 front pass-over reader</td>
</tr>
<tr>
<td>1” x 22”</td>
<td>38</td>
<td>.5”</td>
<td>1.335 mH</td>
<td>1 front pass-over reader</td>
</tr>
</tbody>
</table>

54
Although the wider platform antennas produced a greater vertical range than the pass-over designed coils, this didn’t necessarily translate to improved performance. While the 1cm range used in WeighTrack 1.0 was unreliable, this was due to both the severely deficient range and the small placement area on the coil. By significantly increasing the read-in range to six or seven inches (as is the case for the platform antennas), ambiguities can actually arise during usage. For example, when bottles are placed on the shelf or removed, they would remain in range of the RFID reader for an extended time. This time gap would cause latency in the system for manipulating other bottles. Additionally, the larger vertical ranges also corresponded to wider fields. This causes two issues. First, bottles placed on adjacent platforms could be read into either antenna and cause ambiguity in identification and weight measurements. Additionally, by placing coils with a large range adjacent to each other, interference actually occurs between RF fields. In order for this system to be reliable, the antennas would have to be spaced out, thereby wasting usable space on the shelf.

Further restrictions of the fixed platform design include limited capacity and serial data transmission. By having fixed platforms that can only handle one bottle at a time, the shelf’s capacity is set by the number of readers and antennas used. Additionally, since each antenna requires its own RFID reader, the pin budget of the Arduino would be exhausted. The final limitation that ultimately put the platform design to rest was the Arduino Yun’s ability to handle multiple serial input readings. With only one serial communication line, the Yun was actually limited to using only
one RFID reader. This limitation also aided in the decision to use a single front coil for the pass-over design, rather than two side-by-side.

The last design decision for fabricating the finalized RFID coil was whether to use the 1” or 2” wide antenna design. The advantage of the smaller width was maximizing the usable shelf space. Since the bottles placed on the shelf could not be within range of the reader, the 1” design provided slightly more available space. As Table 6.1 shows, however, the vertical read range of the 1” wide design was a mere .5”. This would have made scanning tags too unreliable and difficult to check out. For this reason, the 2” x 22” antenna design was ultimately selected.

6.4 Information Processing

Throughout the design process, the microcontroller selection underwent few alterations. One adjustment that had to be made, however, was in response to floating analog values. During testing of individual load cells, it was noticed that all unconnected analog pin values would change in response to a single analog pins reading. This was a result of these pins “floating.” To correct this issue and properly zero the analog pins, 11kΩ pull-down resistors were connected between ground and the analog inputs. This immediately corrected the issue.

The main information processing mechanism that evolved throughout the design was the Arduino software that interfaced the RFID readings, load cell readings, the inventory database, and the user feedback warnings. There were two primary scripts of code that were used for WeighTrack 2.0. The first was a load cell calibration code. This calibrated the overall weight of the shelf to zero at startup,
and then printed to serial the weight measurements on each load cell. This then allowed for either adjusting the DC offset through the analog reference voltage, or through subtracting a digital value from the reading in software. During calibration, it was important to set the baseline value of each load cell to approximately the same value. This way, a bottle on the left half of the shelf wouldn’t receive a biased weight reading over a bottle placed on the right half of the shelf.

The second script of code contained the full system integration of all subsystems. At startup, this automatically calibrated the load cells to zero by taking ten measurements of the empty shelf, averaging them, and subtracting them from all subsequent readings. During this calibration process, the RGB LED strip cycled through all of its colors to indicate that the system was starting up and calibrating. Once calibrated, the shelf was lit blue, to indicate idle and ready phase. At this point, the system monitored for weight changes or RFID tag readings. If an RFID tag was introduced, then the system waited for a weight change to occur, allowed the new weight to settle, averaged five readings, and updated the weight of the newly introduced RFID tag’s bottle. During this time, the Arduino would calculate the volume by dividing the new weight by the full weight in the database, and the serial readout would print how full the bottle was by volume. If the bottle had no warnings within its database, and was successfully checked in, then the shelf lit green for approximately 1.5 seconds before returning to the blue idle state. If the bottle were expired, however, the shelf would flash red at a medium rate for 3 seconds. If the bottle were approaching expiration, then the shelf would flash yellow for 3 seconds.
Finally, if the bottle had a recall warning attached to it, the shelf would rapidly flash red for 4 seconds. Additionally, with each system warning, the serial readout also provided a text explanation of the bottle’s status and any warnings. The shelf would also provide these warnings to users as they removed bottles, but would wait to update weight values until the bottle was placed back on the shelf, since bottles remaining on the shelf would presumably remain at the same weight until they are being used.

The full Arduino code scripts are included below in Appendix E for reference.

6.5 Structure and Form Factor

By having all electronics embedded beneath the top platform, accessing the Arduino for resetting and re-programming was a challenge. To overcome this, external power and USB slots were installed in the back of WeighTrack 2.0. To achieve this, the rear aluminum bar was CNC’d to create a rounded slot for the 9 V barrel power jack and a rectangular slot for a 6” micro USB extender to be accessed. By connecting the 9 V jack to the 5 V regulator internally, the system could be powered completely externally. Additionally, by connecting the micro USB extender to the Arduino internally, the system could be updated and programmed by the external plug in the back. With this configuration, once the hardware build was complete, no physical adjustments needed to be made to adjust software functionality.
### 6.6 User Interface

The design evolution of the system's user interface primarily involved RFID tag selection, and development of the shelf's warning/notification system.

With the hardware of WeighTrack 2.0 complete and functioning, a variety of RFID tags were tested on the system for performance optimization. The two primary tags that were considered for the final system were both round 1 mm thick plastic tags that could be discretely affixed to the bottom of bottles. The primary physical distinction between the tags was their respective diameter differences of 25 mm and 50 mm. To test the performance of each tag, a vertical and surface detection range was mapped over an image of the shelf. These mappings are displayed below in Figure 6.4 and Figure 6.5.

- **Large RFID Tag**
  - 4 inch vertical range

- **Small RFID Tag**
  - 1.5 inch vertical range

*Figure 6.4 RFID Tag Vertical Range Mapping*
Although the 50 mm tag provided a greater vertical read range, it also provided a considerably larger surface read range than the 25 mm tag. As discussed earlier, this surface read range limits the usable surface area of the shelf, since multi-tag interference will otherwise occur. As shown in Figure 6.5, the 50 mm read range covers nearly half the shelf surface area, limiting bottle storage to only the rear portion. The 25 mm read range, however, only covers the local area of the RFID antenna, allocating the majority of the shelf for bottle storage. Additionally, the 1.5” vertical read range yielded highly reliable pass-over read-ins of bottles, and limited latency between reading and weighing, allowing for faster bottle manipulation by users. For these reasons, the 25 mm tags were used for the final version of WeighTrack 2.0.

The second major design evolution of the user interface system was the warning system. As previously discussed, the notification system began as basic 5 mm indicator LEDs, and evolved into a full RGB LED strip capable of lighting the entire shelf surface. The full range of notifications and warnings can be viewed in Appendix F.
7 Evaluation

Given the numerous subsystems of WeighTrack 2.0, final testing and evaluations were conducted on each individual subsystem, as well as the overall system performance.

7.1 Overall System Performance

To test the overall system performance, a precision scale was used to compare the WeighTrack 2.0 “Percent Full” readings with known weight percentages of liquid in a bottle. This test was conducted by filling an RFID enabled bottle to the top with water. This was then weighed (400g), and used as the “Full” value, which was entered into WeighTrack 2.0’s database. Then, with three additional bottles already placed on WeighTrack, the test bottle was checked in. The WeighTrack “Percent Full” reading was noted, and the bottle was removed. Then, as water was incrementally poured out of the bottle, the bottle was re-weighed by the precision scale, and returned to WeighTrack for check-in. This process continued for 15 pours until the bottled was empty. To analyze the results, the actual percent full by weight was compared to WeighTrack’s “Percent Full” measurement after each pour. This data was then charted in Excel, with the results shown in Figure 7.1. From the collected data, the mean average was calculated to be 2.02%. On a 400g bottle, this corresponds to an 8 gram measurement accuracy. For liquids the density of water, this corresponds to 8 mL volumetric accuracy.
Although this accuracy did not meet the initial design goal of 500 mg (single pill) accuracy, this is actually an extremely promising result for the targeted applications. Putting the results from this testing into perspective, if WeighTrack 2.0 were at full capacity (measured to be 25 kg after final signal amplification) it would have an overall accuracy of 0.032% accuracy. In the context of a 750ml liquor bottle, WeighTrack 2.0 would have a measurement accuracy of 1.066%. Although the detection accuracy may have missed the 500 mg target, the trade-offs made that decreased this accuracy were actually quite valuable.

The major tradeoff for system accuracy came at the cost of maximum weight capacity. Had the system maintained the 500 mg accuracy using precision load cells, the weight capacity would have been approximately 4 kg, which is 1/10th the desired capacity for a liquor shelf. Although the 500 mg accuracy specification
would have been an impressive achievement, this design specification was ultimately a poor fit for the application purposes designed for.

Beyond showing the volumetric measurement performance of WeighTrack 2.0, this test also illustrated the accuracy capabilities with multiple bottles placed on the shelf.

### 7.2 Analog Data Collection

The primary design requirements of the analog data collection subsystem are outlined below:

- 95% accuracy for volume measurement
- A lower detection limit of 500mg (a single pill)
- Affordability
- Upper detection limit of 20 kg

The overall system performance testing revealed that the system met and exceeded the 95% volumetric measurement accuracy by nearly 3%. While the system did not meet the 500 mg lower detection limit, as discussed in section 7.1, this was an inappropriate design requirement for the selected applications. However, the 500 mg detection limit could be met in a future model of WeighTrack that is designed for precision detection, rather than 20 kg weight capacity. This could be achieved by distributing weight over fewer load cells, increasing the signal processing amplification, and by utilizing more sensitive resistive load cells. The analog data collection subsystem performed very well in terms of affordability. By pulling load cells from mass produced digital scales, costs were considerably reduced from the
$100 per load cell range that was common to individually sold cells. Had the budget for the project been greater the $500, it may have been valuable to extensively test individual load cells for performance. Particularly for mass production of WeighTrack, the load cells that cost ~$100 on their own, would only cost a few dollars in bulk.

The final analog data collection design requirement of providing an upper detection limit of 20 kg was also met. This was determined by progressively adding mass to the shelf until the weight readout reached a maximum. For the final design, this occurred at ~25 kg. Furthermore, by utilizing balanced, full Wheatstone bridge configurations, the analog signal processing is also capable of functioning over a wide temperature range.

### 7.3 Signal Processing

The primary design requirements of the signal processing subsystem are outlined below:

- Protection from noise generated by either external sources or other internal components
- Amplifying the analog signal to a usable 0-5V range
- Allow for analog calibration

The third order lowpass filter system integrated into the signal processing pathway provided 60dB attenuation per decade. Consider the 125 kHz noise produced by the RFID antenna. Since the third order lowpass filter had a critical frequency of 20 Hz, the RFID noise was four decades removed from the cutoff frequency. This resulted
in 240 dB attenuation for the 125 kHz signal. Two tests were conducted to verify this. First, load cell measurements were taken with the RFID antenna connected and disconnected and then compared. The mean of the difference between these measurements produced a result of less than .5 percent. The second test to verify the effectiveness of the lowpass filtering utilized an additional precision scale. To illustrate the noise effect on an unprotected circuit, a scale was used to measure a bottle on a standard desk, and then again on top of the RFID antenna for WeighTrack 2.0. The results are displayed below in Figure 7.2.

![Figure 7.2 Noise Measurement Interference](image)

The precision scale produced an 11 g or 2.7% due to the RFID field. Since the WeighTrack measurements under field and non-field conditions were less than .5%, the lowpass filter system was deemed effective.
The signal processing design requirements of producing a 0-5v output and offering a means of analog calibration were also successful. Under min and max load, the output signal after both stages of amplification was 0.1-4.8v. As discussed earlier, this was a near ideal range, since removing too much DC offset or amplifying the signal too much would produce railing on the GND or 5v line. This railing would correspond to lost data. For a means of analog calibration, the second stage difference amplifier allowed for adjustable DC offset through subtracting a reference voltage. This voltage could be appropriately set using a voltage divider to calibrate the signal output.

7.4 Identification and Data Management

The primary design requirements of the identification and data management subsystem are outlined below:

- Enable bottle placement over entire shelf surface
- Prevent measurement ambiguity between multiple bottles
- Allow for multiple bottle identification within system

Utilizing the ID-3LA driver and customized front pass-over antenna design, each of the three design specifications were largely met. Combining the 25 mm RFID tags with the 2” x 22” front reading antenna, ~80% of the shelf surface area permitted bottle placement and storage. Additionally, this design prevented interference between tags, and allowed as many bottles to be measured and placed on the shelf as could fit. Furthermore, with a 1.5” vertical tag read range, check-in and check out
scans were highly reliable, while also avoiding latency between identification and weight measurement.

### 7.5 Information Processing

The primary design requirements of the information processing subsystem are outlined below:

- Ability to handle sufficient analog inputs for measurement data collection
- Possess enough digital I/O pins to handle identification information and output system feedback to user
- Wi-Fi enablement for web integration and development
- Sufficient storage for managing inventory database

The finish WeighTrack design utilized all six analog input pins of the Arduino Yun, yet only required use of five digital pins, thereby allowing for potential future alterations without changing microcontroller. Additionally, the Yun’s Wi-Fi connectivity allows for passing the collected serial data directly to a MySQL database for web integration. Additionally, this database can be customized to provide any additional relevant bottle information and warnings depending on application. Finally, the Yun’s ability to add an external SD card provides the ability to store data on extremely large liquid inventory databases.

### 7.6 Structure and Form Factor

The primary design requirements of the structure and form factor of WeighTrack 2.0 are outlined below:
• Streamlined design to allow for wide range applications
• Dimensions capable of fitting within standard refrigerator shelf and on lab bench
• Preventing shelf deformation to ensure measurements remain in the linear regime
• Provide electronics with protection from spills

The final dimensions of WeighTrack 2.0 were 1.375” x 12” x 24”. This streamline design allows for unobtrusive placement in a wide range of applications, including each of the targeted markets. These dimensions also allow for feasible integration into a standard refrigerator for proof of concept testing in the consumer market.

The aluminum frame and ¼” acrylic top shelf supported by six load cells provided extensive protection from deformation and allowed measurement data to be effectively summed due to the linear distribution. The accuracy of this is once again reflected in the 2.02% accuracy measurement from section 7.1. To provide the electronics with protection from spills, several actions were taken. First, the indicator LED strip was covered in a weatherproof coating to protect against spills along the front edge of the shelf. Next, the top shelf was made from ¼” acrylic with a 1/8” tolerance gap to the aluminum frame. This prevents any direct spills on top of the microcontroller and other sensitive components. Finally, all electronics, including the load cells, microcontroller, and breadboards were placed upon plastic standoffs to prevent any contact with liquid that breaches the sides of the top shelf.

While circuitry isn’t 100% waterproof, it does have a degree of spill protection.
8 Budget

The overall budget for fabricating WeighTrack 2.0 came out to $308.01. The details of this can be viewed in Appendix A - Bill of Materials. The biggest cost considerations for the final design were focused around load cell, microcontroller, and RFID driver selection. In each case, the cost of acquiring a single part for prototyping was far greater than the cost of mass-producing a manufactured product would be. In the case of the load cells, significant costs savings were obtained through pulling load cells from mass-produced kitchen scales from china and characterizing them. Had individual load cells been purchased for prototyping, the cost of the analog data collection system would have been over at least $100 alone. Since a major requirement of the system was to create a physical platform for web-development and integration, acquiring a Wi-Fi enabled microcontroller was crucial. Although this cost ~$75 for this prototype, a mass produced version would be much less. This is due in part to the rising Internet of Things market, which demands cheap, Wi-Fi enabled microcontrollers. This was discussed briefly in section 4.4 with regard to the Spark.io Photon, a Wi-Fi enabled microcontroller that will cost $20 when it hits market later this spring. To help reduce costs of WeighTrack’s RFID system, a design that required only one RFID driver was selected over a six drivers design, effectively saving $125 on parts. Considering the microcontroller, load cell, and RFID reduced costs, as well as bulk costs of other components, a mass-produced estimate for WeighTrack 2.0 would be ~$100-$150.
9 Conclusion

WeighTrack 2.0 was designed to provide a means of dynamically tracking liquid inventories to provide users with valuable time-dependent information regarding volume levels, expiration warnings, recall warnings, and any other application specific information necessary. To complete this task, WeighTrack 2.0 utilized a load cell network with noise protected signal processing and amplification to capture bottle weight data to within 2% accuracy, while also providing a maximum weight capacity of 25 kg (55 lbs). WeighTrack coupled this data collection ability with customized RFID technology to allow for reliable and consistent unique bottle identification. This integrated system, powered by a Wi-Fi enabled microcontroller allowed for time-dependent consumption tracking of an entire inventory, while also providing users with real time warnings of expiration dates and recall statuses.

9.1 Depth of Applications

By creating a means of dynamically tracking liquid inventories, users can receive benefit beyond knowing current content levels, time until expiration, and
content recall status. By generating a time-dependent history of liquid consumption, software can extract patterns and apply machine-learning algorithms to provide individuals with a customized analysis of their habits.

In the Consumer Market, users could access a virtual content library of their kitchen while shopping in a store. Additionally, software could leverage the tracking data to show users approximately how long the expected lifetime of a product will be in their household before making a purchase. Within a household, families could set up unique user accounts to track what food each individual prefers. This information could even be applied to automatic ordering through Amazon Fresh or Stop & Shop Peapod. By syncing the database with a family calendar, ordering could adjust for when different individuals will be home, to ensure that their preferred foods are ordered when they return.

In the Business Market, managers would be able to compare the changes in liquid content over an evening with the corresponding bill of sales to see exactly how much liquor was poured in each drink. Over time, this data could reveal customer habits and preferences based on the time of the year, or even night of the week. This information could then help dictate business promotions used by restaurants to draw in customers. The patterns in inventory and consumption history could also be utilized to improve predictive purchasing for holidays, special events, and even seasonal changes.

In the Research Market, labs could closely monitor the rate at which they consume individual chemicals during different phases of testing. This could allow
for an automatic ordering and re-stocking system. By syncing with the lab’s calendar, predictive models could be used to ensure sufficient quantities of all reagents necessary to complete experiments. Additionally, this data would be highly valuable to chemical manufacturers and distributors for targeted sales and advertising to lab groups.
10 Next Steps

The next steps for developing WeighTrack 2.0 into an even more robust and dynamic system revolve largely around software development and refinement. While WeighTrack 2.0 currently runs with a clean user interface and relatively high degree of accuracy, clever algorithms could be applied to improve performance considerably. For instance, Markov probability models could be used to track weight distribution across the shelf and identify bottle removal even if the RFID tag weren’t passed across the reader. Additionally, by detecting changes in weight and using these probability models to identify which bottle was removed, any warnings tied to that bottle could still be displayed on the shelf. Furthermore, more extensive automated calibration and measurement techniques could be used to further improve the accuracy of the weight measurements for bottles.

Another step forward for the continued development of WeighTrack would be working with a team of web developers to effectively pass the serial input data into an online database and integrate it with a mobile and native web interface. This would allow users to track inventory data on the go with their smartphones, or at home on their desktop.

From a hardware perspective, the next steps moving forward would involve fabricating the signal processing electronics into a printed circuit board and adopting low-cost Wi-Fi enabled microcontrollers as the system processor. Converting the circuitry to a PCB would provide several advantages over the current
setup. Not only would a PCB reduce the size constraints of the internal electronics, it would also aid in noise reduction for the system. The PCB would allow for improved component shielding and reduce the length of connections between components. This would reduce the amount of area for noise to be introduced to the system, which would further improve the integrated lowpass filtering. Adopting low-cost Wi-Fi enabled microcontrollers would allow for WeighTrack to be a more consumer friendly device, since one of the greatest cost constraints on the current system is the use of the Arduino Yun. My utilizing one of these new controllers, production costs would be cut considerably.

A final step moving forward is reaching out to explore application specific partnerships. This project has recently received interest from Sigma Aldrich, one of the leading chemical producers and manufacturers in the United States, to examine the possibility of applying WeighTrack technology to their chemical bottles. This would allow for improved consumer data collection, and facilitate automatic ordering and targeted advertising. Additional outreach could be explored with refrigerator manufacturers to see about licensing the technology to integrate within a smart-refrigerator. Furthermore, local outreach to restaurants and bars could be considered to test WeighTrack in a live environment and receive feedback for making further improvements.
Acknowledgements

I would like to thank Professor Marko Lončar and Anderson Research Group EE Specialist, Marco Rivero, for their support and guidance as my advisors on this project. I would also like to thank Jim MacArthur and Avinash Uttamchandani for their support and assistance throughout the design, testing, and building stages of WeighTrack 2.0. Their astute debugging and problem solving abilities were instrumental in the success achieved in this capstone project. Additionally, a special thanks goes out to the Jordan Stephens and the rest of the Pierce teaching labs staff for their support and expertise in fabrication that allowed WeighTrack 2.0 to take form. I would also like to thank Professors Woodward Yang and Rob Wood, along with the rest of the ES100hf teaching staff for their patience and dedication to the senior engineers throughout the year. Finally, this senior design project could not have been completed without the support and encouragement of my fellow senior EE concentrators – Thank you.
### Appendix A - Bill of Materials

<table>
<thead>
<tr>
<th>Component</th>
<th>Quantity</th>
<th>Cost</th>
<th>Total Cost</th>
<th>Part #</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load Cell</td>
<td>6</td>
<td>5.17</td>
<td>31.02</td>
<td>B00FG05VTA</td>
<td>Amazon</td>
</tr>
<tr>
<td>Instrumentation Amplifier</td>
<td>6</td>
<td>5.00</td>
<td>30.00</td>
<td>AD623N</td>
<td>DigiKey</td>
</tr>
<tr>
<td>Op-Amp</td>
<td>3</td>
<td>1.73</td>
<td>5.19</td>
<td>LMC6482</td>
<td>DigiKey</td>
</tr>
<tr>
<td>.01uF Electrolytic Capacitor</td>
<td>18</td>
<td>0.02</td>
<td>0.36</td>
<td>-</td>
<td>DigiKey</td>
</tr>
<tr>
<td>10uF Ceramic Capacitor</td>
<td>18</td>
<td>0.02</td>
<td>0.36</td>
<td>-</td>
<td>DigiKey</td>
</tr>
<tr>
<td>.1uF Ceramic Capacitor</td>
<td>6</td>
<td>0.02</td>
<td>0.12</td>
<td>-</td>
<td>DigiKey</td>
</tr>
<tr>
<td>10nF Ceramic Capacitor</td>
<td>18</td>
<td>0.02</td>
<td>0.36</td>
<td>-</td>
<td>DigiKey</td>
</tr>
<tr>
<td>1uF Ceramic Capacitor</td>
<td>6</td>
<td>0.02</td>
<td>0.12</td>
<td>-</td>
<td>DigiKey</td>
</tr>
<tr>
<td>330Ω resistor</td>
<td>4</td>
<td>0.02</td>
<td>0.08</td>
<td>-</td>
<td>DigiKey</td>
</tr>
<tr>
<td>11kΩ resistor</td>
<td>6</td>
<td>0.02</td>
<td>0.12</td>
<td>-</td>
<td>DigiKey</td>
</tr>
<tr>
<td>150kΩ resistor</td>
<td>12</td>
<td>0.02</td>
<td>0.24</td>
<td>-</td>
<td>DigiKey</td>
</tr>
<tr>
<td>100Ω resistor</td>
<td>6</td>
<td>0.02</td>
<td>0.12</td>
<td>-</td>
<td>DigiKey</td>
</tr>
<tr>
<td>750kΩ resistor</td>
<td>24</td>
<td>0.02</td>
<td>0.28</td>
<td>-</td>
<td>DigiKey</td>
</tr>
<tr>
<td>Arduino Yún</td>
<td>1</td>
<td>71.95</td>
<td>71.95</td>
<td>DEV-12053</td>
<td>SparkFun</td>
</tr>
<tr>
<td>125kHz RFID Driver</td>
<td>1</td>
<td>25.95</td>
<td>25.95</td>
<td>ID-3LA</td>
<td>SparkFun</td>
</tr>
<tr>
<td>5mm Red LED</td>
<td>1</td>
<td>0.35</td>
<td>0.35</td>
<td>COM-09590</td>
<td>SparkFun</td>
</tr>
<tr>
<td>32 AWG Mag Wire</td>
<td>1</td>
<td>5.97</td>
<td>5.97</td>
<td>MW0552</td>
<td>Amazon</td>
</tr>
<tr>
<td>50mm 125kHz RFID Tag</td>
<td>6</td>
<td>1.29</td>
<td>7.74</td>
<td>28142-ND</td>
<td>DigiKey</td>
</tr>
<tr>
<td>25mm 125kHz RFID Tag</td>
<td>6</td>
<td>1.29</td>
<td>7.74</td>
<td>32397-ND</td>
<td>DigiKey</td>
</tr>
<tr>
<td>.5” x 1” x 6’ Aluminum Stock</td>
<td>1</td>
<td>23.37</td>
<td>23.37</td>
<td>8975K11</td>
<td>McMaster-Carr</td>
</tr>
<tr>
<td>¼” x 12” x 24” Clear Blue Acrylic</td>
<td>1</td>
<td>19.74</td>
<td>19.74</td>
<td>85635K531</td>
<td>McMaster-Carr</td>
</tr>
<tr>
<td>¾” x 12” x 24” Black Acrylic</td>
<td>1</td>
<td>18.80</td>
<td>18.80</td>
<td>8505K93</td>
<td>McMaster-Carr</td>
</tr>
<tr>
<td>3” x 6” x 0.025” Black Anodized Aluminum</td>
<td>1</td>
<td>6.80</td>
<td>6.80</td>
<td>24908-07</td>
<td>Inventables</td>
</tr>
<tr>
<td>1m Sealed RGB LED Strip</td>
<td>1</td>
<td>14.95</td>
<td>14.95</td>
<td>COM-12023</td>
<td>SparkFun</td>
</tr>
<tr>
<td>NPN Silicon Transistor</td>
<td>3</td>
<td>0.66</td>
<td>1.98</td>
<td>TIP47</td>
<td>DigiKey</td>
</tr>
<tr>
<td>5v Regulator</td>
<td>1</td>
<td>0.67</td>
<td>0.67</td>
<td>LM7805</td>
<td>DigiKey</td>
</tr>
<tr>
<td>Product</td>
<td>Quantity</td>
<td>Price 1</td>
<td>Price 2</td>
<td>Part Number</td>
<td>Supplier</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>----------</td>
<td>---------</td>
<td>---------</td>
<td>-------------</td>
<td>--------------</td>
</tr>
<tr>
<td>9v DC Wall Power Supply</td>
<td>1</td>
<td>5.95</td>
<td>5.95</td>
<td>TOL-00298</td>
<td>SparkFun</td>
</tr>
<tr>
<td>DC Barrel Jack Connector (M)</td>
<td>1</td>
<td>0.50</td>
<td>0.50</td>
<td>PJ1-021-SMT-TR</td>
<td>DigiKey</td>
</tr>
<tr>
<td>DC Barrel Jack Connector (F)</td>
<td>1</td>
<td>0.85</td>
<td>0.85</td>
<td>EP501B</td>
<td>DigiKey</td>
</tr>
<tr>
<td>Micro USB 6&quot; Extender</td>
<td>1</td>
<td>5.39</td>
<td>5.39</td>
<td>B00HA0K7XE</td>
<td>Amazon</td>
</tr>
<tr>
<td>Slotted 4-40 Machine Screw</td>
<td>70</td>
<td>0.07</td>
<td>4.90</td>
<td>96640A056</td>
<td>McMaster-Carr</td>
</tr>
<tr>
<td>4-40 Locking Nut</td>
<td>24</td>
<td>0.08</td>
<td>1.92</td>
<td>90715A005</td>
<td>McMaster-Carr</td>
</tr>
<tr>
<td>Solderless Breadboard</td>
<td>2</td>
<td>7.00</td>
<td>14.00</td>
<td>03MB102</td>
<td>Amazon</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>$308.01</strong></td>
</tr>
</tbody>
</table>
Appendix B - CAD Models & Engineering Drawings

Figure 0.1 CAD Model and Final Build
Appendix C – Circuit Diagrams / Schematics

Figure 0.1 WeighTrack 2.0 Signal Processing Schematic
# Appendix D - Datasheets

<table>
<thead>
<tr>
<th>Part</th>
<th>Manufacturer</th>
<th>Datasheet Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIP47</td>
<td>FAIRCHILD</td>
<td><a href="https://www.fairchildsemi.com/datasheets/TI/TIP49.pdf">https://www.fairchildsemi.com/datasheets/TI/TIP49.pdf</a></td>
</tr>
</tbody>
</table>
Appendix E - WeighTrack 2.0 Arduino Code

// Tyler Kugler
// WeighTrack 2.0
// Senior Thesis
// April 2, 2015

// Load Cell Calibration Code

// Load Cell Constants
const int RearLeftCell = A3;  // Analog input pin that left sensor is attached to
const int RearCenterCell = A2; // Analog input pin that right sensor is attached to
const int RearRightCell = A5;  // Analog input pin that left sensor is attached to
const int FrontLeftCell = A0; // Analog input pin that right sensor is attached to
const int FrontCenterCell = A1;  // Analog input pin that left sensor is attached to
const int FrontRightCell = A4; // Analog input pin that right sensor is attached to

// Indicator Lights
const int GreenLight = 9;
const int RedLight = 10;
const int BlueLight = 11;

// Load Cells
float RearLeftValue = 0;       // value read from rear left sensor
float RearCenterValue = 0;      // value read from rear center sensor
float RearRightValue = 0;       // value read from rear right sensor
float FrontLeftValue = 0;       // value read from front left sensor
float FrontCenterValue = 0;      // value read from center sensor
float FrontRightValue = 0;      // value read from front right sensor

float TotalWeight = 0;           // total weight read in from both sensors
float Current_Weight = 0;
float Calibrated_Weight = 0;
float initial_calibration1 = 0;
float initial_calibration2 = 0;
float initial_calibration3 = 0;
float initial_calibration4 = 0;
float initial_calibration = 0;

void setup() {
    // initialize serial communications at 9600 bps:
    Serial.begin(9600);

    initial_calibration1 = readWeight();
    delay(10);
    initial_calibration2 = readWeight();
    delay(10);
    initial_calibration3 = readWeight();
    delay(10);
    initial_calibration4 = readWeight();

    initial_calibration = (initial_calibration1 + initial_calibration2 +
    initial_calibration3 + initial_calibration4) / 4;

    // Set RFID Reset High
    pinMode(RFIDResetPin, OUTPUT);
    digitalWrite(RFIDResetPin, HIGH);

    // Indicator Lights
    pinMode(GreenLight, OUTPUT);
pinMode(RedLight, OUTPUT);
pinMode(BlueLight, OUTPUT);

}  

void loop() {
  Current_Weight = readWeight();
  Calibrated_Weight = Current_Weight - initial_calibration;
  if (500 > Calibrated_Weight && Calibrated_Weight > 0) {
    analogWrite(BlueLight, 0);
    analogWrite(GreenLight, 255);
    analogWrite(RedLight, 0);
  }
  else if (1000 > Calibrated_Weight && Calibrated_Weight > 501) {
    analogWrite(BlueLight, 255);
    analogWrite(GreenLight, 0);
    analogWrite(RedLight, 0);
  }
  else if (Calibrated_Weight > 1001) {
    analogWrite(BlueLight, 0);
    analogWrite(GreenLight, 0);
    analogWrite(RedLight, 255);
  }

  // print the results to the serial monitor:
  Serial.print("Rear Left = " );
  Serial.print(RearLeftValue);
  Serial.print("\tRear Center = " );
  Serial.print(RearCenterValue);
  Serial.print("\tRear Right = " );
  Serial.print(RearRightValue);
  Serial.print("\tFront Left = " );
  Serial.print(FrontLeftValue);
  Serial.print("\tFront Center = " );
  Serial.print(FrontCenterValue);
  Serial.print("\tFront Right = " );
  Serial.print(FrontRightValue);
  Serial.println("\tCalibrated Weight = " );
  Serial.println(Calibrated_Weight);

  // wait 2 milliseconds before the next loop
  // for the analog-to-digital converter to settle
  // after the last reading:
  delay(200);
}

float readWeight() {
  // read the analog in values:
  RearLeftValue = analogRead(RearLeftCell) + 10;   // Read in Left Analog Sensor value and manually calibrate DC offset
  RearCenterValue = analogRead(RearCenterCell) + 49; // Read in Right Analog Sensor value and manually calibrate DC offset
  RearRightValue = analogRead(RearRightCell) + 170;   // Read in Left Analog Sensor value and manually calibrate DC offset
  FrontLeftValue = analogRead(FrontLeftCell) + 355; // Read in Right Analog Sensor value and manually calibrate DC offset
  FrontCenterValue = analogRead(FrontCenterCell) + 93;   // Read in Left Analog Sensor value and manually calibrate DC offset
  FrontRightValue = analogRead(FrontRightCell) + 365; // Read in Right Analog Sensor value and manually calibrate DC offset

  TotalWeight = RearCenterValue + RearLeftValue + RearRightValue + FrontLeftValue + FrontCenterValue + FrontRightValue;   // Calculate total weight
  return TotalWeight;
}
// Tyler Kugler
// WeighTrack 2.0
// Senior Thesis
// April 2, 2015

// Complete Functional Code

// RFID tag capture code framework from MIT opensource.com

// Assign Constant Variable

// Load Cell Constants
const int RearLeftCell = A3;  // Analog input pin that left sensor is attached to
const int RearCenterCell = A2; // Analog input pin that right sensor is attached to
const int RearRightCell = A5;  // Analog input pin that left sensor is attached to
const int FrontLeftCell = A0; // Analog input pin that right sensor is attached to
const int FrontCenterCell = A1;  // Analog input pin that left sensor is attached to
const int FrontRightCell = A4; // Analog input pin that right sensor is attached to

// Indicator Lights
const int GreenLight = 9;
const int RedLight = 10;
const int BlueLight = 11;

// RFID Constants

//RFID Reset
const int RFIDResetPin = 8;
//Register your RFID tags here
char tag1[13] = "7200773A645B"; // Tag 1
char tag2[13] = "72007749F2BE"; // Tag 2
char tag3[13] = "720077451555"; // Tag 3
char tag4[13] = "7200771E312A"; // Tag 4
char tag5[13] = "72007753D482"; // Tag 5
char tag6[13] = "720077518BDE"; // Tag 6
char tag7[13] = "7200775CF1A8"; // Tag 7
char tag8[13] = "110069192140"; // Tag 8 (Water)

// Bottles
int Magic_FULL = 665; // Full Weight of Bottle
int Water_FULL = 940; // Full Weight of Bottle
int SIAL1_FULL = 110; // Full Weight of Bottle
int SIAL2_FULL = 665; // Full Weight of Bottle
int SIAL3_FULL = 665; // Full Weight of Bottle
int SIAL4_FULL = 570; // Full Weight of Bottle
int SIAL5_FULL = 570; // Full Weight of Bottle
int SIAL6_FULL = 570; // Full Weight of Bottle

// Load Cells
int RearLeftValue = 0;       // value read from rear left sensor
int RearCenterValue = 0;      // value read from rear center sensor
int RearRightValue = 0;       // value read from rear right sensor
int FrontLeftValue = 0;       // value read from front left sensor
int FrontCenterValue = 0;     // value read from front center sensor
int FrontRightValue = 0;      // value read from front right sensor
int TotalWeight = 0;           // total weight read in from both sensors
int CalibratedWeight = 0;
int Bottle_Number = 0;
int Weight_Change = 0;
float Percent_Full = 0;
float Next_Weight = 0;
int Next_Weight1 = 0;
int Next_Weight2 = 0;
int Next_Weight3 = 0;
int Next_Weight4 = 0;
int Next_Weight5 = 0;
int Next_Weight6 = 0;
int Next_Weight7 = 0;
int Next_Weight8 = 0;
int Next_Weight9 = 0;

int Light_Trigger = 10000;
boolean Recall_Check = false;
boolean Expire_Check = false;

// Calibration
int initial_calibration1 = 0;
int initial_calibration2 = 0;
int initial_calibration3 = 0;
int initial_calibration4 = 0;
int initial_calibration = 0;

// Array Database [Bottle #][Good = 0, Approaching = 1, Expired = 2][Full_Weight][Current_Weight][Recall]
int BottleArray[8][5] = {
  {1,0,Magic_FULL,0,0},
  {2,0,SIAL1_FULL,0,0},
  {3,0,SIAL2_FULL,0,0},
  {4,1,SIAL3_FULL,0,0},
  {5,2,SIAL4_FULL,0,0},
  {6,0,SIAL5_FULL,0,1},
  {7,2,SIAL6_FULL,0,1},
  {8,0,Water_FULL,0,0}
};

void setup() {
    light_setup();
    initial_calibration1 = calibrateWeight();
    delay(10);
    initial_calibration2 = calibrateWeight();
    delay(10);
    initial_calibration3 = calibrateWeight();
    delay(10);
    initial_calibration4 = calibrateWeight();
    initial_calibration = (initial_calibration1 + initial_calibration2 +
initial_calibration3 + initial_calibration4) / 4;

    // initialize serial communications at 9600 bps:
    //Serial.begin(9600);
    Serial1.begin(9600);
    // Set RFID Reset High
    pinMode(RFIDResetPin, OUTPUT);
    digitalWrite(RFIDResetPin, HIGH);

    // Indicator Lights
    pinMode(GreenLight, OUTPUT);
    pinMode(RedLight, OUTPUT);
    pinMode(BlueLight, OUTPUT);
    analogWrite(BlueLight, 200);
    analogWrite(GreenLight, 40);
    analogWrite(RedLight, 0);
}

void loop() {
    Current_Weight = readWeight();    // Read in position of weight source
    //Serial.println(Current_Weight);

    if (Light_Trigger < 6){
        // Light up Green
        analogWrite(BlueLight, 0);
        analogWrite(GreenLight, 255);
        analogWrite(RedLight, 0);
        Light_Trigger++;
    }
if (Light_Trigger == 6) {
    // Fade out light
    fade_g2b();
    Light_Trigger++;
}

if (Light_Trigger > 6) {
    // Turn off Light
    //fade_g2b();
    analogWrite(BlueLight, 200);
    analogWrite(GreenLight, 40);
    analogWrite(RedLight, 0);
}

// RFID ACTION BEGIN ///////////////////////////////////////////
char tagString[13];
int index = 0;
boolean reading = false;

while(Serial1.available()){

    int readByte = Serial1.read(); //read next available byte
    if(readByte == 2) reading = true; //beginning of tag
    if(readByte == 3) reading = false; //end of tag
    if(reading && readByte != 2 && readByte != 10 && readByte != 13){
        //store the tag
        tagString[index] = readByte;
        index ++;
    }
}

Bottle_Number = checkTag(tagString); // Check if it is a match

if (Bottle_Number == 0) {
    clearTag(tagString); // Clear the char of all value
    resetReader(); // reset the RFID reader
    return;
}

// Check for Recalled Contents
Recall_Check = recall_check(Bottle_Number);
if (Recall_Check == true){
    Recall_Check = false;
    Bottle_Number = 0;
    clearTag(tagString); //Clear the char of all value
    resetReader(); //reset the RFID reader
    return;
}

// Check for Expired Contents
Expire_Check = expire_check(Bottle_Number);
if (Expire_Check == true){
    Expire_Check = false;
    Bottle_Number = 0;
    clearTag(tagString); //Clear the char of all value
    resetReader(); //reset the RFID reader
    return;
}

clearTag(tagString); //Clear the char of all value
resetReader(); //reset the RFID reader
// Serial.print(Bottle_Number);
if (Bottle_Number != 0){
    Light_Trigger = 1;
    analogWrite(BlueLight, 0);
    analogWrite(GreenLight, 255);
    analogWrite(RedLight, 0);

    // Allow for Weight Change to occur
    delay(1500);

    // Take in Next_Weight Measurement
    Next_Weight1 = readWeight();
    delay(10);
    Next_Weight2 = readWeight();
    delay(10);
    Next_Weight3 = readWeight();
    delay(10);
    Next_Weight4 = readWeight();
    delay(10);
    Next_Weight5 = readWeight();
    delay(10);
    Next_Weight6 = readWeight();
    delay(10);
    Next_Weight7 = readWeight();
    delay(10);
    Next_Weight8 = readWeight();
    delay(10);
    Next_Weight9 = readWeight();

    Next_Weight = (Next_Weight1 + Next_Weight2 + Next_Weight3 + Next_Weight4 +
                   Next_Weight5 + Next_Weight6 + Next_Weight7 + Next_Weight8 + Next_Weight9 ) / 9;

    //Serial.println(Next.Weight);
    // If the next weight is 50 greater than the current weight
    Weight_Change = Next_Weight - Current_Weight;
    // add the weight change to column for of Bottle Array
    //Serial.print("Current Weight:");
    //Serial.println(BottleArray[Bottle_Number - 1][3]);
    //Serial.print("New Current Weight:");
    //Serial.println(BottleArray[Bottle_Number - 1][3]);
    //Serial.print("Full Weight:");
    //Serial.println(BottleArray[Bottle_Number - 1][2]);
    Percent_Full = ( float(BottleArray[Bottle_Number - 1][3]) / BottleArray[Bottle_Number - 1][2] ) * 100;
    if (Percent_Full > 97){
        Percent_Full = 100;
    }
    if (Percent_Full < 1){
        Percent_Full = 0;
    }

    Serial.print("Bottle Number ");
    Serial.print(Bottle_Number);
    Serial.print(" is ");
    Serial.print(Percent_Full);
    Serial.println(" percent full.");
    //Serial.println(BottleArray[Bottle_Number - 1][3]);
    BottleArray[Bottle_Number - 1][3] = 0;
if (Light_Trigger < 5){
    // Light up Green
    analogWrite(BlueLight, 0);
    analogWrite(GreenLight, 255);
    analogWrite(RedLight, 0);
    Light_Trigger++;
}

if (Light_Trigger == 5){
    // Fade out light
    fade_g2b();
    Light_Trigger++;
}

if (Light_Trigger > 5){
    // Turn off Light
    //fade_g2b();
    analogWrite(BlueLight, 200);
    analogWrite(GreenLight, 40);
    analogWrite(RedLight, 0);
}

// RFID ACTION END ///////////////////////////////////////////////////////////////////

// wait 2 milliseconds before the next loop
// for the analog-to-digital converter to settle
// after the last reading:

int calibrateWeight() {
    // read the analog in values:
    RearLeftValue = analogRead(RearLeftCell) + 10;   // Read in Left Analog Sensor value
    and manually calibrate DC offset
    RearCenterValue = analogRead(RearCenterCell) + 49; // Read in Right Analog Sensor value
    and manually calibrate DC offset
    RearRightValue = analogRead(RearRightCell) + 170;   // Read in Left Analog Sensor value
    and manually calibrate DC offset
    FrontLeftValue = analogRead(FrontLeftCell) + 355; // Read in Right Analog Sensor value
    and manually calibrate DC offset
    FrontCenterValue = analogRead(FrontCenterCell) + 93;   // Read in Left Analog Sensor value
    and manually calibrate DC offset
    FrontRightValue = analogRead(FrontRightCell) + 365; // Read in Right Analog Sensor value
    and manually calibrate DC offset

    TotalWeight = RearCenterValue + RearLeftValue + RearRightValue + FrontLeftValue +
    FrontCenterValue + FrontRightValue;      // Calculate total weight
    CalibratedWeight = TotalWeight;
    return CalibratedWeight;
}

int readWeight() {
    // read the analog in values:
    RearLeftValue = analogRead(RearLeftCell) + 10;   // Read in Left Analog Sensor value
    and manually calibrate DC offset
    RearCenterValue = analogRead(RearCenterCell) + 49; // Read in Right Analog Sensor value
    and manually calibrate DC offset
    RearRightValue = analogRead(RearRightCell) + 170;   // Read in Left Analog Sensor value
    and manually calibrate DC offset
    FrontLeftValue = analogRead(FrontLeftCell) + 355; // Read in Right Analog Sensor value
    and manually calibrate DC offset
    FrontCenterValue = analogRead(FrontCenterCell) + 93;   // Read in Left Analog Sensor value
    and manually calibrate DC offset
    FrontRightValue = analogRead(FrontRightCell) + 365; // Read in Right Analog Sensor value
    and manually calibrate DC offset
value and manually calibrate DC offset

    TotalWeight = RearCenterValue + RearLeftValue + RearRightValue + FrontLeftValue + FrontCenterValue + FrontRightValue; // Calculate total weight
    CalibratedWeight = TotalWeight - initial_calibration;
    return CalibratedWeight;
}

int checkTag(char tag[]){
    //////////////////////////////////////////////////////////////////////////
    //Check the read tag against known tags
    //////////////////////////////////////////////////////////////////////////

    //Serial.println(tag); //read out any tag
    if(strlen(tag) == 0) return 0; //empty, no need to continue
    if(compareTag(tag, tag1)){ // if matched tag1, do this
        return 1;
    }
    else if(compareTag(tag, tag2)){ //if matched tag2, do this
        return 2;
    }
    else if(compareTag(tag, tag3)){
        return 3;
    }
    else if(compareTag(tag, tag4)){
        return 4;
    }
    else if(compareTag(tag, tag5)){
        return 5;
    }
    else if(compareTag(tag, tag6)){
        return 6;
    }
    else if(compareTag(tag, tag7)){
        return 7;
    }
    else if(compareTag(tag, tag8)){
        return 8;
    }
    else{
        //Serial.println(tag); //read out any unknown tag
        return 0;
    }
}

void resetReader(){
    //////////////////////////////////////////////////////////////////////////
    //Reset the RFID reader to read again.
    //////////////////////////////////////////////////////////////////////////
    digitalWrite(RFIDResetPin, LOW);
    digitalWrite(RFIDResetPin, HIGH);
    delay(450);
}

void clearTag(char one[]){
    //////////////////////////////////////////////////////////////////////////
    //clear the char array by filling with null - ASCII 0
    //////////////////////////////////////////////////////////////////////////

}
for(int i = 0; i < strlen(one); i++){
    one[i] = 0;
}
}

boolean compareTag(char one[], char two[]){
    //compare two value to see if same,
    //strcmp not working 100% so we do this
    if(strlen(one) == 0) return false; //empty
    for(int i = 0; i < 12; i++){
        if(one[i] != two[i]) return false;
    }
    return true; //no mismatches
}

void light_setup(){
    int r, g, b;
    int FADESPEED = 3;
    // fade from blue to violet
    for (r = 0; r < 256; r++) {
        analogWrite(RedLight, r);
        delay(FADESPEED);
    }
    // fade from violet to red
    for (b = 255; b > 0; b--){
        analogWrite(BlueLight, b);
        delay(FADESPEED);
    }
    // fade from red to yellow
    for (g = 0; g < 256; g++){
        analogWrite(GreenLight, g);
        delay(FADESPEED);
    }
    // fade from yellow to green
    for (r = 255; r > 0; r--){
        analogWrite(RedLight, r);
        delay(FADESPEED);
    }
    // fade from green to teal
    for (b = 0; b < 200; b++){
        analogWrite(BlueLight, b);
        delay(FADESPEED);
    }
    // fade from teal to blue
    for (g = 255; g > 40; g--){
        analogWrite(GreenLight, g);
        delay(FADESPEED);
    }
}

void fade_g2b(){
    int r, g, b;
    int FADESPEED = 3;
    r = 0;
    // fade from teal to blue
    for (g = 255; g > 40; g--){
        analogWrite(GreenLight, g);
        delay(FADESPEED);
    }
    // fade from green to teal
    for (b = 0; b < 200; b++){
        analogWrite(BlueLight, b);
        delay(FADESPEED);
    }
}
boolean recall_check(int bottle) {
    if (BottleArray[bottle - 1][4] == 1) {
        for (int i = 0; i < 20; i++) {
            analogWrite(BlueLight, 0);
            analogWrite(GreenLight, 0);
            analogWrite(RedLight, 255);
            delay(50);
            analogWrite(BlueLight, 0);
            analogWrite(GreenLight, 0);
            analogWrite(RedLight, 0);
            delay(50);
        }
        Serial.print("Bottle ");
        Serial.print(bottle);
        Serial.println(" has been recalled, DO NOT CONSUME!");
        analogWrite(BlueLight, 0);
        analogWrite(GreenLight, 0);
        analogWrite(RedLight, 255);
        delay(1500);
        analogWrite(BlueLight, 200);
        analogWrite(GreenLight, 40);
        analogWrite(RedLight, 0);
        return true;
    } else return false;
}

// Good = 0, Approaching = 1, Expired = 2
boolean expire_check(int bottle) {
    // Red if Expired
    if (BottleArray[bottle - 1][1] == 2) {
        for (int i = 0; i < 4; i++) {
            analogWrite(BlueLight, 0);
            analogWrite(GreenLight, 0);
            analogWrite(RedLight, 255);
            delay(500);
            analogWrite(BlueLight, 0);
            analogWrite(GreenLight, 0);
            analogWrite(RedLight, 0);
            delay(500);
        }
        Serial.print("Bottle ");
        Serial.print(bottle);
        Serial.println(" is expired.");
        analogWrite(BlueLight, 0);
        analogWrite(GreenLight, 0);
        analogWrite(RedLight, 255);
        delay(1000);
        analogWrite(BlueLight, 200);
        analogWrite(GreenLight, 40);
        analogWrite(RedLight, 0);
        return true;
    } else if (BottleArray[bottle - 1][1] == 1) {
        for (int i = 0; i < 4; i++) {
            analogWrite(BlueLight, 0);
            analogWrite(GreenLight, 200);
            analogWrite(RedLight, 200);
            delay(500);
            analogWrite(BlueLight, 0);
            analogWrite(GreenLight, 200);
            analogWrite(RedLight, 200);
            delay(500);
Serial.print("Bottle ");
Serial.print(bottle);
Serial.println(" is approaching expiration.");
analogWrite(BlueLight, 0);
analogWrite(GreenLight, 200);
analogWrite(RedLight, 200);
delay(1000);
//analogWrite(BlueLight, 200);
//analogWrite(GreenLight, 40);
//analogWrite(RedLight, 0);
return false;
}
// Return Safe
else return false;
Appendix F – WeighTrack 2.0 Warning System

Multi-Color Cycle – System Calibrating

Blue – Idle State

Green – All Clear, No Warnings

Yellow Flashing (Medium) – Approaching Expiration
Red Flashing (Medium) – Bottle Expired

Red Flashing (Rapid) – Bottle has been Recalled, Do not consume!
Bibliography


